

Open Group Standard

Base Specifications, Issue 7, 2013 Edition

IEEE Computer Society

and

The Open Group



[This page intentionally blank]

IEEE Std 1003.1™, 2013 Edition

The Open Group Technical Standard
Base Specifications, Issue 7
incorporates IEEE Std 1003.1™-2008, IEEE Std 1003.1™-2008/Cor 1-2013

Standard for Information Technology— Portable Operating System Interface (POSIX®)

Base Specifications, Issue 7

Sponsor

Portable Applications Standards Committee

of the

IEEE Computer Society

and

The Open Group



Abstract

POSIX.1-2008 is simultaneously IEEE Std 1003.1™-2008 and The Open Group Technical Standard Base Specifications, Issue 7. This 2013 Edition includes IEEE Std 1003.1-2008/Cor 1-2013 incorporated into IEEE Std 1003.1-2008 (the base document). The Technical Corrigendum addresses problems discovered since the approval of IEEE Std 1003.1-2008.

POSIX.1-2008 defines a standard operating system interface and environment, including a command interpreter (or “shell”), and common utility programs to support applications portability at the source code level. POSIX.1-2008 is intended to be used by both application developers and system implementors and comprises four major components (each in an associated volume):

- General terms, concepts, and interfaces common to all volumes of this standard, including utility conventions and C-language header definitions, are included in the Base Definitions volume.
- Definitions for system service functions and subroutines, language-specific system services for the C programming language, function issues, including portability, error handling, and error recovery, are included in the System Interfaces volume.
- Definitions for a standard source code-level interface to command interpretation services (a “shell”) and common utility programs for application programs are included in the Shell and Utilities volume.
- Extended rationale that did not fit well into the rest of the document structure, which contains historical information concerning the contents of POSIX.1-2008 and why features were included or discarded by the standard developers, is included in the Rationale (Informative) volume.

The following areas are outside the scope of POSIX.1-2008:

- Graphics interfaces
- Database management system interfaces
- Record I/O considerations
- Object or binary code portability
- System configuration and resource availability

POSIX.1-2008 describes the external characteristics and facilities that are of importance to application developers, rather than the internal construction techniques employed to achieve these capabilities. Special emphasis is placed on those functions and facilities that are needed in a wide variety of commercial applications.

Keywords

application program interface (API), argument, asynchronous, basic regular expression (BRE), batch job, batch system, built-in utility, byte, child, command language interpreter, CPU, extended regular expression (ERE), FIFO, file access control mechanism, IEEE 1003.1™, input/output (I/O), job control, network, parent, portable operating system interface (POSIX®), shell, stream, string, synchronous, system, thread, X/Open System Interface (XSI)

The Institute of Electrical and Electronics Engineers, Inc.
3 Park Avenue, New York, NY 10016-5997, USA

The Open Group
Apex Plaza, Forbury Road, Reading, Berkshire RG1 1AX, U.K.

Copyright © 2013 by The Institute of Electrical and Electronics Engineers, Inc. and The Open Group.
All rights reserved.

Published 19 April 2013 by the IEEE.

PDF: ISBN 978-0-7381-8331-2 STD98188

Print: ISBN 978-0-7381-8333-6 STDPD98188

Published 19 April 2013 by The Open Group.

Doc. Number: C138

ISBN: 1-937218-28-7

IEEE is a registered trademark in the U.S. Patent & Trademark Office, owned by The Institute of Electrical and Electronics Engineers, Incorporated.

POSIX is a registered trademark of IEEE.

This standard has been prepared by the Austin Group. Feedback relating to the material contained within this standard may be submitted by using the Austin Group web site at www.opengroup.org/austin/defectform.html.

IEEE prohibits discrimination, harassment, and bullying. For more information, visit www.ieee.org/web/aboutus/whatis/policies/p9-26.html.

No part of this publication may be reproduced in any form, in an electronic retrieval system or otherwise, without the prior written permission of the publisher. Permission to reproduce all or any part of this standard must be with the consent of both copyright holders and may be subject to a license fee. Both copyright holders will need to be satisfied that the other has granted permission. Requests should be sent by email to austin-group-permissions@opengroup.org.

Please be advised IEEE Std 1003.1, 2013 Edition is for convenience of the user only. It does not supersede the individual published documents listed below:

IEEE Std 1003.1™-2008, Standard for Information Technology—Portable Operating System Interface (POSIX®)

IEEE Std 1003.1™-2008/Cor 1-2013, IEEE Standard for Information Technology—Portable Operating System Interface (POSIX®), Technical Corrigendum 1

Notice and Disclaimer of Liability Concerning the Use of IEEE Documents: IEEE Standards documents are developed within the IEEE Societies and the Standards Coordinating Committees of the IEEE Standards Association (IEEE-SA) Standards Board. IEEE develops its standards through a consensus development process, approved by the American National Standards Institute, which brings together volunteers representing varied viewpoints and interests to achieve the final product. Volunteers are not necessarily members of the Institute and serve without compensation. While IEEE administers the process and establishes rules to promote fairness in the consensus development process, IEEE does not independently evaluate, test, or verify the accuracy of any of the information or the soundness of any judgments contained in its standards.

Use of an IEEE Standard is wholly voluntary. IEEE disclaims liability for any personal injury, property or other damage, of any nature whatsoever, whether special, indirect, consequential, or compensatory, directly or indirectly resulting from the publication, use of, or reliance upon any IEEE Standard document.

IEEE does not warrant or represent the accuracy or content of the material contained in its standards, and expressly disclaims any express or implied warranty, including any implied warranty of merchantability or fitness for a specific purpose, or that the use of the material contained in its standards is free from patent infringement. IEEE Standards documents are supplied “**AS IS**”.

The existence of an IEEE Standard does not imply that there are no other ways to produce, test, measure, purchase, market, or provide other goods and services related to the scope of the IEEE standard. Furthermore, the viewpoint expressed at the time a standard is approved and issued is subject to change brought about through developments in the state of the art and comments received from users of the standard. Every IEEE standard is subjected to review at least every ten years. When a document is more than ten years old and has not undergone a revision process, it is reasonable to conclude that its contents, although still of some value, do not wholly reflect the present state of the art. Users are cautioned to check to determine that they have the latest edition of any IEEE standard.

In publishing and making its standards available, IEEE is not suggesting or rendering professional or other services for, or on behalf of, any person or entity. Nor is IEEE undertaking to perform any duty owed by any other person or entity to another. Any person utilizing any IEEE Standards document should rely upon his or her own independent judgment in the exercise of reasonable care in any given circumstances or, as appropriate, seek the advice of a competent professional in determining the appropriateness of a given IEEE standard.

Translations: The IEEE consensus development process involves the review of documents in English only. In the event that an IEEE standard is translated, only the English version published by IEEE should be considered the approved IEEE standard.

Official Statements: A statement, written or oral, that is not processed in accordance with the IEEE-SA Standards Board Operations Manual shall not be considered the official position of IEEE or any of its committees and shall not be considered to be, nor be relied upon as, a formal position of IEEE. At lectures, symposia, seminars, or educational courses, an individual presenting information on IEEE standards shall make it clear that his or her views should be considered the personal views of that individual rather than the formal position of IEEE.

Comments on Standards: Comments for revision of IEEE Standards documents are welcome from any interested party, regardless of membership affiliation with IEEE. However, IEEE does not provide consulting information or advice pertaining to IEEE Standards documents. Suggestions for changes in documents should be in the form of a proposed change of text, together with appropriate supporting comments. Since IEEE standards represent a consensus of concerned interests, it is important to ensure that any responses to comments and questions also receive the concurrence of a balance of interests. For this reason, IEEE and the members of its societies and Standards Coordinating Committees are not able to provide an instant response to comments or questions except in those cases where the matter has previously been addressed. Any person who would like to participate in evaluating comments or revisions to an IEEE standard is welcome to join the relevant IEEE working group at <http://standards.ieee.org/develop/wg/>.

Comments on standards should be submitted to the following address:

Secretary, IEEE-SA Standards Board
445 Hoes Lane
Piscataway, NJ 08854
USA

Photocopies: Authorization to photocopy portions of any individual standard for internal or personal use is granted by The Institute of Electrical and Electronics Engineers, Inc., provided that the appropriate fee is paid to the Copyright Clearance Center. To arrange for payment of the licensing fee, please contact Copyright Clearance Center, Customer Service, 222 Rosewood Drive, Danvers, MA 01923 USA; +1 978 750 8400. Permission to photocopy portions of any individual standard for educational classroom use can also be obtained through the Copyright Clearance Center.

The Open Group

The Open Group is a global consortium that enables the achievement of business objectives through IT standards. With more than 400 member organizations, The Open Group has a diverse membership that spans all sectors of the IT community – customers, systems and solutions suppliers, tool vendors, integrators, and consultants, as well as academics and researchers – to:

- Capture, understand, and address current and emerging requirements, and establish policies and share best practices
- Facilitate interoperability, develop consensus, and evolve and integrate specifications and open source technologies
- Offer a comprehensive set of services to enhance the operational efficiency of consortia
- Operate the industry's premier certification service

Further information on The Open Group is available at www.opengroup.org.

The Open Group publishes a wide range of technical documentation, most of which is focused on development of Open Group Standards and Guides, but which also includes white papers, technical studies, certification and testing documentation, and business titles. Full details and a catalog are available at www.opengroup.org/bookstore.

Readers should note that updates – in the form of Corrigenda – may apply to any publication. This information is published at www.opengroup.org/corrigenda.

Notice to users

Laws and regulations

Users of IEEE Standards documents should consult all applicable laws and regulations. Compliance with the provisions of any IEEE Standards document does not imply compliance to any applicable regulatory requirements. Implementers of the standard are responsible for observing or referring to the applicable regulatory requirements. IEEE does not, by the publication of its standards, intend to urge action that is not in compliance with applicable laws, and these documents may not be construed as doing so.

Copyrights

This document is copyrighted by the IEEE and The Open Group. It is made available for a wide variety of both public and private uses. These include both use, by reference, in laws and regulations, and use in private self-regulation, standardization, and the promotion of engineering practices and methods. By making this document available for use and adoption by public authorities and private users, the IEEE and The Open Group do not waive any rights in copyright to this document.

Updating of IEEE documents

Users of IEEE Standards documents should be aware that these documents may be superseded at any time by the issuance of new editions or may be amended from time to time through the issuance of amendments, corrigenda, or errata. An official IEEE document at any point in time consists of the current edition of the document together with any amendments, corrigenda, or errata then in effect. In order to determine whether a given document is the current edition and whether it has been amended through the issuance of amendments, corrigenda, or errata, visit the IEEE-SA website at <http://standards.ieee.org/index.html> or contact the IEEE at the address listed previously. For more information about the IEEE Standards Association or the IEEE standards development process, visit the IEEE-SA website at <http://standards.ieee.org/index.html>.

Errata

Errata, if any, for this and all other standards can be accessed at the following URL: <http://standards.ieee.org/findstds/errata/index.html>. Users are encouraged to check this URL for errata periodically.

Patents

Attention is called to the possibility that implementation of this standard may require use of subject matter covered by patent rights. By publication of this standard, no position is taken by the IEEE with respect to the existence or validity of any patent rights in connection therewith. If a patent holder or patent applicant has filed a statement of assurance via an Accepted Letter of Assurance, then the statement is listed on the IEEE-SA website at <http://standards.ieee.org/about/sasb/patcom/patents.html>. Letters of Assurance may indicate whether the Submitter is willing or unwilling to grant licenses under patent rights without compensation or under reasonable rates, with reasonable terms and conditions that are demonstrably free of any unfair discrimination to applicants desiring to obtain such licenses.

Essential Patent Claims may exist for which a Letter of Assurance has not been received. The IEEE is not responsible for identifying Essential Patent Claims for which a license may be required, for conducting inquiries into the legal validity or scope of Patents Claims, or determining whether any licensing terms or conditions provided in connection with submission of a Letter of Assurance, if any, or in any licensing agreements are reasonable or non-discriminatory. Users of this standard are expressly advised that determination of the validity of any patent rights, and the risk of infringement of such rights, is entirely their own responsibility. Further information may be obtained from the IEEE Standards Association.

Participants

The Austin Group

IEEE Std 1003.1-2008 was prepared by the Austin Group, sponsored by the Portable Applications Standards Committee of the IEEE Computer Society, The Open Group, and ISO/IEC JTC 1/SC 22. At the time of approval, the membership of the Austin Group was as follows:

Andrew Josey, *Chair*

Donald W. Cragun, *Organizational Representative, IEEE PASC*

Nicholas Stoughton, *Organizational Representative, ISO/IEC JTC 1/SC 22*

Mark S. Brown, *Organizational Representative, The Open Group*

Cathy Fox, *Technical Editor*

Austin Group Technical Reviewers

Theodore P. Baker	Hallvard B. Furuseth	Stephen Michell
Eric Blake	Michael Gonzalez Harbour	Matthew Rice
Andries E. Brouwer	Joseph M. Gwinn	Joerg Schilling
Mark S. Brown	Bruno Haible	Stephen Schwarm
David Butenhof	Mark Harris	Konrad A. Schwarz
Geoff Clare	Andrew Josey	Keld Simonsen
David N. Clissold	Michael Kerrisk	Nicholas Stoughton
Donald W. Cragun	David Korn	Alexander Terekhov
Lawrence D.K.B. Dwyer	Vincent Lefèvre	William L. Toth
Cynthia Eastham	Wojtek Lerch	Fred J. Tydeman
Paul Eggert	Evan Liebovitch	Mats D. Wichmann
Clive D.W. Feather	Jens Maurer	Garrett Wollman
Glenn Fowler	Roland McGrath	James Youngman
Cathy Fox	Jim Meyering	

Working Group Members

William J. Ackerman	Karen Gordon	Jose Puthenkulam
Adewole Akpose	Randall Groves	G.K. Rajani
Butch Anton	Scott Gudgel	Chet Ramey
Peter Anvin	Joseph M. Gwinn	Matthew Rice
Jay Ashford	Charles Hammons	Gunnar Ritter
Theodore P. Baker	Mark Harris	Robert Robinson
Hugh Barrass	Barry Hedquist	Curtis Royster Jr.
William Bartholomew	Werner Hoelzl	Eusebio Rufian
Andy Bihain	David Hopwood	Wolfgang Sanow
Eric Blake	David P. Howell	Bartien Sayogo
Xie Bo	James Isaak	Nico Schottelius
Tomo Bogataj	Michael Jones	Stephen Schwarm
Walter Briscoe	Jaimon C. Jose	Konrad A. Schwarz
Andries E. Brouwer	Andrew Josey	Glen Seeds
Mark S. Brown	Piotr Karocki	M. Sekiguchi
David Butenhof	Sujit Madhavan Karataparambil	Al Simons
Juan Carreon	Michael Kerrisk	Keld Simonsen
Danila Chernetsov	Alexey V. Khoroshilov	Antonio Soares
Keith Chow	Akio Kido	Alexander A. Sortov
Jimmy Chun	William King	Thomas Starai
Geoff Clare	Klaus Klein	Nicholas Stoughton
Tommy Cooper	David Korn	Ienup Sung
Jeffrey L. Copeland	Wayne Krone	Alfred Mieczyslaw Szmidi

Donald W. Cragun	Thomas Kurihara	Alexander Terekhov
Dragan S. Cvetkovic	Susan Land	Donn Terry
Lee Damico	Kenneth Lang	Michael Thomadakis
Juan A. de la Puente	Shawn Leard	John Thywissen
David Dice	Solomon Lee	Leon Toh
Thomas Dickey	Fernando Lucas Rodriguez	William L. Toth
Loic Domaigne	William Lumpkins	Miloslav Trmač
Steven J. Dovich	Scott Lurndal	Fred J. Tydeman
Ulrich Drepper	Lieven Marchand	Mark-Rene Uchida
Sourav Dutta	Roger Marquis	Steve Valentine
Lawrence D.K.B. Dwyer	Roger J. Martin	Peter Van Der Veen
Paul Eggert	Jack Mccann	Fred Web
Daniel M. Eischen	Mick Meaden	Tao Weijian
Marc Emmelmann	Jim Meyering	David Wexelblat
Laura Fairhead	Gary Michel	Mats D. Wichmann
Joanna Farley	Hai Ming Li	Robert Wilkens
Clive D.W. Feather	Wilhelm Mueller	Martin Wille
John Fendrich	Bruce Muschlitz	Matt Wilson
Yaacov Fenster	Erik Naggum	Michael Wilson
Andrew Fieldsend	Gergely Nagy	Jim Wise
Glenn Fowler	Michael S. Newman	Song Wonchang
Cathy Fox	Liviu Nicoara	Shao A. Wu
Jesus Fuente	Robert Nordier	Oren Yuen
Mark Funkenhauser	Peter Petrov	Janusz Zalewski
Michael Geipel	Frank Prindle	Jim Zepeda
Michael Gonzalez Harbour	Vikram Punj	Jason Zions

The Open Group

When The Open Group approved the Base Specifications, Issue 7 on 24 July 2008, the membership of the Base Working Group was as follows:

Andrew Josey, *Chair*

Mark S. Brown, *Vice Chair and Austin Group Liaison*

Cathy Fox, *Technical Editor*

Base Working Group Members

Basabi Bhattacharya	Donald W. Cragun	Robbin Kawabata
Mark S. Brown	Ulrich Drepper	Craig Mohrman
David Butenhof	Lawrence D.K.B. Dwyer	Nicholas Stoughton
April Chin	Cynthia Eastham	William L. Toth
Geoff Clare	Roger Faulkner	Kenjiro Tsuji
David N. Clissold	Carol Fields	Kevin Van Vechten

IEEE

When the IEEE Standards Board approved IEEE Std 1003.1-2008 on 26 September 2008, the membership of the committees was as follows:

Portable Application Standards Committee (PASC)

Lowell G. Johnson, *Chair*

Joseph M. Gwinn, *Vice Chair*

Jay Ashford, *Functional Chair (Balloting)*

Andrew Josey, *Functional Chair (Interpretations)*

Curtis Royster Jr., *Functional Chair (Logistics)*

Nicholas Stoughton, *Secretary*

Balloting Committee

The following members of the individual balloting committee voted on this standard. Balloters may have voted for approval, disapproval, or abstention.

William J. Ackerman	Andrew Fieldsend	Gary Michel
Adewole Akpose	Michael Geipel	Bruce Muschlitz
Butch Anton	Randall Groves	Michael S. Newman
Jay Ashford	Scott Gudgel	Peter Petrov
Hugh Barrass	Joseph M. Gwinn	Vikram Punj
Andy Bihain	Charles Hammons	Jose Puthenkulam
Tomo Bogataj	Michael Gonzalez Harbour	Robert Robinson
Mark S. Brown	Barry Hedquist	Fernando Lucas Rodriguez
Juan Carreon	Werner Hoelzl	Bartien Sayogo
Danila Chernetsov	Andrew Josey	Stephen Schwarm
Keith Chow	Piotr Karocki	Thomas Starai
Tommy Cooper	Thomas Kurihara	John Thywissen
Donald W. Cragun	Susan Land	Mark-Rene Uchida
Ulrich Drepper	Kenneth Lang	Robert Wilkens
Souray Dutta	Shawn Leard	Oren Yuen
Marc Emmelmann	William Lumpkins	Janusz Zalewski
John Fendrich	Roger J. Martin	

IEEE-SA Standards Board

When the IEEE-SA Standards Board approved this standard on 26 September 2008, it had the following membership:

Robert M. Grow, *Chair*

Tom A. Prevost, *Vice Chair*

Steve M. Mills, *Past Chair*

Judith Gorman, *Secretary*

Victor Berman	James Hughes	Chuck Powers
Richard DeBlasio	Richard H. Hulett	Narayanan Ramachandran
Andrew Drozd	Young Kyun Kim	Jon Walter Rosdahl
Mark Epstein	Joseph L. Koepfinger *	Anne-Marie Sahazizian
Alexander Gelman	John Kulick	Malcolm V. Thaden
William R. Goldbach	David J. Law	Howard L. Wolfman
Arnold M. Greenspan	Glenn Parsons	Don Wright
Kenneth S. Hanus	Ronald C. Petersen	

* Member Emeritus

Also included are the following non-voting IEEE-SA Standards Board liaisons:

Satish K. Aggarwal, *NRC Representative*

Michael H. Kelley, *NIST Representative*

Don Messina, *IEEE Standards Program Manager, Document Development*

Michael Kipness, *IEEE Standards Program Manager, Technical Program Development*

IEEE Std 1003.1™-2008/Cor 1-2013 was prepared by the Austin Group, sponsored by the Portable Applications Standards Committee of the IEEE Computer Society, The Open Group, and ISO/IEC JTC 1/SC22.

The Austin Group

At the time this IEEE standard was completed, the Austin Group had the following membership:

Andrew Josey, *Chair*

Donald W. Cragun, *Organizational Representative, IEEE PASC*

Nicholas Stoughton, *Organizational Representative, ISO/IEC JTC 1/SC22*

Mark S. Brown, *Organizational Representative, The Open Group*

Cathy Fox, *Technical Editor*

Austin Group Technical Reviewers

Eric Blake	Ross Johnson	Jörg Schilling
Paolo Bonzini	Andrew Josey	Ed Schouten
Mark S. Brown	Alexey V. Khoroshilov	Konrad A. Schwarz
Geoff Clare	Antoine Leca	Martin Sebor
Donald W. Cragun	Vincent Lefèvre	Nicolas Sitbon
Ulrich Drepper	Matt McCutchen	Nicholas Stoughton
Laszlo Ersek	Christoph Anton Mitterer	Jilles Tjoelker
Hal Finkel	Peter O'Gorman	Nathan Weeks
Mark Harris	Peter Petrov	David A. Wheeler
Tom Honermann	James C. Pugsley	

Austin Group Working Group Members

Hans Aberg	Mike Fryssinger	Peter O'Gorman
Eitan Adler	Pierre Gaston	Peter Petrov
Tanaka Akira	Thorsten Glaser	Wayne Pollock
Matthias Andree	Philip Guenther	James C. Pugsley
Matt W. Benjamin	Bruno Haible	Chet Ramey
Guido Berhoerster	Mark Harris	Xavier Roche
Eric Blake	David Holland	Jeroen Ruigrok van der Werven
Hans Boehm	Tom Honermann	Bart Schaefer
Paolo Bonzini	Ross Johnson	Jörg Schilling
Davide Brini	Andrew Josey	Nico Schottelius
Andries E. Brouwer	Dan Kegel	Ed Schouten
Mark S. Brown	Michael Kerrisk	Konrad A. Schwarz
David Butenhof	Alexey V. Khoroshilov	Jens Schweikhardt
Albert Cahalan	Josh Knight	Martin Sebor
Stephane Chazelas	Jonathan A. Kollasch	Glen Seeds
Geoff Clare	Bruce Korb	Thor Lancelot Simon
Garrett Cooper	David Korn	Keld Simonsen
Donald W. Cragun	Terry Lambert	Nicolas Sitbon
Matthew Dempsky	Antoine Leca	Johannes Sixt
Casper Dik	Vincent Lefèvre	Paul Smith
Niall Douglas	Wojtek Lerch	Nicholas Stoughton

Ulrich Drepper
Lawrence D.K.B. Dwyer
Paul Eggert
Julian Elischer
Laszlo Ersek
Bruce Evans
Sean C. Farley
Hal Finkel
Glenn Fowler

Scott Lurndal
Roger Marquis
Matt McCutchen
Esmaeil Mirzaee
Christoph Anton Mitterer
Joseph S. Myers
Alexander Nasonov
Jonathan Nieder
Carlos O'Donell

Marcel Telka
Jilles Tjoelker
Fred J. Tydeman
Nathan Weeks
David A. Wheeler
Mats D. Wichmann
Garrett Wollman
Jörg Wunsch

The Open Group

When The Open Group approved the Base Specifications, Issue 7, Technical Corrigendum 1 on **25 October 2012**, the membership of The Open Group Base Working Group was as follows:

Andrew Josey, *Chair*
Mark S. Brown, *Austin Group Liaison*
Cathy Fox, *Technical Editor*

Base Working Group Members

Eric Blake	Donald W. Cragun	William L. Toth
Mark S. Brown	Lawrence D.K.B. Dwyer	Kevin Van Vechten
David Butenhof	Darrin Johnson	
Geoff Clare	James C. Pugsley	

Portable Applications Standards Committee (PASC)

Joseph M. Gwinn, *Chair*
Andrew Josey, *Functional Chair (Interpretations)*
Curtis Royster Jr., *Functional Chair (Logistics)*
Nicholas Stoughton, *Secretary*

IEEE

The following members of the individual balloting committee voted on this standard. Balloters may have voted for approval, disapproval, or abstention.

Mark S. Brown	Randall Groves	Peter Petrov
Michael Browne	Joseph M. Gwinn	Bartien Sayogo
Keith Chow	Werner Hoelzl	Stephen Schwarm
Donald W. Cragun	Andrew Josey	Gil Shultz
Thomas Dineen	Piotr Karocki	Walter Struppler
Andrew Fieldsend	Fernando Lucas Rodriguez	Oren Yuen
David Fuschi	Greg Luri	

When the IEEE-SA Standards Board approved this standard on 6 February 2013, it had the following membership:

John Kulick, *Chair*
Richard H. Hulett, *Past Chair*
Konstantinos Karachalios, *Secretary*

Masayuki Ariyoshi	Mark Halpin	Ronald C. Petersen
Peter Balma	Gary Hoffman	Gary Robinson
Farooq Bari	Paul Houzé	Jon Walter Rosdahl
Ted Burse	Jim Hughes	Adrian Stephens
Wael William Diab	Michael Janezic	Peter Sutherland

Stephen Dukes
Jean-Philippe Faure
Alexander Gelman

Joseph L. Koepfinger*
David J. Law
Oleg Logvinov

Yatin Trivedi
Phil Winston
Yu Yuan

*Member Emeritus

Also included are the following nonvoting IEEE-SA Standards Board liaisons:

Richard DeBlasio, *DOE Representative*

Michael Janezic, *NIST Representative*

Don Messina, *IEEE Standards Program Manager, Document Development*

Michael Kipness, *IEEE Standards Program Manager, Technical Program Development*

Introduction

This introduction is not part of IEEE Std 1003.1, 2013 Edition, Standard for Information Technology – Portable Operating System Interface (POSIX).

This standard was developed, and is maintained, by a joint working group of members of the IEEE Portable Applications Standards Committee, members of The Open Group, and members of ISO/IEC Joint Technical Committee 1. This joint working group is known as the Austin Group.^A

The Austin Group arose out of discussions amongst the parties which started in early 1998, leading to an initial meeting and formation of the group in September 1998. The purpose of the Austin Group is to develop and maintain the core open systems interfaces that are the POSIX[®] 1003.1 (and former 1003.2) standards, ISO/IEC 9945, and the core of the Single UNIX Specification.

The approach to specification development has been one of “write once, adopt everywhere”, with the deliverables being a set of specifications that carry the IEEE POSIX designation, The Open Group Technical Standard designation, and an ISO/IEC designation.

This unique development has combined both the industry-led efforts and the formal standardization activities into a single initiative, and included a wide spectrum of participants. The Austin Group continues as the maintenance body for this document.

Anyone wishing to participate in the Austin Group should contact the chair with their request. There are no fees for participation or membership. You may participate as an observer or as a contributor. You do not have to attend face-to-face meetings to participate; electronic participation is most welcome. For more information on the Austin Group and how to participate, see www.opengroup.org/austin.

Background

The developers of POSIX.1-2008 represent a cross-section of hardware manufacturers, vendors of operating systems and other software development tools, software designers, consultants, academics, authors, applications programmers, and others.

Conceptually, POSIX.1-2008 describes a set of fundamental services needed for the efficient construction of application programs. Access to these services has been provided by defining an interface, using the C programming language, a command interpreter, and common utility programs that establish standard semantics and syntax. Since this interface enables application developers to write portable applications – it was developed with that goal in mind – it has been designated POSIX^B, an acronym for Portable Operating System Interface.

Although originated to refer to the original IEEE Std 1003.1-1988, the name POSIX more correctly refers to a *family* of related standards: IEEE Std 1003.*n* and the parts of ISO/IEC 9945. In earlier editions of the IEEE standard, the term POSIX was used as a synonym for IEEE Std 1003.1-1988. A preferred term, POSIX.1, emerged. This maintained the advantages of readability of the symbol “POSIX” without being ambiguous with the POSIX family of standards.

^A The Austin Group is named after the location of the inaugural meeting held at the IBM facility in Austin, Texas in September 1998.

^B The Name POSIX was suggested by Richard Stallman. It is expected to be pronounced *pahz-icks*, as in *positive*, not *poh-six*, or other variations. The pronunciation has been published in an attempt to promulgate a standardized way of referring to a standard operating system interface.

Audience

The intended audience for POSIX.1-2008 is all persons concerned with an industry-wide standard operating system based on the UNIX system. This includes at least four groups of people:

- Persons buying hardware and software systems
- Persons managing companies that are deciding on future corporate computing directions
- Persons implementing operating systems, and especially
- Persons developing applications where portability is an objective

Purpose

Several principles guided the development of POSIX.1-2008:

- **Application-Oriented** – The basic goal was to promote portability of application programs across UNIX system environments by developing a clear, consistent, and unambiguous standard for the interface specification of a portable operating system based on the UNIX system documentation. POSIX.1-2008 codifies the common, existing definition of the UNIX system.
- **Interface, Not Implementation** – POSIX.1-2008 defines an interface, not an implementation. No distinction is made between library functions and system calls; both are referred to as functions. No details of the implementation of any function are given (although historical practice is sometimes indicated in the RATIONALE section). Symbolic names are given for constants (such as signals and error numbers) rather than numbers.
- **Source, Not Object, Portability** – POSIX.1-2008 has been written so that a program written and translated for execution on one conforming implementation may also be translated for execution on another conforming implementation. POSIX.1-2008 does not guarantee that executable (object or binary) code will execute under a different conforming implementation than that for which it was translated, even if the underlying hardware is identical.
- **The C Language** – The system interfaces and header definitions are written in terms of the standard C language as specified in the ISO C standard.
- **No Superuser, No System Administration** – There was no intention to specify all aspects of an operating system. System administration facilities and functions are excluded from this standard, and functions usable only by the superuser have not been included. Still, an implementation of the standard interface may also implement features not in POSIX.1-2008. POSIX.1-2008 is also not concerned with hardware constraints or system maintenance.
- **Minimal Interface, Minimally Defined** – In keeping with the historical design principles of the UNIX system, the mandatory core facilities of POSIX.1-2008 have been kept as minimal as possible. Additional capabilities have been added as optional extensions.
- **Broadly Implementable** – The developers of POSIX.1-2008 endeavored to make all specified functions implementable across a wide range of existing and potential systems, including:
 - All of the current major systems that are ultimately derived from the original UNIX system code (Version 7 or later)
 - Compatible systems that are not derived from the original UNIX system code
 - Emulations hosted on entirely different operating systems
 - Networked systems
 - Distributed systems
 - Systems running on a broad range of hardware

No direct references to this goal appear in POSIX.1-2008, but some results of it are mentioned in the Rationale (Informative) volume.

- Minimal Changes to Historical Implementations – When the original version – IEEE Std 1003.1-1988 – was published, there were no known historical implementations that did not have to change. However, there was a broad consensus on a set of functions, types, definitions, and concepts that formed an interface that was common to most historical implementations.

The adoption of the 1988 and 1990 IEEE system interface standards, the 1992 IEEE shell and utilities standard, the various Open Group (formerly X/Open) specifications, and IEEE Std 1003.1-2001 and its technical corrigenda have consolidated this consensus, and this version reflects the significantly increased level of consensus arrived at since the original versions. The authors of the original versions tried, as much as possible, to follow the principles below when creating new specifications:

- By standardizing an interface like one in an historical implementation; for example, directories
- By specifying an interface that is readily implementable in terms of, and backwards-compatible with, historical implementations, such as the extended *tar* format defined in the *pax* utility
- By specifying an interface that, when added to an historical implementation, will not conflict with it; for example, the *sigaction()* function

POSIX.1-2008 is specifically not a codification of a particular vendor's product.

It should be noted that implementations will have different kinds of extensions. Some will reflect “historical usage” and will be preserved for execution of pre-existing applications. These functions should be considered “obsolescent” and the standard functions used for new applications. Some extensions will represent functions beyond the scope of POSIX.1-2008. These need to be used with careful management to be able to adapt to future extensions of POSIX.1-2008 and/or port to implementations that provide these services in a different manner.

- Minimal Changes to Existing Application Code – A goal of POSIX.1-2008 was to minimize additional work for application developers. However, because every known historical implementation will have to change at least slightly to conform, some applications will have to change.

POSIX.1-2008

POSIX.1-2008 defines the Portable Operating System Interface (POSIX) requirements and consists of the following topics arranged as a series of volumes within the standard:

- Base Definitions
- System Interfaces
- Shell and Utilities
- Rationale (Informative)

Base Definitions

The Base Definitions volume provides common definitions for this standard, therefore readers should be familiar with it before using the other volumes.

This volume is structured as follows:

- Chapter 1 is an introduction.
- Chapter 2 defines the conformance requirements.
- Chapter 3 defines general terms used.
- Chapter 4 describes general concepts used.
- Chapter 5 describes the notation used to specify file input and output formats in this volume and the Shell and Utilities volume.
- Chapter 6 describes the portable character set and the process of character set definition.

- Chapter 7 describes the syntax for defining internationalization locales as well as the POSIX locale provided on all systems.
- Chapter 8 describes the use of environment variables for internationalization and other purposes.
- Chapter 9 describes the syntax of pattern matching using regular expressions employed by many utilities and matched by the *regcomp()* and *regexexec()* functions.
- Chapter 10 describes files and devices found on all systems.
- Chapter 11 describes the asynchronous terminal interface for many of the functions in the System Interfaces volume and the *stty* utility in the Shell and Utilities volume.
- Chapter 12 describes the policies for command line argument construction and parsing.
- Chapter 13 defines the contents of headers which declare the functions and global variables, and define types, constants, macros, and data structures that are needed by programs using the services provided by the System Interfaces volume.

Comprehensive references are available in the index.

System Interfaces

The System Interfaces volume describes the interfaces offered to application programs by POSIX-conformant systems. Readers are expected to be experienced C language programmers, and to be familiar with the Base Definitions volume.

This volume is structured as follows:

- Chapter 1 explains the status of this volume and its relationship to other formal standards.
- Chapter 2 contains important concepts, terms, and caveats relating to the rest of this volume.
- Chapter 3 defines the functional interfaces to the POSIX-conformant system.

Comprehensive references are available in the index.

Shell and Utilities

The Shell and Utilities volume describes the commands and utilities offered to application programs on POSIX-conformant systems. Readers are expected to be familiar with the Base Definitions volume.

This volume is structured as follows:

- Chapter 1 explains the status of this volume and its relationship to other formal standards. It also describes the defaults used by the utility descriptions.
- Chapter 2 describes the command language used in POSIX-conformant systems, and special built-in utilities.
- Chapter 3 describes a set of services and utilities that are implemented on systems supporting the Batch Environment Services and Utilities option.
- Chapter 4 consists of reference pages for all utilities, other than the special built-in utilities described in Chapter 2, available on POSIX-conformant systems.

Comprehensive references are available in the index.

Rationale (Informative)

The Rationale volume is published to assist in the process of review. It contains historical information concerning the contents of this standard and why features were included or discarded by the standard developers. It also contains notes of interest to application programmers on recommended programming practices, emphasizing the consequences of some aspects of POSIX.1-2008 that may not be immediately apparent.

This volume is organized in parallel to the normative volumes of this standard, with a separate part for each of the three normative volumes.

Within this volume, the following terms are used:

- Base standard – The portions of POSIX.1-2008 that are not optional, equivalent to the definitions of classic POSIX.1 and POSIX.2.
- POSIX.0 – Although this term is not used in the normative text of POSIX.1-2008, it is used in this volume to refer to IEEE Std 1003.0™-1995.
- POSIX.1b – Although this term is not used in the normative text of POSIX.1-2008, it is used in this volume to refer to the elements of the POSIX Realtime Extension amendment. (This was earlier referred to as POSIX.4 during the standard development process.)
- POSIX.1c – Although this term is not used in the normative text of POSIX.1-2008, it is used in this volume to refer to the POSIX Threads Extension amendment. (This was earlier referred to as POSIX.4a during the standard development process.)
- Standard developers – The individuals and companies in the development organizations responsible for POSIX.1-2008: the IEEE P1003.1 working groups, The Open Group Base working group, advised by the hundreds of individual technical experts who balloted the draft standards within the Austin Group, and the member bodies and technical experts of ISO/IEC JTC 1/SC 22.
- XSI option – The portions of POSIX.1-2008 addressing the extension added for support of the Single UNIX Specification.

Typographical Conventions

The following typographical conventions are used throughout this standard. In the text, this standard is referred to as POSIX.1-2008, which is technically identical to The Open Group Base Specifications, Issue 7.

The typographical conventions listed here are for ease of reading only. Editorial inconsistencies in the use of typography are unintentional and have no normative meaning in POSIX.1-2008.

Reference	Example	Notes
C-Language Data Structure	aiocb	
C-Language Data Structure Member	<i>aio_lio_opcode</i>	
C-Language Data Type	long	
C-Language External Variable	<i>errno</i>	
C-Language Function	<i>system()</i>	
C-Language Function Argument	<i>arg</i>	
C-Language Function Family	<i>exec</i>	
C-Language Header	<sys/stat.h>	
C-Language Keyword	return	
C-Language Macro with Argument	<i>assert()</i>	
C-Language Macro with No Argument	NET_ADDRSTRLEN	
C-Language Preprocessing Directive	#define	
Commands within a Utility	a, c	
Conversion Specifier, Specifier/Modifier Character	%A, g, E	1
Environment Variable	<i>PATH</i>	
Error Number	[EINTR]	
Example Output	Hello, World	
Filename	/tmp	
Literal Character	'c', '\r'	2
Literal String	"abcde"	2
Optional Items in Utility Syntax	[]	
Parameter	<i><directory pathname></i>	

Reference	Example	Notes
Special Character	<newline>	3
Symbolic Constant	POSIX_VDISABLE	
Symbolic Limit, Configuration Value	{LINE_MAX}	4
Syntax	#include <sys/stat.h>	
User Input and Example Code	echo Hello, World	5
Utility Name	awk	
Utility Operand	file_name	
Utility Option	-c	
Utility Option with Option-Argument	-w width	

Note that:

1. Conversion specifications, specifier characters, and modifier characters are used primarily in date-related functions and utilities and the *fprintf()* and *fscanf()* formatting functions.
2. Unless otherwise noted, the quotes shall not be used as input or output. When used in a list item, the quotes are omitted. The literal characters <apostrophe> (also known as single-quote) and <backslash> are either shown as the C constants ' \' and ' \ ' , respectively, or as the special characters <apostrophe>, single-quote, and <backslash> depending on context.
3. The style selected for some of the special characters, such as <newline>, matches the form of the input given to the *localedef* utility. Generally, the characters selected for this special treatment are those that are not visually distinct, such as the control characters <tab> or <newline>.
4. Names surrounded by braces represent symbolic limits or configuration values which may be declared in appropriate headers by means of the C **#define** construct.
5. Brackets shown in this font, " [] ", are part of the syntax and do not indicate optional items. In syntax the ' | ' symbol is used to separate alternatives, and ellipses (" . . . ") are used to show that additional arguments are optional.

Shading is used to identify extensions and options.

Footnotes and notes within the body of the normative text are for information only (informative).

Informative sections (such as Rationale, Change History, Application Usage, and so on) are denoted by continuous shading bars in the margins.

Ranges of values are indicated with parentheses or brackets as follows:

1. (a,b) means the range of all values from a to b , including neither a nor b .
2. $[a,b]$ means the range of all values from a to b , including a and b .
3. $[a,b)$ means the range of all values from a to b , including a , but not b .
4. $(a,b]$ means the range of all values from a to b , including b , but not a .

Note: A symbolic limit beginning with POSIX is treated differently, depending on context. In a C-language header, the symbol *POSIXstring* (where *string* may contain underscores) is represented by the C identifier `_POSIXstring`, with a leading underscore required to prevent ISO C standard name space pollution. However, in other contexts, such as languages other than C, the leading underscore is not used because this requirement does not exist.

Contents

Volume	1	Base Definitions, Issue 7.....	1
Chapter	1	Introduction.....	3
	1.1	Scope	3
	1.2	Conformance.....	4
	1.3	Normative References	4
	1.4	Change History	5
	1.5	Terminology	5
	1.6	Definitions and Concepts.....	6
	1.7	Portability	6
	1.7.1	Codes	7
	1.7.2	Margin Code Notation	13
Chapter	2	Conformance.....	15
	2.1	Implementation Conformance	15
	2.1.1	Requirements	15
	2.1.2	Documentation	16
	2.1.3	POSIX Conformance.....	16
	2.1.4	XSI Conformance	19
	2.1.5	Option Groups.....	20
	2.1.6	Options	26
	2.2	Application Conformance.....	29
	2.2.1	Strictly Conforming POSIX Application.....	29
	2.2.2	Conforming POSIX Application	30
	2.2.3	Conforming POSIX Application Using Extensions.....	30
	2.2.4	Strictly Conforming XSI Application	30
	2.2.5	Conforming XSI Application Using Extensions	31
	2.3	Language-Dependent Services for the C Programming Language	31
	2.4	Other Language-Related Specifications.....	31
Chapter	3	Definitions.....	33
	3.1	Abortive Release.....	33
	3.2	Absolute Pathname.....	33
	3.3	Access Mode	33
	3.4	Additional File Access Control Mechanism.....	33
	3.5	Address Space.....	33
	3.6	Advisory Information.....	34
	3.7	Affirmative Response	34
	3.8	Alert	34
	3.9	Alert Character (<alert>).....	34
	3.10	Alias Name.....	34
	3.11	Alignment	35
	3.12	Alternate File Access Control Mechanism	35

3.13	Alternate Signal Stack.....	35
3.14	Ancillary Data.....	35
3.15	Angle Brackets.....	35
3.16	Apostrophe Character (<apostrophe>).....	35
3.17	Application.....	35
3.18	Application Address.....	36
3.19	Application Program Interface (API).....	36
3.20	Appropriate Privileges.....	36
3.21	Argument.....	36
3.22	Arm (a Timer).....	36
3.23	Asterisk Character (<asterisk>).....	36
3.24	Async-Cancel-Safe Function.....	36
3.25	Asynchronous Events.....	37
3.26	Asynchronous Input and Output.....	37
3.27	Async-Signal-Safe Function.....	37
3.28	Asynchronously-Generated Signal.....	37
3.29	Asynchronous I/O Completion.....	37
3.30	Asynchronous I/O Operation.....	37
3.31	Authentication.....	37
3.32	Authorization.....	38
3.33	Background Job.....	38
3.34	Background Process.....	38
3.35	Background Process Group (or Background Job).....	38
3.36	Backquote Character.....	38
3.37	Backslash Character (<backslash>).....	38
3.38	Backspace Character (<backspace>).....	38
3.39	Barrier.....	39
3.40	Basename.....	39
3.41	Basic Regular Expression (BRE).....	39
3.42	Batch Access List.....	39
3.43	Batch Administrator.....	39
3.44	Batch Client.....	39
3.45	Batch Destination.....	40
3.46	Batch Destination Identifier.....	40
3.47	Batch Directive.....	40
3.48	Batch Job.....	40
3.49	Batch Job Attribute.....	40
3.50	Batch Job Identifier.....	40
3.51	Batch Job Name.....	41
3.52	Batch Job Owner.....	41
3.53	Batch Job Priority.....	41
3.54	Batch Job State.....	41
3.55	Batch Name Service.....	41
3.56	Batch Name Space.....	41
3.57	Batch Node.....	42
3.58	Batch Operator.....	42
3.59	Batch Queue.....	42
3.60	Batch Queue Attribute.....	42
3.61	Batch Queue Position.....	42
3.62	Batch Queue Priority.....	42
3.63	Batch Rerunability.....	43
3.64	Batch Restart.....	43

3.65	Batch Server	43
3.66	Batch Server Name.....	43
3.67	Batch Service	43
3.68	Batch Service Request.....	43
3.69	Batch Submission	43
3.70	Batch System	44
3.71	Batch Target User	44
3.72	Batch User	44
3.73	Bind	44
3.74	Blank Character (<blank>).....	44
3.75	Blank Line.....	44
3.76	Blocked Process (or Thread)	44
3.77	Blocking	44
3.78	Block-Mode Terminal	45
3.79	Block Special File.....	45
3.80	Braces	45
3.81	Brackets.....	45
3.82	Broadcast	45
3.83	Built-In Utility (or Built-In).....	46
3.84	Byte.....	46
3.85	Byte Input/Output Functions	46
3.86	Carriage-Return Character (<carriage-return>)	46
3.87	Character	47
3.88	Character Array	47
3.89	Character Class.....	47
3.90	Character Set	47
3.91	Character Special File	47
3.92	Character String.....	47
3.93	Child Process	48
3.94	Circumflex Character (<circumflex>).....	48
3.95	Clock	48
3.96	Clock Jump.....	48
3.97	Clock Tick.....	48
3.98	Coded Character Set	48
3.99	Codeset	49
3.100	Collating Element.....	49
3.101	Collation	49
3.102	Collation Sequence.....	49
3.103	Column Position.....	50
3.104	Command.....	50
3.105	Command Language Interpreter	50
3.106	Composite Graphic Symbol.....	50
3.107	Condition Variable	50
3.108	Connected Socket	51
3.109	Connection	51
3.110	Connection Mode.....	51
3.111	Connectionless Mode	51
3.112	Control Character.....	51
3.113	Control Operator	51
3.114	Controlling Process.....	51
3.115	Controlling Terminal	52
3.116	Conversion Descriptor	52

3.117	Core File.....	52
3.118	CPU Time (Execution Time)	52
3.119	CPU-Time Clock.....	52
3.120	CPU-Time Timer.....	52
3.121	Current Job.....	52
3.122	Current Working Directory.....	53
3.123	Cursor Position.....	53
3.124	Datagram.....	53
3.125	Data Segment.....	53
3.126	Deferred Batch Service	53
3.127	Device	53
3.128	Device ID	53
3.129	Directory	53
3.130	Directory Entry (or Link)	53
3.131	Directory Stream	54
3.132	Disarm (a Timer)	54
3.133	Display	54
3.134	Display Line.....	54
3.135	Dollar-Sign Character (<dollar-sign>)	54
3.136	Dot.....	54
3.137	Dot-Dot.....	55
3.138	Double-Quote Character.....	55
3.139	Downshifting.....	55
3.140	Driver	55
3.141	Effective Group ID	55
3.142	Effective User ID	55
3.143	Eight-Bit Transparency	55
3.144	Empty Directory.....	56
3.145	Empty Line.....	56
3.146	Empty String (or Null String).....	56
3.147	Empty Wide-Character String.....	56
3.148	Encoding Rule	56
3.149	Entire Regular Expression.....	56
3.150	Epoch	57
3.151	Equivalence Class.....	57
3.152	Era.....	57
3.153	Event Management.....	57
3.154	Executable File.....	57
3.155	Execute.....	58
3.156	Execution Time	58
3.157	Execution Time Monitoring.....	58
3.158	Expand.....	58
3.159	Extended Regular Expression (ERE)	58
3.160	Extended Security Controls.....	58
3.161	Feature Test Macro	59
3.162	Field.....	59
3.163	FIFO Special File (or FIFO)	59
3.164	File	59
3.165	File Description	59
3.166	File Descriptor	60
3.167	File Group Class	60
3.168	File Mode.....	60

3.169	File Mode Bits	60
3.170	Filename	60
3.171	Filename String.....	61
3.172	File Offset	61
3.173	File Other Class	61
3.174	File Owner Class	61
3.175	File Permission Bits.....	61
3.176	File Serial Number	61
3.177	File System	61
3.178	File Type	62
3.179	Filter	62
3.180	First Open (of a File)	62
3.181	Flow Control	62
3.182	Foreground Job.....	62
3.183	Foreground Process	62
3.184	Foreground Process Group (or Foreground Job).....	62
3.185	Foreground Process Group ID	62
3.186	Form-Feed Character (<form-feed>).....	63
3.187	Graphic Character.....	63
3.188	Group Database.....	63
3.189	Group ID.....	63
3.190	Group Name	63
3.191	Hard Limit.....	63
3.192	Hard Link	64
3.193	Home Directory	64
3.194	Host Byte Order.....	64
3.195	Incomplete Line.....	64
3.196	Inf.....	64
3.197	Instrumented Application	64
3.198	Interactive Shell	64
3.199	Internationalization	65
3.200	Interprocess Communication	65
3.201	Invoke	65
3.202	Job.....	65
3.203	Job Control	65
3.204	Job Control Job ID	65
3.205	Last Close (of a File).....	66
3.206	Line.....	66
3.207	Linger	66
3.208	Link	66
3.209	Link Count	66
3.210	Local Customs	66
3.211	Local Interprocess Communication (Local IPC).....	66
3.212	Locale	67
3.213	Localization.....	67
3.214	Login	67
3.215	Login Name	67
3.216	Map	67
3.217	Marked Message	67
3.218	Matched	68
3.219	Memory Mapped Files	68
3.220	Memory Object.....	68

3.221	Memory-Resident.....	68
3.222	Message	68
3.223	Message Catalog.....	68
3.224	Message Catalog Descriptor	69
3.225	Message Queue.....	69
3.226	Mode	69
3.227	Monotonic Clock	69
3.228	Mount Point	69
3.229	Multi-Character Collating Element	69
3.230	Mutex	69
3.231	Name.....	70
3.232	Named STREAM.....	70
3.233	NaN (Not a Number)	70
3.234	Native Language	70
3.235	Negative Response.....	70
3.236	Network.....	70
3.237	Network Address	70
3.238	Network Byte Order	71
3.239	Newline Character (<newline>)	71
3.240	Nice Value	71
3.241	Non-Blocking.....	71
3.242	Non-Spacing Characters	71
3.243	NUL.....	72
3.244	Null Byte.....	72
3.245	Null Pointer.....	72
3.246	Null String.....	72
3.247	Null Wide-Character Code	72
3.248	Number-Sign Character (<number-sign>)	72
3.249	Object File.....	72
3.250	Octet	72
3.251	Offset Maximum	73
3.252	Opaque Address.....	73
3.253	Open File	73
3.254	Open File Description.....	73
3.255	Operand.....	73
3.256	Operator	73
3.257	Option.....	74
3.258	Option-Argument	74
3.259	Orientation	74
3.260	Orphaned Process Group.....	74
3.261	Page.....	74
3.262	Page Size	74
3.263	Parameter	75
3.264	Parent Directory	75
3.265	Parent Process.....	75
3.266	Parent Process ID	75
3.267	Pathname.....	75
3.268	Pathname Component.....	76
3.269	Path Prefix	76
3.270	Pattern.....	76
3.271	Period Character (<period>)	76
3.272	Permissions	77

3.273	Persistence.....	77
3.274	Pipe.....	77
3.275	Polling.....	77
3.276	Portable Character Set.....	77
3.277	Portable Filename.....	78
3.278	Portable Filename Character Set.....	78
3.279	Positional Parameter.....	78
3.280	Preallocation.....	78
3.281	Preempted Process (or Thread).....	78
3.282	Previous Job.....	78
3.283	Printable Character.....	79
3.284	Printable File.....	79
3.285	Priority.....	79
3.286	Priority Band.....	79
3.287	Priority Inversion.....	79
3.288	Priority Scheduling.....	79
3.289	Priority-Based Scheduling.....	79
3.290	Privilege.....	80
3.291	Process.....	80
3.292	Process Group.....	80
3.293	Process Group ID.....	80
3.294	Process Group Leader.....	80
3.295	Process Group Lifetime.....	80
3.296	Process ID.....	81
3.297	Process Lifetime.....	81
3.298	Process Memory Locking.....	81
3.299	Process Termination.....	81
3.300	Process-To-Process Communication.....	81
3.301	Process Virtual Time.....	82
3.302	Program.....	82
3.303	Protocol.....	82
3.304	Pseudo-Terminal.....	82
3.305	Radix Character.....	82
3.306	Read-Only File System.....	82
3.307	Read-Write Lock.....	82
3.308	Real Group ID.....	83
3.309	Real Time.....	83
3.310	Realtime Signal Extension.....	83
3.311	Real User ID.....	83
3.312	Record.....	83
3.313	Redirection.....	83
3.314	Redirection Operator.....	84
3.315	Referenced Shared Memory Object.....	84
3.316	Refresh.....	84
3.317	Regular Expression.....	84
3.318	Region.....	84
3.319	Regular File.....	84
3.320	Relative Pathname.....	85
3.321	Relocatable File.....	85
3.322	Relocation.....	85
3.323	Requested Batch Service.....	85
3.324	(Time) Resolution.....	85

3.325	Robust Mutex.....	85
3.326	Root Directory	85
3.327	Runnable Process (or Thread)	85
3.328	Running Process (or Thread).....	86
3.329	Saved Resource Limits	86
3.330	Saved Set-Group-ID.....	86
3.331	Saved Set-User-ID	86
3.332	Scheduling.....	86
3.333	Scheduling Allocation Domain	86
3.334	Scheduling Contention Scope	86
3.335	Scheduling Policy.....	87
3.336	Screen	87
3.337	Scroll.....	87
3.338	Semaphore.....	87
3.339	Session	88
3.340	Session Leader	88
3.341	Session Lifetime.....	88
3.342	Shared Memory Object.....	88
3.343	Shell.....	88
3.344	Shell, the	88
3.345	Shell Script.....	89
3.346	Signal.....	89
3.347	Signal Stack	89
3.348	Single-Quote Character	89
3.349	Slash Character (<slash>).....	89
3.350	Socket	89
3.351	Socket Address	89
3.352	Soft Limit	90
3.353	Source Code	90
3.354	Space Character (<space>).....	90
3.355	Spawn	90
3.356	Special Built-In.....	90
3.357	Special Parameter.....	91
3.358	Spin Lock.....	91
3.359	Sporadic Server.....	91
3.360	Standard Error	91
3.361	Standard Input.....	91
3.362	Standard Output	91
3.363	Standard Utilities	91
3.364	Stream	92
3.365	STREAM	92
3.366	STREAM End.....	92
3.367	STREAM Head	92
3.368	STREAMS Multiplexor.....	92
3.369	String.....	92
3.370	Subshell.....	93
3.371	Successfully Transferred	93
3.372	Supplementary Group ID	93
3.373	Suspended Job	93
3.374	Symbolic Constant	93
3.375	Symbolic Link	94
3.376	Synchronized Input and Output.....	94

3.377	Synchronized I/O Completion	94
3.378	Synchronized I/O Data Integrity Completion.....	94
3.379	Synchronized I/O File Integrity Completion	94
3.380	Synchronized I/O Operation	94
3.381	Synchronous I/O Operation.....	95
3.382	Synchronously-Generated Signal	95
3.383	System.....	95
3.384	System Boot.....	95
3.385	System Clock.....	95
3.386	System Console	95
3.387	System Crash	95
3.388	System Databases.....	96
3.389	System Documentation	96
3.390	System Process.....	96
3.391	System Reboot	96
3.392	System Trace Event	96
3.393	System-Wide	96
3.394	Tab Character (<tab>).....	97
3.395	Terminal (or Terminal Device)	97
3.396	Text Column.....	97
3.397	Text File.....	97
3.398	Thread.....	97
3.399	Thread ID	97
3.400	Thread List	98
3.401	Thread-Safe	98
3.402	Thread-Specific Data Key.....	98
3.403	Tilde Character (<tilde>).....	98
3.404	Timeouts	98
3.405	Timer	98
3.406	Timer Overrun	98
3.407	Token.....	99
3.408	Trace Analyzer Process.....	99
3.409	Trace Controller Process.....	99
3.410	Trace Event.....	99
3.411	Trace Event Type	99
3.412	Trace Event Type Mapping	99
3.413	Trace Filter.....	99
3.414	Trace Generation Version	99
3.415	Trace Log	100
3.416	Trace Point.....	100
3.417	Trace Stream.....	100
3.418	Trace Stream Identifier	100
3.419	Trace System	100
3.420	Traced Process.....	100
3.421	Tracing Status of a Trace Stream	100
3.422	Typed Memory Name Space	100
3.423	Typed Memory Object.....	101
3.424	Typed Memory Pool	101
3.425	Typed Memory Port.....	101
3.426	Unbind	101
3.427	Unit Data	101
3.428	Upshifting	101

3.429	User Database	101
3.430	User ID.....	102
3.431	User Name	102
3.432	User Trace Event.....	102
3.433	Utility	102
3.434	Variable	103
3.435	Vertical-Tab Character (<vertical-tab>).....	103
3.436	White Space.....	103
3.437	Wide-Character Code (C Language)	103
3.438	Wide-Character Input/Output Functions	103
3.439	Wide-Character String.....	103
3.440	Word.....	104
3.441	Working Directory (or Current Working Directory)	104
3.442	Worldwide Portability Interface	104
3.443	Write	104
3.444	XSI	104
3.445	XSI-Conformant	105
3.446	Zombie Process.....	105
3.447	±0	105
Chapter 4	General Concepts	107
4.1	Concurrent Execution.....	107
4.2	Directory Protection.....	107
4.3	Extended Security Controls	107
4.4	File Access Permissions.....	108
4.5	File Hierarchy	108
4.6	Filenames.....	109
4.7	Filename Portability.....	109
4.8	File Times Update	109
4.9	Host and Network Byte Orders	110
4.10	Measurement of Execution Time	110
4.11	Memory Synchronization	110
4.12	Pathname Resolution.....	111
4.13	Process ID Reuse	112
4.14	Scheduling Policy.....	112
4.15	Seconds Since the Epoch	113
4.16	Semaphore.....	113
4.17	Thread-Safety	114
4.18	Tracing	114
4.19	Treatment of Error Conditions for Mathematical Functions	116
4.19.1	Domain Error	116
4.19.2	Pole Error.....	117
4.19.3	Range Error	117
4.20	Treatment of NaN Arguments for the Mathematical Functions	118
4.21	Utility	118
4.22	Variable Assignment.....	118
Chapter 5	File Format Notation.....	121
Chapter 6	Character Set	125

	6.1	Portable Character Set	125
	6.2	Character Encoding	128
	6.3	C Language Wide-Character Codes	129
	6.4	Character Set Description File	129
	6.4.1	State-Dependent Character Encodings	132
Chapter	7	Locale	135
	7.1	General.....	135
	7.2	POSIX Locale	136
	7.3	Locale Definition	136
	7.3.1	LC_CTYPE	139
	7.3.2	LC_COLLATE.....	147
	7.3.3	LC_MONETARY	154
	7.3.4	LC_NUMERIC.....	157
	7.3.5	LC_TIME	159
	7.3.6	LC_MESSAGES	164
	7.4	Locale Definition Grammar	165
	7.4.1	Locale Lexical Conventions	165
	7.4.2	Locale Grammar.....	166
Chapter	8	Environment Variables	173
	8.1	Environment Variable Definition.....	173
	8.2	Internationalization Variables	174
	8.3	Other Environment Variables.....	177
Chapter	9	Regular Expressions.....	181
	9.1	Regular Expression Definitions.....	181
	9.2	Regular Expression General Requirements.....	182
	9.3	Basic Regular Expressions	183
	9.3.1	BREs Matching a Single Character or Collating Element	183
	9.3.2	BRE Ordinary Characters.....	183
	9.3.3	BRE Special Characters	183
	9.3.4	Periods in BREs	184
	9.3.5	RE Bracket Expression.....	184
	9.3.6	BREs Matching Multiple Characters	186
	9.3.7	BRE Precedence	187
	9.3.8	BRE Expression Anchoring.....	187
	9.4	Extended Regular Expressions.....	188
	9.4.1	EREs Matching a Single Character or Collating Element	188
	9.4.2	ERE Ordinary Characters.....	188
	9.4.3	ERE Special Characters	188
	9.4.4	Periods in EREs	189
	9.4.5	ERE Bracket Expression	189
	9.4.6	EREs Matching Multiple Characters	189
	9.4.7	ERE Alternation.....	190
	9.4.8	ERE Precedence	190
	9.4.9	ERE Expression Anchoring.....	191
	9.5	Regular Expression Grammar	191
	9.5.1	BRE/ERE Grammar Lexical Conventions.....	191
	9.5.2	RE and Bracket Expression Grammar.....	192

	9.5.3	ERE Grammar	195
Chapter	10	Directory Structure and Devices	197
	10.1	Directory Structure and Files.....	197
	10.2	Output Devices and Terminal Types.....	198
Chapter	11	General Terminal Interface	199
	11.1	Interface Characteristics	199
	11.1.1	Opening a Terminal Device File.....	199
	11.1.2	Process Groups	200
	11.1.3	The Controlling Terminal.....	200
	11.1.4	Terminal Access Control	201
	11.1.5	Input Processing and Reading Data	201
	11.1.6	Canonical Mode Input Processing.....	202
	11.1.7	Non-Canonical Mode Input Processing.....	202
	11.1.8	Writing Data and Output Processing	203
	11.1.9	Special Characters	204
	11.1.10	Modem Disconnect	205
	11.1.11	Closing a Terminal Device File.....	205
	11.2	Parameters that Can be Set	205
	11.2.1	The termios Structure	205
	11.2.2	Input Modes.....	206
	11.2.3	Output Modes.....	207
	11.2.4	Control Modes	209
	11.2.5	Local Modes.....	210
	11.2.6	Special Control Characters.....	212
Chapter	12	Utility Conventions	213
	12.1	Utility Argument Syntax.....	213
	12.2	Utility Syntax Guidelines.....	215
Chapter	13	Headers	219
Volume	2	System Interfaces, Issue 7.....	467
Chapter	1	Introduction.....	469
	1.1	Relationship to Other Formal Standards.....	469
	1.2	Format of Entries.....	469
Chapter	2	General Information	471
	2.1	Use and Implementation of Interfaces	471
	2.1.1	Use and Implementation of Functions.....	471
	2.1.2	Use and Implementation of Macros	472
	2.2	The Compilation Environment	472
	2.2.1	POSIX.1 Symbols.....	472
	2.2.2	The Name Space.....	473
	2.3	Error Numbers.....	481
	2.3.1	Additional Error Numbers	488
	2.4	Signal Concepts	488
	2.4.1	Signal Generation and Delivery.....	488
	2.4.2	Realtime Signal Generation and Delivery	489

2.4.3	Signal Actions	490
2.4.4	Signal Effects on Other Functions.....	495
2.5	Standard I/O Streams	495
2.5.1	Interaction of File Descriptors and Standard I/O Streams	496
2.5.2	Stream Orientation and Encoding Rules	498
2.6	STREAMS.....	499
2.6.1	Accessing STREAMS	500
2.7	XSI Interprocess Communication	501
2.7.1	IPC General Description	501
2.8	Realtime.....	502
2.8.1	Realtime Signals	502
2.8.2	Asynchronous I/O.....	503
2.8.3	Memory Management.....	504
2.8.4	Process Scheduling.....	506
2.8.5	Clocks and Timers.....	510
2.9	Threads	512
2.9.1	Thread-Safety.....	512
2.9.2	Thread IDs.....	513
2.9.3	Thread Mutexes.....	513
2.9.4	Thread Scheduling	514
2.9.5	Thread Cancellation.....	516
2.9.6	Thread Read-Write Locks.....	520
2.9.7	Thread Interactions with Regular File Operations.....	521
2.9.8	Use of Application-Managed Thread Stacks.....	521
2.10	Sockets	522
2.10.1	Address Families	522
2.10.2	Addressing	522
2.10.3	Protocols	522
2.10.4	Routing	523
2.10.5	Interfaces	523
2.10.6	Socket Types.....	523
2.10.7	Socket I/O Mode.....	524
2.10.8	Socket Owner.....	524
2.10.9	Socket Queue Limits.....	524
2.10.10	Pending Error	524
2.10.11	Socket Receive Queue.....	525
2.10.12	Socket Out-of-Band Data State.....	525
2.10.13	Connection Indication Queue	526
2.10.14	Signals.....	526
2.10.15	Asynchronous Errors.....	526
2.10.16	Use of Options	527
2.10.17	Use of Sockets for Local UNIX Connections	530
2.10.18	Use of Sockets over Internet Protocols.....	530
2.10.19	Use of Sockets over Internet Protocols Based on IPv4.....	531
2.10.20	Use of Sockets over Internet Protocols Based on IPv6.....	531
2.11	Tracing	534
2.11.1	Tracing Data Definitions	536
2.11.2	Trace Event Type Definitions.....	540
2.11.3	Trace Functions.....	544

	2.12	Data Types.....	545
	2.12.1	Defined Types	545
	2.12.2	The char Type.....	546
Chapter	3	System Interfaces.....	547
Volume	3	Shell and Utilities, Issue 7	2299
Chapter	1	Introduction.....	2301
	1.1	Relationship to Other Documents	2301
	1.1.1	System Interfaces.....	2301
	1.1.2	Concepts Derived from the ISO C Standard	2305
	1.2	Utility Limits.....	2307
	1.3	Grammar Conventions.....	2309
	1.4	Utility Description Defaults.....	2310
	1.5	Considerations for Utilities in Support of Files of Arbitrary Size.....	2317
	1.6	Built-In Utilities	2318
Chapter	2	Shell Command Language	2319
	2.1	Shell Introduction.....	2319
	2.2	Quoting.....	2320
	2.2.1	Escape Character (Backslash)	2320
	2.2.2	Single-Quotes.....	2320
	2.2.3	Double-Quotes.....	2320
	2.3	Token Recognition.....	2321
	2.3.1	Alias Substitution.....	2322
	2.4	Reserved Words.....	2323
	2.5	Parameters and Variables.....	2323
	2.5.1	Positional Parameters	2323
	2.5.2	Special Parameters	2324
	2.5.3	Shell Variables.....	2324
	2.6	Word Expansions	2327
	2.6.1	Tilde Expansion	2327
	2.6.2	Parameter Expansion.....	2328
	2.6.3	Command Substitution	2331
	2.6.4	Arithmetic Expansion.....	2332
	2.6.5	Field Splitting	2333
	2.6.6	Pathname Expansion	2333
	2.6.7	Quote Removal.....	2333
	2.7	Redirection	2334
	2.7.1	Redirecting Input	2334
	2.7.2	Redirecting Output	2335
	2.7.3	Appending Redirected Output	2335
	2.7.4	Here-Document.....	2335
	2.7.5	Duplicating an Input File Descriptor	2336
	2.7.6	Duplicating an Output File Descriptor	2336
	2.7.7	Open File Descriptors for Reading and Writing.....	2337
	2.8	Exit Status and Errors	2337
	2.8.1	Consequences of Shell Errors	2337
	2.8.2	Exit Status for Commands	2337

	2.9	Shell Commands	2338
	2.9.1	Simple Commands.....	2338
	2.9.2	Pipelines	2341
	2.9.3	Lists	2341
	2.9.4	Compound Commands.....	2343
	2.9.5	Function Definition Command	2346
	2.10	Shell Grammar.....	2347
	2.10.1	Shell Grammar Lexical Conventions.....	2347
	2.10.2	Shell Grammar Rules.....	2348
	2.11	Signals and Error Handling.....	2353
	2.12	Shell Execution Environment.....	2353
	2.13	Pattern Matching Notation	2354
	2.13.1	Patterns Matching a Single Character	2354
	2.13.2	Patterns Matching Multiple Characters.....	2355
	2.13.3	Patterns Used for Filename Expansion.....	2355
	2.14	Special Built-In Utilities.....	2356
Chapter	3	Batch Environment Services	2397
	3.1	General Concepts	2397
	3.1.1	Batch Client-Server Interaction	2397
	3.1.2	Batch Queues	2398
	3.1.3	Batch Job Creation.....	2398
	3.1.4	Batch Job Tracking.....	2398
	3.1.5	Batch Job Routing.....	2399
	3.1.6	Batch Job Execution	2399
	3.1.7	Batch Job Exit.....	2400
	3.1.8	Batch Job Abort.....	2400
	3.1.9	Batch Authorization.....	2400
	3.1.10	Batch Administration	2400
	3.1.11	Batch Notification.....	2401
	3.2	Batch Services	2401
	3.2.1	Batch Job States.....	2402
	3.2.2	Deferred Batch Services.....	2403
	3.2.3	Requested Batch Services.....	2412
	3.3	Common Behavior for Batch Environment Utilities.....	2419
	3.3.1	Batch Job Identifier.....	2419
	3.3.2	Destination	2420
	3.3.3	Multiple Keyword-Value Pairs.....	2421
Chapter	4	Utilities.....	2423
Volume	4	Rationale (Informative), Issue 7.....	3435
Part	A	Base Definitions	3437
Appendix	A	Rationale for Base Definitions.....	3439
	A.1	Introduction	3439
	A.1.1	Scope	3439
	A.1.2	Conformance.....	3442
	A.1.3	Normative References	3442
	A.1.4	Change History	3442

A.1.5	Terminology	3442
A.1.6	Definitions and Concepts	3444
A.1.7	Portability	3444
A.2	Conformance	3445
A.2.1	Implementation Conformance	3445
A.2.2	Application Conformance	3449
A.2.3	Language-Dependent Services for the C Programming Language	3449
A.2.4	Other Language-Related Specifications	3450
A.3	Definitions	3450
A.4	General Concepts	3472
A.4.1	Concurrent Execution	3472
A.4.2	Directory Protection	3473
A.4.3	Extended Security Controls	3473
A.4.4	File Access Permissions	3473
A.4.5	File Hierarchy	3473
A.4.6	Filenames	3474
A.4.7	Filename Portability	3475
A.4.8	File Times Update	3475
A.4.9	Host and Network Byte Order	3476
A.4.10	Measurement of Execution Time	3476
A.4.11	Memory Synchronization	3476
A.4.12	Pathname Resolution	3478
A.4.13	Process ID Reuse	3479
A.4.14	Scheduling Policy	3479
A.4.15	Seconds Since the Epoch	3479
A.4.16	Semaphore	3481
A.4.17	Thread-Safety	3481
A.4.18	Tracing	3481
A.4.19	Treatment of Error Conditions for Mathematical Functions	3481
A.4.20	Treatment of NaN Arguments for Mathematical Functions	3481
A.4.21	Utility	3481
A.4.22	Variable Assignment	3481
A.5	File Format Notation	3482
A.6	Character Set	3482
A.6.1	Portable Character Set	3482
A.6.2	Character Encoding	3483
A.6.3	C Language Wide-Character Codes	3483
A.6.4	Character Set Description File	3483
A.7	Locale	3486
A.7.1	General	3486
A.7.2	POSIX Locale	3486
A.7.3	Locale Definition	3486
A.7.4	Locale Definition Grammar	3493
A.7.5	Locale Definition Example	3494
A.8	Environment Variables	3497
A.8.1	Environment Variable Definition	3497
A.8.2	Internationalization Variables	3497
A.8.3	Other Environment Variables	3498
A.9	Regular Expressions	3500

	A.9.1	Regular Expression Definitions.....	3500
	A.9.2	Regular Expression General Requirements.....	3501
	A.9.3	Basic Regular Expressions	3502
	A.9.4	Extended Regular Expressions.....	3505
	A.9.5	Regular Expression Grammar	3506
	A.10	Directory Structure and Devices.....	3507
	A.10.1	Directory Structure and Files.....	3507
	A.10.2	Output Devices and Terminal Types.....	3508
	A.11	General Terminal Interface	3508
	A.11.1	Interface Characteristics	3509
	A.11.2	Parameters that Can be Set	3513
	A.12	Utility Conventions.....	3514
	A.12.1	Utility Argument Syntax.....	3514
	A.12.2	Utility Syntax Guidelines	3515
	A.13	Headers.....	3518
	A.13.1	Format of Entries.....	3518
	A.13.2	Removed Headers in Issue 7	3518
Part	B	System Interfaces.....	3519
Appendix	B	Rationale for System Interfaces.....	3521
	B.1	Introduction	3521
	B.1.1	Change History	3521
	B.1.2	Relationship to Other Formal Standards.....	3524
	B.1.3	Format of Entries.....	3524
	B.2	General Information	3525
	B.2.1	Use and Implementation of Interfaces.....	3525
	B.2.2	The Compilation Environment	3526
	B.2.3	Error Numbers.....	3531
	B.2.4	Signal Concepts	3535
	B.2.5	Standard I/O Streams	3545
	B.2.6	STREAMS.....	3545
	B.2.7	XSI Interprocess Communication	3546
	B.2.8	Realtime	3547
	B.2.9	Threads	3592
	B.2.10	Sockets	3620
	B.2.11	Tracing	3622
	B.2.12	Data Types.....	3647
	B.3	System Interfaces.....	3649
	B.3.1	System Interfaces Removed in this Version	3649
	B.3.2	System Interfaces Removed in the Previous Version.....	3652
	B.3.3	Examples for Spawn	3652
Part	C	Shell and Utilities	3663
Appendix	C	Rationale for Shell and Utilities.....	3665
	C.1	Introduction	3665
	C.1.1	Change History	3665
	C.1.2	Relationship to Other Documents	3666
	C.1.3	Utility Limits.....	3667
	C.1.4	Grammar Conventions.....	3670

C.1.5	Utility Description Defaults.....	3670
C.1.6	Considerations for Utilities in Support of Files of Arbitrary Size	3674
C.1.7	Built-In Utilities	3674
C.2	Shell Command Language	3676
C.2.1	Shell Introduction.....	3676
C.2.2	Quoting.....	3676
C.2.3	Token Recognition.....	3678
C.2.4	Reserved Words.....	3679
C.2.5	Parameters and Variables.....	3679
C.2.6	Word Expansions	3682
C.2.7	Redirection	3689
C.2.8	Exit Status and Errors	3691
C.2.9	Shell Commands	3692
C.2.10	Shell Grammar.....	3699
C.2.11	Signals and Error Handling.....	3700
C.2.12	Shell Execution Environment.....	3700
C.2.13	Pattern Matching Notation	3700
C.2.14	Special Built-In Utilities.....	3702
C.3	Batch Environment Services and Utilities	3702
C.3.1	Batch General Concepts	3705
C.3.2	Batch Services	3707
C.3.3	Common Behavior for Batch Environment Utilities.....	3708
C.4	Utilities.....	3708
C.4.1	Utilities Removed in this Version	3708
C.4.2	Utilities Removed in the Previous Version.....	3708
C.4.3	Exclusion of Utilities.....	3709
Part	D Portability Considerations.....	3713
Appendix	D Portability Considerations (Informative).....	3715
D.1	User Requirements.....	3715
D.1.1	Configuration Interrogation	3716
D.1.2	Process Management	3716
D.1.3	Access to Data.....	3716
D.1.4	Access to the Environment	3716
D.1.5	Access to Determinism and Performance Enhancements.....	3716
D.1.6	Operating System-Dependent Profile	3717
D.1.7	I/O Interaction	3717
D.1.8	Internationalization Interaction	3717
D.1.9	C-Language Extensions.....	3717
D.1.10	Command Language	3717
D.1.11	Interactive Facilities	3717
D.1.12	Accomplish Multiple Tasks Simultaneously	3717
D.1.13	Complex Data Manipulation.....	3718
D.1.14	File Hierarchy Manipulation	3718
D.1.15	Locale Configuration	3718
D.1.16	Inter-User Communication.....	3718
D.1.17	System Environment	3718
D.1.18	Printing	3718

D.1.19	Software Development.....	3718
D.2	Portability Capabilities.....	3719
D.2.1	Configuration Interrogation	3719
D.2.2	Process Management	3720
D.2.3	Access to Data.....	3720
D.2.4	Access to the Environment	3721
D.2.5	Bounded (Realtime) Response	3722
D.2.6	Operating System-Dependent Profile	3722
D.2.7	I/O Interaction	3722
D.2.8	Internationalization Interaction	3723
D.2.9	C-Language Extensions.....	3723
D.2.10	Command Language	3723
D.2.11	Interactive Facilities	3724
D.2.12	Accomplish Multiple Tasks Simultaneously	3724
D.2.13	Complex Data Manipulation.....	3724
D.2.14	File Hierarchy Manipulation	3725
D.2.15	Locale Configuration	3725
D.2.16	Inter-User Communication.....	3725
D.2.17	System Environment	3726
D.2.18	Printing	3726
D.2.19	Software Development.....	3726
D.2.20	Future Growth	3726
D.3	Profiling Considerations	3727
D.3.1	Configuration Options	3727
D.3.2	Configuration Options (Shell and Utilities)	3727
D.3.3	Configurable Limits	3729
D.3.4	Configuration Options (System Interfaces).....	3729
D.3.5	Configurable Limits	3734
D.3.6	Optional Behavior	3737
Part E	Subprofiling Considerations.....	3739
Appendix E	Subprofiling Considerations (Informative)	3741
E.1	Subprofiling Option Groups.....	3741
	Index	3747
List of Figures		
B-1	Example of a System with Typed Memory	3565
B-2	Trace System Overview: for Offline Analysis	3627
B-3	Trace System Overview: for Online Analysis	3629
B-4	Trace System Overview: States of a Trace Stream	3630
B-5	Trace Another Process	3640
B-6	Trace Name Space Overview: With Third-Party Library	3641
List of Tables		
3-1	Job Control Job ID Formats.....	66
5-1	Escape Sequences and Associated Actions	121
6-1	Portable Character Set	125

6-2	Control Character Set	130
7-1	Valid Character Class Combinations.....	142
10-1	Control Character Names	198
2-1	Value of Level for Socket Options.....	527
2-2	Socket-Level Options.....	528
2-3	Trace Option: System Trace Events.....	542
2-4	Trace and Trace Event Filter Options: System Trace Events.....	542
2-5	Trace and Trace Log Options: System Trace Events.....	543
2-6	Trace, Trace Log, and Trace Event Filter Options: System Trace Events	543
2-7	Trace Option: User Trace Event.....	544
1-1	Actions when Creating a File that Already Exists.....	2303
1-2	Selected ISO C Standard Operators and Control Flow Keywords	2306
1-3	Utility Limit Minimum Values.....	2307
1-4	Symbolic Utility Limits	2308
1-5	Regular Built-In Utilities	2318
3-1	Batch Utilities.....	2397
3-2	Environment Variable Summary	2401
3-3	Next State Table.....	2403
3-4	Results/Output Table	2405
3-5	Batch Services Summary	2412
A-1	Historical Practice for Symbolic Links.....	3469

Trademarks

The following information is given for the convenience of users of POSIX.1-2008 and does not constitute an endorsement by the IEEE or The Open Group of these products. Equivalent products may be used if they can be shown to lead to the same results.

There may be other products mentioned in the text that might be covered by trademark protection and readers are advised to verify them independently.

AIX[®] and IBM[®] are registered trademarks of International Business Machines Corporation.

ArchiMate[®], Jericho Forum[®], Making Standards Work[®], The Open Group[®], TOGAF[®], UNIX[®], and the “X”[®] device are registered trademarks and Boundaryless Information Flow[™], DirecNet[™], FACE[™], and The Open Group Certification Mark[™] are trademarks of The Open Group.

AT&T[®] is a registered trademark of AT&T in the USA and other countries.

BSD[™] is a trademark of the University of California, Berkeley, USA.

Hewlett-Packard[®], HP[®], and HP-UX[®] are registered trademarks of Hewlett-Packard Company.

IEEE[®] and POSIX[®] are registered trademarks, and 754[™], 854[™], 1003.0[™], 1003.1[™], 1003.1d[™], 1003.1g[™], 1003.1j[™], 1003.1q[™], 1003.2[™], 1003.2a[™], 1003.2d[™], 1003.9[™], and 1003.13[™] are trademarks of The Institute of Electrical and Electronic Engineers, Inc.

Linux[®] is a registered trademark of Linus Torvalds.

Sun[®] and Sun Microsystems[®] are registered trademarks of Oracle America, Inc.

/usr/group[®] is a registered trademark of UniForum, the International Network of UNIX System Users.

Acknowledgements

The contributions of the following organizations to the development of POSIX.1-2008 are gratefully acknowledged:

- AT&T for permission to reproduce portions of its copyrighted System V Interface Definition (SVID) and material from the UNIX System V Release 2.0 documentation.
- Hewlett-Packard Company, International Business Machines Corporation, Novell Inc., The Open Software Foundation, and Sun Microsystems Inc. for permission to reproduce portions of their copyrighted documentation
- ISO/IEC JTC 1/SC 22/WG 14 C Language Committee
- Red Hat Inc. for permission to reproduce portions of its copyrighted documentation

POSIX.1-2008 was prepared by the Austin Group, a joint working group of the IEEE, The Open Group, and ISO/IEC JTC 1/SC 22.

Referenced Documents

Normative References

Normative references for POSIX.1-2008 are defined in [Section 1.3](#) (on page 4).

Informative References

The following documents are referenced in POSIX.1-2008:

1984 /usr/group Standard

/usr/group Standards Committee, Santa Clara, CA, UniForum 1984.

Almasi and Gottlieb

George S. Almasi and Allan Gottlieb, *Highly Parallel Computing*, The Benjamin/Cummings Publishing Company, Inc., 1989, ISBN: 0-8053-0177-1.

ANSI C

American National Standard for Information Systems: Standard X3.159-1989, Programming Language C.

ANSI X3.226-1994

American National Standard for Information Systems: Standard X3.226-1994, Programming Language Common LISP.

Brawer

Steven Brawer, *Introduction to Parallel Programming*, Academic Press, 1989, ISBN: 0-12-128470-0.

DeRemer and Pennello Article

DeRemer, Frank and Pennello, Thomas J., *Efficient Computation of LALR(1) Look-Ahead Sets*, SigPlan Notices, Volume 15, No. 8, August 1979.

Draft ANSI X3J11.1

IEEE Floating Point draft report of ANSI X3J11.1 (NCEG).

FIPS 151-1

Federal Information Procurement Standard (FIPS) 151-1. Portable Operating System Interface (POSIX)—Part 1: System Application Program Interface (API) [C Language].

FIPS 151-2

Federal Information Procurement Standards (FIPS) 151-2, Portable Operating System Interface (POSIX)—Part 1: System Application Program Interface (API) [C Language].

HP-UX Manual

Hewlett-Packard HP-UX Release 9.0 Reference Manual, Third Edition, August 1992.

IEC 60559: 1989

IEC 60559: 1989, Binary Floating-Point Arithmetic for Microprocessor Systems (previously designated IEC 559: 1989).

IEEE Standards Terms

IEEE 100, The Authoritative Dictionary of IEEE Standards Terms, Seventh Edition.

- IEEE Std 754TM-1985
IEEE Std 754-1985 (Reaff 1990), IEEE Standard for Binary Floating-Point Arithmetic.
- IEEE Std 854TM-1987
IEEE Std 854-1987, IEEE Standard for Radix-Independent Floating-Point Arithmetic.
- IEEE Std 1003.9TM-1992
IEEE Std 1003.9-1992, IEEE Standard for Information Technology — POSIX FORTRAN 77 Language Interfaces — Part 1: Binding for System Application Program Interface API.
- IETF RFC 791
Internet Protocol, Version 4 (IPv4), September 1981 (available at: www.ietf.org/rfc/rfc0791.txt).
- IETF RFC 819
The Domain Naming Convention for Internet User Applications, Z. Su, J. Postel, August 1982 (available at: www.ietf.org/rfc/rfc0819.txt).
- IETF RFC 822
Standard for the Format of ARPA Internet Text Messages, D.H. Crocker, August 1982 (available at: www.ietf.org/rfc/rfc0822.txt).
- IETF RFC 919
Broadcasting Internet Datagrams, J. Mogul, October 1984 (available at: www.ietf.org/rfc/rfc0919.txt).
- IETF RFC 920
Domain Requirements, J. Postel, J. Reynolds, October 1984 (available at: www.ietf.org/rfc/rfc0920.txt).
- IETF RFC 921
Domain Name System Implementation Schedule, J. Postel, October 1984 (available at: www.ietf.org/rfc/rfc0921.txt).
- IETF RFC 922
Broadcasting Internet Datagrams in the Presence of Subnets, J. Mogul, October 1984 (available at: www.ietf.org/rfc/rfc0922.txt).
- IETF RFC 1034
Domain Names — Concepts and Facilities, P. Mockapetris, November 1987 (available at: www.ietf.org/rfc/rfc1034.txt).
- IETF RFC 1035
Domain Names — Implementation and Specification, P. Mockapetris, November 1987 (available at: www.ietf.org/rfc/rfc1035.txt).
- IETF RFC 1123
Requirements for Internet Hosts — Application and Support, R. Braden, October 1989 (available at: www.ietf.org/rfc/rfc1123.txt).
- IETF RFC 1886
DNS Extensions to Support Internet Protocol, Version 6 (IPv6), C. Huitema, S. Thomson, December 1995 (available at: www.ietf.org/rfc/rfc1886.txt).
- IETF RFC 2045
Multipurpose Internet Mail Extensions (MIME), Part 1: Format of Internet Message Bodies, N. Freed, N. Borenstein, November 1996 (available at: www.ietf.org/rfc/rfc2045.txt).

IETF RFC 2181

Clarifications to the DNS Specification, R. Elz, R. Bush, July 1997 (available at: www.ietf.org/rfc/rfc2181.txt).

IETF RFC 2373

Internet Protocol, Version 6 (IPv6) Addressing Architecture, S. Deering, R. Hinden, July 1998 (available at: www.ietf.org/rfc/rfc2373.txt).

IETF RFC 2460

Internet Protocol, Version 6 (IPv6), S. Deering, R. Hinden, December 1998 (available at: www.ietf.org/rfc/rfc2460.txt).

Internationalisation Guide

Guide, July 1993, Internationalisation Guide, Version 2 (ISBN: 1-859120-02-4, G304), published by The Open Group.

ISO 2375: 1985

ISO 2375: 1985, Data Processing — Procedure for Registration of Escape Sequences.

ISO 8652: 1987

ISO 8652: 1987, Programming Languages — Ada (technically identical to ANSI standard 1815A-1983).

ISO/IEC 1539: 1991

ISO/IEC 1539: 1991, Information Technology — Programming Languages — Fortran (technically identical to the ANSI X3.9-1978 standard [FORTRAN 77]).

ISO/IEC 4873: 1991

ISO/IEC 4873: 1991, Information Technology — ISO 8-bit Code for Information Interchange — Structure and Rules for Implementation.

ISO/IEC 6429: 1992

ISO/IEC 6429: 1992, Information Technology — Control Functions for Coded Character Sets.

ISO/IEC 6937: 1994

ISO/IEC 6937: 1994, Information Technology — Coded Graphic Character Set for Text Communication — Latin Alphabet.

ISO/IEC 8802-3: 1996

ISO/IEC 8802-3: 1996, Information Technology — Telecommunications and Information Exchange Between Systems — Local and Metropolitan Area Networks — Specific Requirements — Part 3: Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications.

ISO/IEC 8859

ISO/IEC 8859, Information Technology — 8-Bit Single-Byte Coded Graphic Character Sets:

Part 1: Latin Alphabet No. 1

Part 2: Latin Alphabet No. 2

Part 3: Latin Alphabet No. 3

Part 4: Latin Alphabet No. 4

Part 5: Latin/Cyrillic Alphabet

Part 6: Latin/Arabic Alphabet

Part 7: Latin/Greek Alphabet

Part 8: Latin/Hebrew Alphabet

Part 9: Latin Alphabet No. 5

Part 10: Latin Alphabet No. 6

Part 11: Latin/Thai Alphabet

- Part 13: Latin Alphabet No. 7
- Part 14: Latin Alphabet No. 8 (Celtic)
- Part 15: Latin Alphabet No. 9
- Part 16: Latin Alphabet No. 10

ISO/IEC 9899:1990

ISO/IEC 9899:1990, Programming Languages — C, including Amendment 1:1995 (E), C Integrity (Multibyte Support Extensions (MSE) for ISO C).

ISO POSIX-1:1996

ISO/IEC 9945-1:1996, Information Technology — Portable Operating System Interface (POSIX) — Part 1: System Application Program Interface (API) [C Language] (identical to ANSI/IEEE Std 1003.1-1996). Incorporating ANSI/IEEE Stds 1003.1-1990, 1003.1b-1993, 1003.1c-1995, and 1003.1i-1995.

ISO POSIX-2:1993

ISO/IEC 9945-2:1993, Information Technology — Portable Operating System Interface (POSIX) — Part 2: Shell and Utilities (identical to ANSI/IEEE Std 1003.2™-1992, as amended by ANSI/IEEE Std 1003.2a™-1992).

Issue 1

X/Open Portability Guide, July 1985 (ISBN: 0-444-87839-4).

Issue 2

X/Open Portability Guide, January 1987:

- Volume 1: XVS Commands and Utilities (ISBN: 0-444-70174-5)
- Volume 2: XVS System Calls and Libraries (ISBN: 0-444-70175-3)

Issue 3

X/Open Specification, 1988, 1989, February 1992:

- Commands and Utilities, Issue 3 (ISBN: 1-872630-36-7, C211); this specification was formerly X/Open Portability Guide, Issue 3, Volume 1, January 1989, XSI Commands and Utilities (ISBN: 0-13-685835-X, XO/XPG/89/002)
- System Interfaces and Headers, Issue 3 (ISBN: 1-872630-37-5, C212); this specification was formerly X/Open Portability Guide, Issue 3, Volume 2, January 1989, XSI System Interface and Headers (ISBN: 0-13-685843-0, XO/XPG/89/003)
- Curses Interface, Issue 3, contained in Supplementary Definitions, Issue 3 (ISBN: 1-872630-38-3, C213), Chapters 9 to 14 inclusive; this specification was formerly X/Open Portability Guide, Issue 3, Volume 3, January 1989, XSI Supplementary Definitions (ISBN: 0-13-685850-3, XO/XPG/89/004)
- Headers Interface, Issue 3, contained in Supplementary Definitions, Issue 3 (ISBN: 1-872630-38-3, C213), Chapter 19, Cpio and Tar Headers; this specification was formerly X/Open Portability Guide Issue 3, Volume 3, January 1989, XSI Supplementary Definitions (ISBN: 0-13-685850-3, XO/XPG/89/004)

Issue 4

CAE Specification, July 1992, published by The Open Group:

- System Interface Definitions (XBD), Issue 4 (ISBN: 1-872630-46-4, C204)
- Commands and Utilities (XCU), Issue 4 (ISBN: 1-872630-48-0, C203)
- System Interfaces and Headers (XSH), Issue 4 (ISBN: 1-872630-47-2, C202)

Referenced Documents

Issue 4, Version 2

CAE Specification, August 1994, published by The Open Group:

- System Interface Definitions (XBD), Issue 4, Version 2 (ISBN: 1-85912-036-9, C434)
- Commands and Utilities (XCU), Issue 4, Version 2 (ISBN: 1-85912-034-2, C436)
- System Interfaces and Headers (XSH), Issue 4, Version 2 (ISBN: 1-85912-037-7, C435)

Issue 5

Technical Standard, February 1997, published by The Open Group:

- System Interface Definitions (XBD), Issue 5 (ISBN: 1-85912-186-1, C605)
- Commands and Utilities (XCU), Issue 5 (ISBN: 1-85912-191-8, C604)
- System Interfaces and Headers (XSH), Issue 5 (ISBN: 1-85912-181-0, C606)

Issue 6

Technical Standard, April 2004, published by The Open Group:

- Base Definitions (XBD), Issue 6 (ISBN: 1-931624-43-7, C046)
- System Interfaces (XSH), Issue 6 (ISBN: 1-931624-44-5, C047)
- Shell and Utilities (XCU), Issue 6 (ISBN: 1-931624-45-3, C048)

Knuth Article

Knuth, Donald E., *On the Translation of Languages from Left to Right*, Information and Control, Volume 8, No. 6, October 1965.

KornShell

Bolsky, Morris I. and Korn, David G., *The New KornShell Command and Programming Language*, March 1995, Prentice Hall.

MSE Working Draft

Working draft of ISO/IEC 9899:1990/Add3:Draft, Addendum 3 — Multibyte Support Extensions (MSE) as documented in the ISO Working Paper SC22/WG14/N205 dated 31 March 1992.

POSIX.0: 1995

IEEE Std 1003.0™-1995, IEEE Guide to the POSIX Open System Environment (OSE) (identical to ISO/IEC TR 14252).

POSIX.1: 1988

IEEE Std 1003.1™-1988, IEEE Standard for Information Technology — Portable Operating System Interface (POSIX) — Part 1: System Application Program Interface (API) [C Language].

POSIX.1: 1990

IEEE Std 1003.1™-1990, IEEE Standard for Information Technology — Portable Operating System Interface (POSIX) — Part 1: System Application Program Interface (API) [C Language].

POSIX.1a

P1003.1a, Standard for Information Technology — Portable Operating System Interface (POSIX) — Part 1: System Application Program Interface (API) — (C Language) Amendment.

POSIX.1d: 1999

IEEE Std 1003.1d™-1999, IEEE Standard for Information Technology — Portable Operating System Interface (POSIX) — Part 1: System Application Program Interface (API) —

- Amendment 4: Additional Realtime Extensions [C Language].
- POSIX.1g: 2000
IEEE Std 1003.1g™-2000, IEEE Standard for Information Technology — Portable Operating System Interface (POSIX) — Part 1: System Application Program Interface (API) — Amendment 6: Protocol-Independent Interfaces (PII).
- POSIX.1j: 2000
IEEE Std 1003.1j™-2000, IEEE Standard for Information Technology — Portable Operating System Interface (POSIX) — Part 1: System Application Program Interface (API) — Amendment 5: Advanced Realtime Extensions [C Language].
- POSIX.1q: 2000
IEEE Std 1003.1q™-2000, IEEE Standard for Information Technology — Portable Operating System Interface (POSIX) — Part 1: System Application Program Interface (API) — Amendment 7: Tracing [C Language].
- POSIX.2: 1992
IEEE Std 1003.2™-1992, IEEE Standard for Information Technology — Portable Operating System Interface (POSIX) — Part 2: Shell and Utilities.
- POSIX.2b
P1003.2b, Standard for Information Technology — Portable Operating System Interface (POSIX) — Part 2: Shell and Utilities — Amendment.
- POSIX.2d: 1994
IEEE Std 1003.2d™-1994, IEEE Standard for Information Technology — Portable Operating System Interface (POSIX) — Part 2: Shell and Utilities — Amendment 1: Batch Environment.
- POSIX.13: 1998
IEEE Std 1003.13™-1998, IEEE Standard for Information Technology — Standardized Application Environment Profile (AEP) — POSIX Realtime Application Support.
- Sarwate Article
Sarwate, Dilip V., *Computation of Cyclic Redundancy Checks via Table Lookup*, Communications of the ACM, Volume 30, No. 8, August 1988.
- Sprunt, Sha, and Lehoczky
Sprunt, B., Sha, L., and Lehoczky, J.P., *Aperiodic Task Scheduling for Hard Real-Time Systems*, The Journal of Real-Time Systems, Volume 1, 1989, Pages 27-60.
- SVID, Issue 1
American Telephone and Telegraph Company, System V Interface Definition (SVID), Issue 1; Morristown, NJ, UNIX Press, 1985.
- SVID, Issue 2
American Telephone and Telegraph Company, System V Interface Definition (SVID), Issue 2; Morristown, NJ, UNIX Press, 1986.
- SVID, Issue 3
American Telephone and Telegraph Company, System V Interface Definition (SVID), Issue 3; Morristown, NJ, UNIX Press, 1989.
- The AWK Programming Language
Aho, Alfred V., Kernighan, Brian W., and Weinberger, Peter J., *The AWK Programming Language*, Reading, MA, Addison-Wesley 1988.
- The C Programming Language
Kernighan, Brian W. and Ritchie, Dennis M., *The C Programming Language*, Englewood Cliffs, NJ, Prentice Hall, 1st Edition (February 1978) ISBN 0-13-110163-3; 2nd Edition (March

Referenced Documents

- 1988) ISBN 0-13-110362-8.
- UNIX Programmer's Manual
American Telephone and Telegraph Company, *UNIX Time-Sharing System: UNIX Programmer's Manual*, 7th Edition, Murray Hill, NJ, Bell Telephone Laboratories, January 1979.
- XNS, Issue 4
CAE Specification, August 1994, Networking Services, Issue 4 (ISBN: 1-85912-049-0, C438), published by The Open Group.
- XNS, Issue 5
CAE Specification, February 1997, Networking Services, Issue 5 (ISBN: 1-85912-165-9, C523), published by The Open Group.
- XNS, Issue 5.2
Technical Standard, January 2000, Networking Services (XNS), Issue 5.2 (ISBN: 1-85912-241-8, C808), published by The Open Group.
- X/Open Curses, Issue 4, Version 2
CAE Specification, May 1996, X/Open Curses, Issue 4, Version 2 (ISBN: 1-85912-171-3, C610), published by The Open Group.
- Yacc
Yacc: Yet Another Compiler Compiler, Stephen C. Johnson, 1978.

Source Documents

Parts of the following documents were used to create the base documents for POSIX.1-2008:

- AIX 3.2 Manual
AIX Version 3.2 For RISC System/6000, Technical Reference: Base Operating System and Extensions, 1990, 1992 (Part No. SC23-2382-00).
- OSF/1
OSF/1 Programmer's Reference, Release 1.2 (ISBN: 0-13-020579-6).
- OSF AES
Application Environment Specification (AES) Operating System Programming Interfaces Volume, Revision A (ISBN: 0-13-043522-8).
- System V Release 2.0
— UNIX System V Release 2.0 Programmer's Reference Manual (April 1984 - Issue 2).
— UNIX System V Release 2.0 Programming Guide (April 1984 - Issue 2).
- System V Release 4.2
Operating System API Reference, UNIX[®] SVR4.2 (1992) (ISBN: 0-13-017658-3).

Standard for Information Technology— Portable Operating System Interface (POSIX[®])

Base Specifications, Issue 7

Prepared by the Austin Group (www.opengroup.org/austin).

IMPORTANT NOTICE: This standard is not intended to assure safety, security, health, or environmental protection in all circumstances. Implementors of the standard are responsible for determining appropriate safety, security, environmental, and health practices or regulatory requirements.

This IEEE document is made available for use subject to important notices and legal disclaimers. These notices and disclaimers appear in all publications containing this document and may be found under the heading “Important Notice” or “Important Notices and Disclaimers Concerning IEEE Documents”. They can also be obtained on request from IEEE or viewed at <http://standards.ieee.org/IPR/disclaimers.html>.

Technical Standard

1 **Vol. 1:**
2 **Base Definitions, Issue 7**

3 *The Open Group*
4 *The Institute of Electrical and Electronics Engineers, Inc.*

1.1 Scope

POSIX.1-2008 defines a standard operating system interface and environment, including a command interpreter (or “shell”), and common utility programs to support applications portability at the source code level. It is intended to be used by both application developers and system implementors.

POSIX.1-2008 comprises four major components (each in an associated volume):

1. General terms, concepts, and interfaces common to all volumes of POSIX.1-2008, including utility conventions and C-language header definitions, are included in the Base Definitions volume of POSIX.1-2008.
2. Definitions for system service functions and subroutines, language-specific system services for the C programming language, function issues, including portability, error handling, and error recovery, are included in the System Interfaces volume of POSIX.1-2008.
3. Definitions for a standard source code-level interface to command interpretation services (a “shell”) and common utility programs for application programs are included in the Shell and Utilities volume of POSIX.1-2008.
4. Extended rationale that did not fit well into the rest of the document structure, containing historical information concerning the contents of POSIX.1-2008 and why features were included or discarded by the standard developers, is included in the Rationale (Informative) volume of POSIX.1-2008.

The following areas are outside of the scope of POSIX.1-2008:

- Graphics interfaces
- Database management system interfaces
- Record I/O considerations
- Object or binary code portability
- System configuration and resource availability

POSIX.1-2008 describes the external characteristics and facilities that are of importance to application developers, rather than the internal construction techniques employed to achieve these capabilities. Special emphasis is placed on those functions and facilities that are needed in a wide variety of commercial applications.

The facilities provided in POSIX.1-2008 are drawn from the following base documents:

- IEEE Std 1003.1, 2004 Edition (POSIX-1) (incorporating IEEE Std 1003.1-2001, IEEE Std 1003.1-2001/Cor 1-2002, and IEEE Std 1003.1-2001/Cor 2-2004)

- The Open Group Technical Standard, 2006, Extended API Set Part 1
- The Open Group Technical Standard, 2006, Extended API Set Part 2
- The Open Group Technical Standard, 2006, Extended API Set Part 3
- The Open Group Technical Standard, 2006, Extended API Set Part 4
- ISO/IEC 9899:1999, Programming Languages — C, including ISO/IEC 9899:1999/Cor.1:2001(E), ISO/IEC 9899:1999/Cor.2:2004(E), and ISO/IEC 9899:1999/Cor.3

Emphasis has been placed on standardizing existing practice for existing users, with changes and additions limited to correcting deficiencies in the following areas:

- Issues raised by Austin Group defect reports, IEEE Interpretations against IEEE Std 1003.1, and ISO/IEC defect reports against ISO/IEC 9945
- Issues raised in corrigenda for The Open Group Technical Standards and working group resolutions from The Open Group
- Issues arising from ISO TR 24715:2006, Conflicts between POSIX and the LSB
- Changes to make the text self-consistent with the additional material merged
- Features, marked Legacy or obsolescent in the base documents, have been considered for removal in this version
- A review and reorganization of the options within the standard
- Alignment with the ISO/IEC 9899:1999 standard, including ISO/IEC 9899:1999/Cor.2:2004(E)

1.2 Conformance

Conformance requirements for POSIX.1-2008 are defined in [Chapter 2](#) (on page 15).

1.3 Normative References

The following standards contain provisions which, through references in POSIX.1-2008, constitute provisions of POSIX.1-2008. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on POSIX.1-2008 are encouraged to investigate the possibility of applying the most recent editions of the standards listed below. Members of IEC and ISO maintain registers of currently valid International Standards.

ANS X3.9-1978

(Reaffirmed 1989) American National Standard for Information Systems: Standard X3.9-1978, Programming Language FORTRAN.¹

ISO/IEC 646:1991

ISO/IEC 646:1991, Information Processing — ISO 7-Bit Coded Character Set for Information Interchange.²

1. ANSI documents can be obtained from the Sales Department, American National Standards Institute, 1430 Broadway, New York, NY 10018, USA.

- ISO 4217:2001
ISO 4217:2001, Codes for the Representation of Currencies and Funds.
- ISO 8601:2004
ISO 8601:2004, Data Elements and Interchange Formats — Information Interchange — Representation of Dates and Times.
- ISO C (1999)
ISO/IEC 9899:1999, Programming Languages — C, including ISO/IEC 9899:1999/Cor.1:2001(E), ISO/IEC 9899:1999/Cor.2:2004(E), and ISO/IEC 9899:1999/Cor.3.
- ISO/IEC 10646-1:2000
ISO/IEC 10646-1:2000, Information Technology — Universal Multiple-Octet Coded Character Set (UCS) — Part 1: Architecture and Basic Multilingual Plane.

1.4 Change History

Change history is described in the Rationale (Informative) volume of POSIX.1-2008, and in the CHANGE HISTORY section of reference pages.

1.5 Terminology

For the purposes of POSIX.1-2008, the following terminology definitions apply:

can

Describes a permissible optional feature or behavior available to the user or application. The feature or behavior is mandatory for an implementation that conforms to POSIX.1-2008. An application can rely on the existence of the feature or behavior.

implementation-defined

Describes a value or behavior that is not defined by POSIX.1-2008 but is selected by an implementor. The value or behavior may vary among implementations that conform to POSIX.1-2008. An application should not rely on the existence of the value or behavior. An application that relies on such a value or behavior cannot be assured to be portable across conforming implementations.

The implementor shall document such a value or behavior so that it can be used correctly by an application.

legacy

Describes a feature or behavior that is being retained for compatibility with older applications, but which has limitations which make it inappropriate for developing portable applications. New applications should use alternative means of obtaining equivalent functionality.

may

Describes a feature or behavior that is optional for an implementation that conforms to POSIX.1-2008. An application should not rely on the existence of the feature or behavior. An application that relies on such a feature or behavior cannot be assured to be portable across conforming implementations.

2. ISO/IEC documents can be obtained from the ISO office: 1 Rue de Varembe, Case Postale 56, CH-1211, Genève 20, Switzerland/Suisse

To avoid ambiguity, the opposite of *may* is expressed as *need not*, instead of *may not*.

shall

For an implementation that conforms to POSIX.1-2008, describes a feature or behavior that is mandatory. An application can rely on the existence of the feature or behavior.

For an application or user, describes a behavior that is mandatory.

should

For an implementation that conforms to POSIX.1-2008, describes a feature or behavior that is recommended but not mandatory. An application should not rely on the existence of the feature or behavior. An application that relies on such a feature or behavior cannot be assured to be portable across conforming implementations.

For an application, describes a feature or behavior that is recommended programming practice for optimum portability.

undefined

Describes the nature of a value or behavior not defined by POSIX.1-2008 which results from use of an invalid program construct or invalid data input.

The value or behavior may vary among implementations that conform to POSIX.1-2008. An application should not rely on the existence or validity of the value or behavior. An application that relies on any particular value or behavior cannot be assured to be portable across conforming implementations.

unspecified

Describes the nature of a value or behavior not specified by POSIX.1-2008 which results from use of a valid program construct or valid data input.

The value or behavior may vary among implementations that conform to POSIX.1-2008. An application should not rely on the existence or validity of the value or behavior. An application that relies on any particular value or behavior cannot be assured to be portable across conforming implementations.

1.6 Definitions and Concepts

Definitions and concepts are defined in [Chapter 3](#) (on page 33) and [Chapter 4](#) (on page 107).

1.7 Portability

Some of the utilities in the Shell and Utilities volume of POSIX.1-2008 and functions in the System Interfaces volume of POSIX.1-2008 describe functionality that might not be fully portable to systems meeting the requirements for POSIX conformance (see [Chapter 2](#), on page 15).

Where optional, enhanced, or reduced functionality is specified, the text is shaded and a code in the margin identifies the nature of the option, extension, or warning (see [Section 1.7.1](#), on page 7). For maximum portability, an application should avoid such functionality.

Unless the primary task of a utility is to produce textual material on its standard output, application developers should not rely on the format or content of any such material that may be produced. Where the primary task *is* to provide such material, but the output format is incompletely specified, the description is marked with the OF margin code and shading. Application developers are warned not to expect that the output of such an interface on one

system is any guide to its behavior on another system.

1.7.1 Codes

The codes and their meanings are as follows. See also [Section 1.7.2](#) (on page 13).

ADV Advisory Information

The functionality described is optional. The functionality described is also an extension to the ISO C standard.

Where applicable, functions are marked with the ADV margin legend in the SYNOPSIS section. Where additional semantics apply to a function, the material is identified by use of the ADV margin legend.

BE Batch Environment Services and Utilities

The functionality described is optional.

Where applicable, utilities are marked with the BE margin legend in the SYNOPSIS section. Where additional semantics apply to a utility, the material is identified by use of the BE margin legend.

CD C-Language Development Utilities

The functionality described is optional.

Where applicable, utilities are marked with the CD margin legend in the SYNOPSIS section. Where additional semantics apply to a utility, the material is identified by use of the CD margin legend.

CPT Process CPU-Time Clocks

The functionality described is optional. The functionality described is also an extension to the ISO C standard.

Where applicable, functions are marked with the CPT margin legend in the SYNOPSIS section. Where additional semantics apply to a function, the material is identified by use of the CPT margin legend.

CX Extension to the ISO C standard

The functionality described is an extension to the ISO C standard. Application developers may make use of an extension as it is supported on all POSIX.1-2008-conforming systems.

With each function or header from the ISO C standard, a statement to the effect that “any conflict is unintentional” is included. That is intended to refer to a direct conflict. POSIX.1-2008 acts in part as a profile of the ISO C standard, and it may choose to further constrain behaviors allowed to vary by the ISO C standard. Such limitations and other compatible differences are not considered conflicts, even if a CX mark is missing. The markings are for information only.

Where additional semantics apply to a function or header, the material is identified by use of the CX margin legend.

FD FORTRAN Development Utilities

The functionality described is optional.

Where applicable, utilities are marked with the FD margin legend in the SYNOPSIS section. Where additional semantics apply to a utility, the material is identified by use of the FD margin legend.

FR FORTRAN Runtime Utilities

The functionality described is optional.

Where applicable, utilities are marked with the FR margin legend in the SYNOPSIS section.

200		Where additional semantics apply to a utility, the material is identified by use of the FR margin legend.
201		
202	FSC	File Synchronization
203		The functionality described is optional. The functionality described is also an extension to the
204		ISO C standard.
205		Where applicable, functions are marked with the FSC margin legend in the SYNOPSIS section.
206		Where additional semantics apply to a function, the material is identified by use of the FSC
207		margin legend.
208	IP6	IPV6
209		The functionality described is optional. The functionality described is also an extension to the
210		ISO C standard.
211		Where applicable, functions are marked with the IP6 margin legend in the SYNOPSIS section.
212		Where additional semantics apply to a function, the material is identified by use of the IP6
213		margin legend.
214	MC1	Non-Robust Mutex Priority Protection or Non-Robust Mutex Priority Inheritance or Robust
215		Mutex Priority Protection or Robust Mutex Priority Inheritance
216		The functionality described is optional. The functionality described is also an extension to the
217		ISO C standard.
218		This is a shorthand notation for combinations of multiple option codes.
219		Where applicable, functions are marked with the MC1 margin legend in the SYNOPSIS section.
220		Where additional semantics apply to a function, the material is identified by use of the MC1
221		margin legend.
222		Refer to Section 1.7.2 (on page 13).
223	ML	Process Memory Locking
224		The functionality described is optional. The functionality described is also an extension to the
225		ISO C standard.
226		Where applicable, functions are marked with the ML margin legend in the SYNOPSIS section.
227		Where additional semantics apply to a function, the material is identified by use of the ML
228		margin legend.
229	MLR	Range Memory Locking
230		The functionality described is optional. The functionality described is also an extension to the
231		ISO C standard.
232		Where applicable, functions are marked with the MLR margin legend in the SYNOPSIS section.
233		Where additional semantics apply to a function, the material is identified by use of the MLR
234		margin legend.
235	MON	Monotonic Clock
236		The functionality described is optional. The functionality described is also an extension to the
237		ISO C standard.
238		Where applicable, functions are marked with the MON margin legend in the SYNOPSIS section.
239		Where additional semantics apply to a function, the material is identified by use of the MON
240		margin legend.
241	MSG	Message Passing
242		The functionality described is optional. The functionality described is also an extension to the
243		ISO C standard.
244		Where applicable, functions are marked with the MSG margin legend in the SYNOPSIS section.

245		Where additional semantics apply to a function, the material is identified by use of the MSG
246		margin legend.
247	MX	IEC 60559 Floating-Point
248		The functionality described is optional. The functionality described is mandated by the ISO C
249		standard only for implementations that define <code>__STDC_IEC_559__</code> .
250	MXX	IEC 60559 Floating-Point Extension
251		The functionality described is optional. The functionality described is part of the IEC 60559
252		Floating-Point option, but is an extension to the ISO C standard.
253	OB	Obsolescent
254		The functionality described may be removed in a future version of this volume of POSIX.1-2008.
255		Strictly Conforming POSIX Applications and Strictly Conforming XSI Applications shall not use
256		obsolescent features.
257		Where applicable, the material is identified by use of the OB margin legend.
258	OF	Output Format Incompletely Specified
259		The functionality described is an XSI extension. The format of the output produced by the
260		utility is not fully specified. It is therefore not possible to post-process this output in a consistent
261		fashion. Typical problems include unknown length of strings and unspecified field delimiters.
262		Where applicable, the material is identified by use of the OF margin legend.
263	OH	Optional Header
264		In the SYNOPSIS section of some interfaces in the System Interfaces volume of POSIX.1-2008 an
265		included header is marked as in the following example:
266	OH	<pre>#include <sys/types.h></pre>
267		<pre>#include <grp.h></pre>
268		<pre>struct group *getgrnam(const char *name);</pre>
269		The OH margin legend indicates that the marked header is not required on XSI-conformant
270		systems.
271	PIO	Prioritized Input and Output
272		The functionality described is optional. The functionality described is also an extension to the
273		ISO C standard.
274		Where applicable, functions are marked with the PIO margin legend in the SYNOPSIS section.
275		Where additional semantics apply to a function, the material is identified by use of the PIO
276		margin legend.
277	PS	Process Scheduling
278		The functionality described is optional. The functionality described is also an extension to the
279		ISO C standard.
280		Where applicable, functions are marked with the PS margin legend in the SYNOPSIS section.
281		Where additional semantics apply to a function, the material is identified by use of the PS
282		margin legend.
283	RPI	Robust Mutex Priority Inheritance
284		The functionality described is optional. The functionality described is also an extension to the
285		ISO C standard.
286		Where applicable, functions are marked with the RPI margin legend in the SYNOPSIS section.
287		Where additional semantics apply to a function, the material is identified by use of the RPI
288		margin legend.

289	RPP	Robust Mutex Priority Protection
290		The functionality described is optional. The functionality described is also an extension to the
291		ISO C standard.
292		Where applicable, functions are marked with the RPP margin legend in the SYNOPSIS section.
293		Where additional semantics apply to a function, the material is identified by use of the RPP
294		margin legend.
295	RS	Raw Sockets
296		The functionality described is optional. The functionality described is also an extension to the
297		ISO C standard.
298		Where applicable, functions are marked with the RS margin legend in the SYNOPSIS section.
299		Where additional semantics apply to a function, the material is identified by use of the RS
300		margin legend.
301	SD	Software Development Utilities
302		The functionality described is optional.
303		Where applicable, utilities are marked with the SD margin legend in the SYNOPSIS section.
304		Where additional semantics apply to a utility, the material is identified by use of the SD margin
305		legend.
306	SHM	Shared Memory Objects
307		The functionality described is optional. The functionality described is also an extension to the
308		ISO C standard.
309		Where applicable, functions are marked with the SHM margin legend in the SYNOPSIS section.
310		Where additional semantics apply to a function, the material is identified by use of the SHM
311		margin legend.
312	SIO	Synchronized Input and Output
313		The functionality described is optional. The functionality described is also an extension to the
314		ISO C standard.
315		Where applicable, functions are marked with the SIO margin legend in the SYNOPSIS section.
316		Where additional semantics apply to a function, the material is identified by use of the SIO
317		margin legend.
318	SPN	Spawn
319		The functionality described is optional. The functionality described is also an extension to the
320		ISO C standard.
321		Where applicable, functions are marked with the SPN margin legend in the SYNOPSIS section.
322		Where additional semantics apply to a function, the material is identified by use of the SPN
323		margin legend.
324	SS	Process Sporadic Server
325		The functionality described is optional. The functionality described is also an extension to the
326		ISO C standard.
327		Where applicable, functions are marked with the SS margin legend in the SYNOPSIS section.
328		Where additional semantics apply to a function, the material is identified by use of the SS
329		margin legend.
330	TCT	Thread CPU-Time Clocks
331		The functionality described is optional. The functionality described is also an extension to the
332		ISO C standard.
333		Where applicable, functions are marked with the TCT margin legend in the SYNOPSIS section.

334		Where additional semantics apply to a function, the material is identified by use of the TCT
335		margin legend.
336	TEF	Trace Event Filter
337		The functionality described is optional. This functionality is dependent on support for the Trace
338		option. The functionality described is also an extension to the ISO C standard.
339		Where applicable, functions are marked with the TEF margin legend in the SYNOPSIS section.
340		Where additional semantics apply to a function, the material is identified by use of the TEF
341		margin legend.
342	TPI	Non-Robust Mutex Priority Inheritance
343		The functionality described is optional. The functionality described is also an extension to the
344		ISO C standard.
345		Where applicable, functions are marked with the TPI margin legend in the SYNOPSIS section.
346		Where additional semantics apply to a function, the material is identified by use of the TPI
347		margin legend.
348	TPP	Non-Robust Mutex Priority Protection
349		The functionality described is optional. The functionality described is also an extension to the
350		ISO C standard.
351		Where applicable, functions are marked with the TPP margin legend in the SYNOPSIS section.
352		Where additional semantics apply to a function, the material is identified by use of the TPP
353		margin legend.
354	TPS	Thread Execution Scheduling
355		The functionality described is optional. The functionality described is also an extension to the
356		ISO C standard.
357		Where applicable, functions are marked with the TPS margin legend for the SYNOPSIS section.
358		Where additional semantics apply to a function, the material is identified by use of the TPS
359		margin legend.
360	TRC	Trace
361		The functionality described is optional. The functionality described is also an extension to the
362		ISO C standard.
363		Where applicable, functions are marked with the TRC margin legend in the SYNOPSIS section.
364		Where additional semantics apply to a function, the material is identified by use of the TRC
365		margin legend.
366	TRI	Trace Inherit
367		The functionality described is optional. This functionality is dependent on support for the Trace
368		option. The functionality described is also an extension to the ISO C standard.
369		Where applicable, functions are marked with the TRI margin legend in the SYNOPSIS section.
370		Where additional semantics apply to a function, the material is identified by use of the TRI
371		margin legend.
372	TRL	Trace Log
373		The functionality described is optional. This functionality is dependent on support for the Trace
374		option. The functionality described is also an extension to the ISO C standard.
375		Where applicable, functions are marked with the TRL margin legend in the SYNOPSIS section.
376		Where additional semantics apply to a function, the material is identified by use of the TRL
377		margin legend.

378	TSA	Thread Stack Address Attribute
379		The functionality described is optional. The functionality described is also an extension to the
380		ISO C standard.
381		Where applicable, functions are marked with the TSA margin legend for the SYNOPSIS section.
382		Where additional semantics apply to a function, the material is identified by use of the TSA
383		margin legend.
384	TSH	Thread Process-Shared Synchronization
385		The functionality described is optional. The functionality described is also an extension to the
386		ISO C standard.
387		Where applicable, functions are marked with the TSH margin legend in the SYNOPSIS section.
388		Where additional semantics apply to a function, the material is identified by use of the TSH
389		margin legend.
390	TSP	Thread Sporadic Server
391		The functionality described is optional. The functionality described is also an extension to the
392		ISO C standard.
393		Where applicable, functions are marked with the TSP margin legend in the SYNOPSIS section.
394		Where additional semantics apply to a function, the material is identified by use of the TSP
395		margin legend.
396	TSS	Thread Stack Size Attribute
397		The functionality described is optional. The functionality described is also an extension to the
398		ISO C standard.
399		Where applicable, functions are marked with the TSS margin legend in the SYNOPSIS section.
400		Where additional semantics apply to a function, the material is identified by use of the TSS
401		margin legend.
402	TYM	Typed Memory Objects
403		The functionality described is optional. The functionality described is also an extension to the
404		ISO C standard.
405		Where applicable, functions are marked with the TYM margin legend in the SYNOPSIS section.
406		Where additional semantics apply to a function, the material is identified by use of the TYM
407		margin legend.
408	UP	User Portability Utilities
409		The functionality described is optional.
410		Where applicable, utilities are marked with the UP margin legend in the SYNOPSIS section.
411		Where additional semantics apply to a utility, the material is identified by use of the UP margin
412		legend.
413	UU	UUCP Utilities
414		The functionality described is optional. The functionality described is also an extension to the
415		ISO C standard.
416		Where applicable, functions are marked with the UU margin legend in the SYNOPSIS section.
417		Where additional semantics apply to a function, the material is identified by use of the UU
418		margin legend.
419	XSI	X/Open System Interfaces
420		The functionality described is part of the X/Open Systems Interfaces option. Functionality
421		marked XSI is an extension to the ISO C standard. Application developers may confidently
422		make use of such extensions on all systems supporting the X/Open System Interfaces option.

If an entire SYNOPSIS section is shaded and marked XSI, all the functionality described in that reference page is an extension. See [Section 2.1.4](#) (on page 19).

XSR

XSI STREAMS

The functionality described is optional. The functionality described is also an extension to the ISO C standard.

Where applicable, functions are marked with the XSR margin legend in the SYNOPSIS section. Where additional semantics apply to a function, the material is identified by use of the XSR margin legend.

1.7.2 Margin Code Notation

Some of the functionality described in POSIX.1-2008 depends on support of more than one option, or independently may depend on several options. The following notation for margin codes is used to denote the following cases.

A Feature Dependent on One or Two Options

In this case, margin codes have a <space> separator; for example:

SHM

This feature requires support for only the Shared Memory Objects option.

SHM TYM

This feature requires support for both the Shared Memory Objects option and the Typed Memory Objects option; that is, an application which uses this feature is portable only between implementations that provide both options.

A Feature Dependent on Either of the Options Denoted

In this case, margin codes have a ' | ' separator to denote the logical OR; for example:

SHM | TYM

This feature is dependent on support for either the Shared Memory Objects option or the Typed Memory Objects option; that is, an application which uses this feature is portable between implementations that provide any (or all) of the options.

A Feature Dependent on More than Two Options

The following shorthand notations are used:

MC1

The MC1 margin code is shorthand for TPP | TPI | RPP | RPI. Features which are shaded with this margin code require support of either the Non-Robust Mutex Priority Protection option or the Non-Robust Mutex Priority Inheritance option or the Robust Mutex Priority Protection option or the Robust Mutex Priority Inheritance option.

Large Sections Dependent on an Option

Where large sections of text are dependent on support for an option, a lead-in text block is provided and shaded accordingly; for example:

XSI

This section describes extensions to support interprocess communication. The functionality described in this section shall be provided on implementations that support the XSI option (and the rest of this section is not further shaded).

Conformance

460

2.1 Implementation Conformance

461

For the purposes of POSIX.1-2008, the implementation conformance requirements given in this section apply.

462

463

2.1.1 Requirements

464

A *conforming implementation* shall meet all of the following criteria:

465

1. The system shall support all utilities, functions, and facilities defined within POSIX.1-2008 that are required for POSIX conformance (see [Section 2.1.3](#), on page 16). These interfaces shall support the functional behavior described herein.

466

467

468

2. The system may support the X/Open System Interfaces (XSI) option as described in [Section 2.1.4](#) (on page 19).

469

470

3. The system may support one or more options as described under [Section 2.1.5](#) (on page 20). When an implementation claims that an option is supported, all of its constituent parts shall be provided.

471

472

473

4. The system may provide non-standard extensions. These are features not required by POSIX.1-2008 and may include, but are not limited to:

474

475

- Additional functions

476

- Additional headers

477

- Additional symbols in standard headers

478

- Additional utilities

479

- Additional options for standard utilities

480

- Additional environment variables

481

- Additional file types

482

- Non-conforming file systems (for example, legacy file systems for which `_POSIX_NO_TRUNC` is false, case-insensitive file systems, or network file systems)

483

484

- Dynamically populated file systems (for example, `/proc`)

485

- Additional character special files with special properties (for example, `/dev/stdin`, `/dev/stdout`, and `/dev/stderr`)

486

487

Non-standard extensions of the utilities, functions, or facilities specified in POSIX.1-2008 should be identified as such in the system documentation. Non-standard extensions, when used, may change the behavior of utilities, functions, or facilities defined by POSIX.1-2008. The conformance document shall define an environment in which an application can be run with the behavior specified by POSIX.1-2008. In no case shall such

488

489

490

491

an environment require modification of a Strictly Conforming POSIX Application (see [Section 2.2.1](#), on page 29).

2.1.2 Documentation

A conformance document with the following information shall be available for an implementation claiming conformance to POSIX.1-2008. The conformance document shall have the same structure as POSIX.1-2008, with the information presented in the appropriate sections and subsections. Sections and subsections that consist solely of subordinate section titles, with no other information, are not required. The conformance document shall not contain information about extended facilities or capabilities outside the scope of POSIX.1-2008.

The conformance document shall contain a statement that indicates the full name, number, and date of the standard that applies. The conformance document may also list international software standards that are available for use by a Conforming POSIX Application. Applicable characteristics where documentation is required by one of these standards, or by standards of government bodies, may also be included.

The conformance document shall describe the limit values found in the headers [<limits.h>](#) (on page 270) and [<unistd.h>](#) (on page 433), stating values, the conditions under which those values may change, and the limits of such variations, if any.

The conformance document shall describe the behavior of the implementation for all implementation-defined features defined in POSIX.1-2008. This requirement shall be met by listing these features and providing either a specific reference to the system documentation or providing full syntax and semantics of these features. When the value or behavior in the implementation is designed to be variable or customized on each instantiation of the system, the implementation provider shall document the nature and permissible ranges of this variation.

The conformance document may specify the behavior of the implementation for those features where POSIX.1-2008 states that implementations may vary or where features are identified as undefined or unspecified.

The conformance document shall not contain documentation other than that specified in the preceding paragraphs except where such documentation is specifically allowed or required by other provisions of POSIX.1-2008.

The phrases “shall document” or “shall be documented” in POSIX.1-2008 mean that documentation of the feature shall appear in the conformance document, as described previously, unless there is an explicit reference in the conformance document to show where the information can be found in the system documentation.

The system documentation should also contain the information found in the conformance document.

2.1.3 POSIX Conformance

A conforming implementation shall meet the following criteria for POSIX conformance.

2.1.3.1 POSIX System Interfaces

The following requirements apply to the system interfaces (functions and headers):

- The system shall support all the mandatory functions and headers defined in POSIX.1-2008, and shall set the symbolic constant `_POSIX_VERSION` to the value 200809L.
- Although all implementations conforming to POSIX.1-2008 support all the features described below, there may be system-dependent or file system-dependent configuration procedures that can remove or modify any or all of these features. Such configurations should not be made if strict compliance is required.

The following symbolic constants shall be defined with a value other than `-1`. If a constant is defined with the value zero, applications should use the `sysconf()`, `pathconf()`, or `fpathconf()` functions, or the `getconf` utility, to determine which features are present on the system at that time or for the particular pathname in question.

— `_POSIX_CHOWN_RESTRICTED`

The use of `chown()` is restricted to a process with appropriate privileges, and to changing the group ID of a file only to the effective group ID of the process or to one of its supplementary group IDs.

— `_POSIX_NO_TRUNC`

Pathname components longer than `{NAME_MAX}` generate an error.

- The following symbolic constants shall be defined by the implementation as follows:

— Symbolic constants defined with the value 200809L:

```

_POSIX_ASYNCHRONOUS_IO
_POSIX_BARRIERS
_POSIX_CLOCK_SELECTION
_POSIX_MAPPED_FILES
_POSIX_MEMORY_PROTECTION
_POSIX_READER_WRITER_LOCKS
_POSIX_REALTIME_SIGNALS
_POSIX_SEMAPHORES
_POSIX_SPIN_LOCKS
_POSIX_THREAD_SAFE_FUNCTIONS
_POSIX_THREADS
_POSIX_TIMEOUTS
_POSIX_TIMERS

```

— Symbolic constants defined with a value greater than zero:

```

_POSIX_JOB_CONTROL
_POSIX_SAVED_IDS

```

— Symbolic constants defined with a value other than `-1`.

```

_POSIX_VDISABLE

```

Note: The symbols above represent historical options that are no longer allowed as options, but are retained here for backwards-compatibility of applications.

- The system may support one or more options (see [Section 2.1.6](#), on page 26) denoted by the following symbolic constants:

```

_POSIX_ADVISORY_INFO
_POSIX_CPUTIME
_POSIX_FSYNC
_POSIX_IPV6
_POSIX_MEMLOCK
_POSIX_MEMLOCK_RANGE
_POSIX_MESSAGE_PASSING
_POSIX_MONOTONIC_CLOCK
_POSIX_PRIORITIZED_IO
_POSIX_PRIORITY_SCHEDULING
_POSIX_RAW_SOCKETS
_POSIX_SHARED_MEMORY_OBJECTS
_POSIX_SPAWN
_POSIX_SPORADIC_SERVER
_POSIX_SYNCHRONIZED_IO
_POSIX_THREAD_ATTR_STACKADDR
_POSIX_THREAD_CPUTIME
_POSIX_THREAD_ATTR_STACKSIZE
_POSIX_THREAD_PRIO_INHERIT
_POSIX_THREAD_PRIO_PROTECT
_POSIX_THREAD_PRIORITY_SCHEDULING
_POSIX_THREAD_PROCESS_SHARED
_POSIX_THREAD_SPORADIC_SERVER
_POSIX_TRACE
_POSIX_TRACE_EVENT_FILTER
_POSIX_TRACE_INHERIT
_POSIX_TRACE_LOG
_POSIX_TYPED_MEMORY_OBJECTS
_XOPEN_CRYPT
_XOPEN_REALTIME
_XOPEN_REALTIME_THREADS
_XOPEN_STREAMS
_XOPEN_UNIX

```

If any of the symbolic constants `_POSIX_TRACE_EVENT_FILTER`, `_POSIX_TRACE_LOG`, or `_POSIX_TRACE_INHERIT` is defined to have a value other than `-1`, then the symbolic constant `_POSIX_TRACE` shall also be defined to have a value other than `-1`.

If the Advisory Information option is supported, there shall be at least one file system that supports the functionality.

2.1.3.2 POSIX Shell and Utilities

The following requirements apply to the shell and utilities:

- The system shall provide all the mandatory utilities in the Shell and Utilities volume of POSIX.1-2008 with all the functional behavior described therein.
- The system shall support the Large File capabilities described in the Shell and Utilities volume of POSIX.1-2008.

- The system may support one or more options (see [Section 2.1.6](#), on page 26) denoted by the following symbolic constants. (The literal names below apply to the *getconf* utility.)

```

        POSIX2_C_DEV
        POSIX2_CHAR_TERM
        POSIX2_FORT_DEV
        POSIX2_FORT_RUN
        POSIX2_LOCALEDEF
        POSIX2_PBS
        POSIX2_PBS_ACCOUNTING
        POSIX2_PBS_LOCATE
        POSIX2_PBS_MESSAGE
        POSIX2_PBS_TRACK
        POSIX2_SW_DEV
        POSIX2_UPE
        XOPEN_UNIX
        XOPEN_UUCP

```

Additional language bindings and development utility options may be provided in other related standards or in a future version of this standard. In the former case, additional symbolic constants of the same general form as shown in this subsection should be defined by the related standard document and made available to the application without requiring POSIX.1-2008 to be updated.

2.1.4 XSI Conformance

XSI This section describes the criteria for implementations providing conformance to the X/Open System Interfaces (XSI) option (see [Section 3.444](#), on page 104). The functionality described in this section shall be provided on implementations that support the XSI option (and the rest of this section is not further shaded).

POSIX.1-2008 describes utilities, functions, and facilities offered to application programs by the X/Open System Interfaces (XSI) option. An XSI-conforming implementation shall meet the criteria for POSIX conformance and the following requirements listed in this section.

XSI-conforming implementations shall set the symbolic constant `_XOPEN_UNIX` to a value other than `-1` and shall set the symbolic constant `_XOPEN_VERSION` to the value 700.

2.1.4.1 XSI System Interfaces

The following requirements apply to the system interfaces when the XSI option is supported:

- The system shall support all the functions and headers defined in POSIX.1-2008 as part of the XSI option denoted by the XSI marking in the SYNOPSIS section, and any extensions marked with the XSI option marking (see [Section 1.7.1](#), on page 7) within the text.
- The system shall support the following options defined within POSIX.1-2008 (see [Section 2.1.6](#), on page 26):

```

        _POSIX_FSYNC
        _POSIX_THREAD_ATTR_STACKADDR
        _POSIX_THREAD_ATTR_STACKSIZE
        _POSIX_THREAD_PROCESS_SHARED

```

- The system may support the following XSI Option Groups (see [Section 2.1.5.2](#), on page 22) defined within POSIX.1-2008:
 - Encryption
 - Realtime
 - Advanced Realtime
 - Realtime Threads
 - Advanced Realtime Threads
 - Tracing
 - XSI STREAMS

2.1.4.2 XSI Shell and Utilities Conformance

The following requirements apply to the shell and utilities when the XSI option is supported:

- The system shall support all the utilities defined in the Shell and Utilities volume of POSIX.1-2008 as part of the XSI option denoted by the XSI marking in the SYNOPSIS section, and any extensions marked with the XSI option marking (see [Section 1.7.1](#), on page 7) within the text.
- The system shall support the User Portability Utilities option and the Terminal Characteristics option.
- The system shall support creation of locales (see [Chapter 7](#), on page 135).
- The C-language Development utility *c99* shall be supported.
- The XSI Development Utilities option may be supported. It consists of the following software development utilities:

<i>admin</i>	<i>delta</i>	<i>rmdel</i>	<i>val</i>
<i>cflow</i>	<i>get</i>	<i>sact</i>	<i>what</i>
<i>ctags</i>	<i>nm</i>	<i>sccs</i>	
<i>cxref</i>	<i>prs</i>	<i>unget</i>	

2.1.5 Option Groups

An Option Group is a group of related functions or options defined within the System Interfaces volume of POSIX.1-2008.

If an implementation supports an Option Group, then the system shall support the functional behavior described herein.

If an implementation does not support an Option Group, then the system need not support the functional behavior described herein.

2.1.5.1 Subprofiling Considerations

Profiling standards supporting functional requirements less than that required in POSIX.1-2008 may subset both mandatory and optional functionality required for POSIX Conformance (see [Section 2.1.3](#), on page 16) or XSI Conformance (see [Section 2.1.4](#), on page 19). Such profiles shall organize the subsets into Subprofiling Option Groups.

XRAT [Appendix E](#) (on page 3741) describes a representative set of such Subprofiling Option Groups for use by profiles applicable to specialized realtime systems. POSIX.1-2008 does not require that the presence of Subprofiling Option Groups be testable at compile-time (as symbols defined in any header) or at runtime (via *sysconf()* or *getconf()*).

A Subprofiling Option Group may provide basic system functionality that other Subprofiling Option Groups and other options depend upon.³ If a profile of POSIX.1-2008 does not require an implementation to provide a Subprofiling Option Group that provides features utilized by a required Subprofiling Option Group (or option),⁴ the profile shall specify⁵ all of the following:

- Restricted or altered behavior of interfaces defined in POSIX.1-2008 that may differ on an implementation of the profile
- Additional behaviors that may produce undefined or unspecified results
- Additional implementation-defined behavior that implementations shall be required to document in the profile's conformance document

if any of the above is a result of the profile not requiring an interface required by POSIX.1-2008.

The following additional rules shall apply to all profiles of POSIX.1-2008:

- Any application that conforms to that profile shall also conform to POSIX.1-2008, unless the application depends on the definition of a profile support indicator macro in **<unistd.h>** (that is, a profile shall not require restricted, altered, or extended behaviors of an implementation of POSIX.1-2008).
- Profiles are permitted to require the definition of a *profile support indicator macro* with a name beginning **_POSIX_AEP_** in **<unistd.h>**.
- Profiles shall require the definition of the macro **_POSIX_SUBPROFILE** in **<unistd.h>** on implementations that do not meet all of the requirements of a POSIX.1-conforming implementation.
- Profiles are permitted to add additional requirements to the limits defined in **<limits.h>** and **<stdint.h>**, subject to the following:

For the limits in **<limits.h>** and **<stdint.h>**:

- If the limit is specified as having a fixed value, it shall not be changed by a profile.
- If a limit is specified as having a minimum or maximum acceptable value, it may be changed by a profile as follows:
 - A profile may increase a minimum acceptable value, but shall not make a minimum acceptable value smaller.

3. As an example, the File System profiling option group provides underlying support for pathname resolution and file creation which are needed by any interface in POSIX.1-2008 that parses a *path* argument. If a profile requires support for the Device Input and Output profiling option group but does not require support for the File System profiling option group, the profile must specify how pathname resolution is to behave in that profile, how the **O_CREAT** flag to *open()* is to be handled (and the use of the character 'a' in the *mode* argument of *fopen()* when a pathname argument names a file that does not exist), and specify lots of other details.

4. As an example, POSIX.1-2008 requires that implementations claiming to support the Range Memory Locking option also support the Process Memory Locking option. A profile could require that the Range Memory Locking option had to be supplied without requiring that the Process Memory Locking option be supplied as long as the profile specifies everything an application developer or system implementor would have to know to build an application or implementation conforming to the profile.

5. Note that the profile could just specify that any use of the features not specified by the profile would produce undefined or unspecified results.

— A profile may reduce a maximum acceptable value, but shall not make a maximum acceptable value larger.

- A profile shall not change a limit specified as having a minimum or maximum value into a limit specified as having a fixed value.
- A profile shall not create new limits.
- Any implementation that conforms to POSIX.1-2008 (including all options and extended limits required by the profile) shall also conform to that profile, except for the possible omission from `<unistd.h>` of a profile support indicator macro required by the profile.

2.1.5.2 XSI Option Groups

XSI This section describes Option Groups to support the definition of XSI conformance within the System Interfaces volume of POSIX.1-2008. The functionality described in this section shall be provided on implementations that support the XSI option and the appropriate Option Group (and the rest of this section is not further shaded).

The following Option Groups are defined.

Encryption

The Encryption Option Group is denoted by the symbolic constant `_XOPEN_CRYPT`. It includes the following functions:

`crypt()`, `encrypt()`, `setkey()`

These functions are marked CRYPT.

Due to export restrictions on the decoding algorithm in some countries, implementations may be restricted in making these functions available. All the functions in the Encryption Option Group may therefore return `[ENOSYS]` or, alternatively, `encrypt()` shall return `[ENOSYS]` for the decryption operation.

An implementation that claims conformance to this Option Group shall set `_XOPEN_CRYPT` to a value other than `-1`.

Realtime

The Realtime Option Group is denoted by the symbolic constant `_XOPEN_REALTIME`.

This Option Group includes a set of realtime functions drawn from options within POSIX.1-2008 (see [Section 2.1.6](#), on page 26).

Where entire functions are included in the Option Group, the NAME section is marked with REALTIME. Where additional semantics have been added to existing pages, the new material is identified by use of the appropriate margin legend for the underlying option defined within POSIX.1-2008.

An implementation that claims conformance to this Option Group shall set `_XOPEN_REALTIME` to a value other than `-1`.

This Option Group consists of the set of the following options from within POSIX.1-2008 (see [Section 2.1.6](#), on page 26):

```

774     _POSIX_FSYNC
775     _POSIX_MEMLOCK
776     _POSIX_MEMLOCK_RANGE
777     _POSIX_MESSAGE_PASSING
778     _POSIX_PRIORITIZED_IO
779     _POSIX_PRIORITY_SCHEDULING
780     _POSIX_SHARED_MEMORY_OBJECTS
781     _POSIX_SYNCHRONIZED_IO

```

782 If the symbolic constant `_XOPEN_REALTIME` is defined to have a value other than `-1`, then the
 783 following symbolic constants shall be defined by the implementation to have the value 200809L:

```

784     _POSIX_MEMLOCK
785     _POSIX_MEMLOCK_RANGE
786     _POSIX_MESSAGE_PASSING
787     _POSIX_PRIORITY_SCHEDULING
788     _POSIX_SHARED_MEMORY_OBJECTS
789     _POSIX_SYNCHRONIZED_IO

```

790 The functionality associated with `_POSIX_FSYNC` shall always be supported on XSI-conformant
 791 systems.

792 Support of `_POSIX_PRIORITIZED_IO` on XSI-conformant systems is optional. If
 793 `_POSIX_PRIORITIZED_IO` is supported, then asynchronous I/O operations performed by
 794 `aio_read()`, `aio_write()`, and `lio_listio()` shall be submitted at a priority equal to the scheduling
 795 priority equal to a base scheduling priority minus `aioctx->aio_reqprio`. If Thread Execution
 796 Scheduling is not supported, then the base scheduling priority is that of the calling process;
 797 otherwise, the base scheduling priority is that of the calling thread. The implementation shall
 798 also document for which files I/O prioritization is supported.

799 **Advanced Realtime**

800 An implementation that claims conformance to this Option Group shall also support the
 801 Realtime Option Group.

802 Where entire functions are included in the Option Group, the NAME section is marked with
 803 **ADVANCED REALTIME**. Where additional semantics have been added to existing pages, the
 804 new material is identified by use of the appropriate margin legend for the underlying option
 805 defined within POSIX.1-2008.

806 This Option Group consists of the set of the following options from within POSIX.1-2008 (see
 807 [Section 2.1.6](#), on page 26):

```

808     _POSIX_ADVISORY_INFO
809     _POSIX_CPUTIME
810     _POSIX_MONOTONIC_CLOCK
811     _POSIX_SPAWN
812     _POSIX_SPORADIC_SERVER
813     _POSIX_TYPED_MEMORY_OBJECTS

```

814 If the implementation supports the Advanced Realtime Option Group, then the following
 815 symbolic constants shall be defined by the implementation to have the value 200809L:

816 _POSIX_ADVISORY_INFO
 817 _POSIX_CPUTIME
 818 _POSIX_MONOTONIC_CLOCK
 819 _POSIX_SPAWN
 820 _POSIX_SPORADIC_SERVER
 821 _POSIX_TYPED_MEMORY_OBJECTS

822 If the symbolic constant _POSIX_SPORADIC_SERVER is defined, then the symbolic constant
 823 _POSIX_PRIORITY_SCHEDULING shall also be defined by the implementation to have the
 824 value 200809L.

825 **Realtime Threads**

826 The Realtime Threads Option Group is denoted by the symbolic constant
 827 _XOPEN_REALTIME_THREADS.

828 This Option Group consists of the set of the following options from within POSIX.1-2008 (see
 829 [Section 2.1.6](#), on page 26):

830 _POSIX_THREAD_PRIO_INHERIT
 831 _POSIX_THREAD_PRIO_PROTECT
 832 _POSIX_THREAD_PRIORITY_SCHEDULING
 833 _POSIX_THREAD_ROBUST_PRIO_INHERIT +
 834 _POSIX_THREAD_ROBUST_PRIO_PROTECT +

835 Where applicable, whole pages are marked REALTIME THREADS, together with the
 836 appropriate option margin legend for the SYNOPSIS section (see [Section 1.7.1](#), on page 7).

837 An implementation that claims conformance to this Option Group shall set
 838 _XOPEN_REALTIME_THREADS to a value other than -1.

839 If the symbol _XOPEN_REALTIME_THREADS is defined to have a value other than -1, then the
 840 following options shall also be defined by the implementation to have the value 200809L:

841 _POSIX_THREAD_PRIO_INHERIT
 842 _POSIX_THREAD_PRIO_PROTECT
 843 _POSIX_THREAD_PRIORITY_SCHEDULING
 844 _POSIX_THREAD_ROBUST_PRIO_INHERIT +
 845 _POSIX_THREAD_ROBUST_PRIO_PROTECT +

846 **Advanced Realtime Threads**

847 An implementation that claims conformance to this Option Group shall also support the
 848 Realtime Threads Option Group.

849 Where entire functions are included in the Option Group, the NAME section is marked with
 850 ADVANCED REALTIME THREADS. Where additional semantics have been added to existing
 851 pages, the new material is identified by use of the appropriate margin legend for the underlying
 852 option defined within POSIX.1-2008.

853 This Option Group consists of the set of the following options from within POSIX.1-2008 (see
 854 [Section 2.1.6](#), on page 26):

855 _POSIX_THREAD_CPUTIME
 856 _POSIX_THREAD_SPORADIC_SERVER

857 If the symbolic constant _POSIX_THREAD_SPORADIC_SERVER is defined to have the value

200809L, then the symbolic constant `_POSIX_THREAD_PRIORITY_SCHEDULING` shall also be defined by the implementation to have the value 200809L.

If the implementation supports the Advanced Realtime Threads Option Group, then the following symbolic constants shall be defined by the implementation to have the value 200809L:

```
_POSIX_THREAD_CPUTIME
_POSIX_THREAD_SPORADIC_SERVER
```

Tracing

This Option Group includes a set of tracing functions drawn from options within POSIX.1-2008 (see [Section 2.1.6](#), on page 26).

Where entire functions are included in the Option Group, the NAME section is marked with TRACING. Where additional semantics have been added to existing pages, the new material is identified by use of the appropriate margin legend for the underlying option defined within POSIX.1-2008.

This Option Group consists of the set of the following options from within POSIX.1-2008 (see [Section 2.1.6](#), on page 26):

```
_POSIX_TRACE
_POSIX_TRACE_EVENT_FILTER
_POSIX_TRACE_LOG
_POSIX_TRACE_INHERIT
```

If the implementation supports the Tracing Option Group, then the following symbolic constants shall be defined by the implementation to have the value 200809L:

```
_POSIX_TRACE
_POSIX_TRACE_EVENT_FILTER
_POSIX_TRACE_LOG
_POSIX_TRACE_INHERIT
```

XSI STREAMS

OB XSR This section describes the XSI STREAMS Option Group, denoted by the symbolic constant `_XOPEN_STREAMS`. The functionality described in this section shall be provided on implementations that support the XSI STREAMS option (and the rest of this section is not further shaded).

This Option Group includes functionality related to STREAMS, a uniform mechanism for implementing networking services and other character-based I/O as described in XSH [Section 2.6](#) (on page 499).

It includes the following functions:

```
fattach()      ioctl()
fdetach()      isastream()
getmsg()       putmsg()
getpmsg()      putpmsg()
```

and the `<stropts.h>` header.

Where applicable, whole pages are marked STREAMS, together with the appropriate option margin legend for the SYNOPSIS section (see [Section 1.7.1](#), on page 7). Where additional semantics have been added to existing pages, the new material is identified by use of the

appropriate margin legend for the underlying option defined within POSIX.1-2008.

An implementation that claims conformance to this Option Group shall set `_XOPEN_STREAMS` to a value other than `-1`.

2.1.6 Options

The symbolic constants defined in `<unistd.h>`, [Constants for Options and Option Groups](#) (on page 433) reflect implementation options for POSIX.1-2008. These symbols can be used by the application to determine which of three categories of support for optional facilities are provided by the implementation.

1. Option not supported for compilation.

The implementation advertises at compile time (by defining the constant in `<unistd.h>` with value `-1`, or by leaving it undefined) that the option is not supported for compilation and, at the time of compilation, is not supported for runtime use. In this case, the headers, data types, function interfaces, and utilities required only for the option need not be present. A later runtime check using the `fpathconf()`, `pathconf()`, or `sysconf` functions defined in the System Interfaces volume of POSIX.1-2008 or the `getconf` utility defined in the Shell and Utilities volume of POSIX.1-2008 can in some circumstances indicate that the option is supported at runtime. (For example, an old application binary might be run on a newer implementation to which support for the option has been added.)

2. Option always supported.

The implementation advertises at compile time (by defining the constant in `<unistd.h>` with a value greater than zero) that the option is supported both for compilation and for use at runtime. In this case, all headers, data types, function interfaces, and utilities required only for the option shall be available and shall operate as specified. Runtime checks with `fpathconf()`, `pathconf()`, or `sysconf` shall indicate that the option is supported.

3. Option might or might not be supported at runtime.

The implementation advertises at compile time (by defining the constant in `<unistd.h>` with value zero) that the option is supported for compilation and might or might not be supported at runtime. In this case, the `fpathconf()`, `pathconf()`, or `sysconf()` functions defined in the System Interfaces volume of POSIX.1-2008 or the `getconf` utility defined in the Shell and Utilities volume of POSIX.1-2008 can be used to retrieve the value of each symbol on each specific implementation to determine whether the option is supported at runtime. All headers, data types, and function interfaces required to compile and execute applications which use the option at runtime (after checking at runtime that the option is supported) shall be provided, but if the option is not supported at runtime they need not operate as specified. Utilities or other facilities required only for the option, but not needed to compile and execute such applications, need not be present.

If an option is not supported for compilation, an application that attempts to use anything associated only with the option is considered to be requiring an extension. Unless explicitly specified otherwise, the behavior of functions associated with an option that is not supported at runtime is unspecified, and an application that uses such functions without first checking `fpathconf()`, `pathconf()`, or `sysconf` is considered to be requiring an extension.

Margin codes are defined for each option (see [Section 1.7.1](#), on page 7).

942	2.1.6.1	<i>System Interfaces</i>
943		Refer to <unistd.h> , Constants for Options and Option Groups (on page 433) for the list of
944		options.
945	2.1.6.2	<i>Shell and Utilities</i>
946		Each of these symbols shall be considered valid names by the implementation. Refer to
947		<unistd.h> , Constants for Options and Option Groups (on page 433).
948		The literal names shown below apply only to the <i>getconf</i> utility.
949	CD	POSIX2_C_DEV
950		The system supports the C-Language Development Utilities option.
951		The utilities in the C-Language Development Utilities option are used for the development
952		of C-language applications, including compilation or translation of C source code and
953		complex program generators for simple lexical tasks and processing of context-free
954		grammars.
955		The utilities listed below may be provided by a conforming system; however, any system
956		claiming conformance to the C-Language Development Utilities option shall provide all of
957		the utilities listed.
958		<i>c99</i>
959		<i>lex</i>
960		<i>yacc</i>
961		POSIX2_CHAR_TERM
962		The system supports the Terminal Characteristics option. This value need not be present on
963		a system not supporting the User Portability Utilities option.
964		Where applicable, the dependency is noted within the description of the utility.
965		This option applies only to systems supporting the User Portability Utilities option. If
966		supported, then the system supports at least one terminal type capable of all operations
967		described in POSIX.1-2008; see Section 10.2 (on page 198).
968	FD	POSIX2_FORT_DEV
969		The system supports the FORTRAN Development Utilities option.
970		The <i>fort77</i> FORTRAN compiler is the only utility in the FORTRAN Development Utilities
971		option. This is used for the development of FORTRAN language applications, including
972		compilation or translation of FORTRAN source code.
973		The <i>fort77</i> utility may be provided by a conforming system; however, any system claiming
974		conformance to the FORTRAN Development Utilities option shall provide the <i>fort77</i> utility.
975	FR	POSIX2_FORT_RUN
976		The system supports the FORTRAN Runtime Utilities option.
977		The <i>asa</i> utility is the only utility in the FORTRAN Runtime Utilities option.
978		The <i>asa</i> utility may be provided by a conforming system; however, any system claiming
979		conformance to the FORTRAN Runtime Utilities option shall provide the <i>asa</i> utility.
980		POSIX2_LOCALEDEF
981		The system supports the Locale Creation Utilities option.
982		If supported, the system supports the creation of locales as described in the <i>localedef</i> utility.

983 The *localedef* utility may be provided by a conforming system; however, any system
 984 claiming conformance to the Locale Creation Utilities option shall provide the *localedef*
 985 utility.

986 OB BE POSIX2_PBS

987 The system supports the Batch Environment Services and Utilities option (see XCU [Chapter](#)
 988 [3](#), on page 2397).

989 **Note:** The Batch Environment Services and Utilities option is a combination of mandatory and
 990 optional batch services and utilities. The POSIX2_PBS symbolic constant implies the system
 991 supports all the mandatory batch services and utilities.

992 POSIX2_PBS_ACCOUNTING

993 The system supports the Batch Accounting option.

994 POSIX2_PBS_CHECKPOINT

995 The system supports the Batch Checkpoint/Restart option.

996 POSIX2_PBS_LOCATE

997 The system supports the Locate Batch Job Request option.

998 POSIX2_PBS_MESSAGE

999 The system supports the Batch Job Message Request option.

1000 POSIX2_PBS_TRACK

1001 The system supports the Track Batch Job Request option.

1002 SD POSIX2_SW_DEV

1003 The system supports the Software Development Utilities option.

1004 The utilities in the Software Development Utilities option are used for the development of
 1005 applications, including compilation or translation of source code, the creation and
 1006 maintenance of library archives, and the maintenance of groups of inter-dependent
 1007 programs.

1008 The utilities listed below may be provided by the conforming system; however, any system
 1009 claiming conformance to the Software Development Utilities option shall provide all of the
 1010 utilities listed here.

1011 *ar*
 1012 *make*
 1013 *nm*
 1014 *strip*

1015 UP POSIX2_UPE

1016 The system supports the User Portability Utilities option.

1017 The utilities in the User Portability Utilities option shall be implemented on all systems that
 1018 claim conformance to this option, except for the *vi* utility which is noted as having features
 1019 that cannot be implemented on all terminal types; if the POSIX2_CHAR_TERM option is
 1020 supported, the system shall support all such features on at least one terminal type; see
 1021 [Section 10.2](#) (on page 198).

1022 The list of utilities in the User Portability Utilities option is as follows:

1023 *bg* *fc* *jobs* *talk*
 1024 *ex* *fg* *more* *vi*

1025	XSI	XOPEN_UNIX
1026		The system supports the X/Open System Interfaces (XSI) option (see Section 2.1.4 , on page 19).
1027		
1028	UU	XOPEN_UUCP
1029		The system supports the UUCP Utilities option.
1030		The list of utilities in the UUCP Utilities option is as follows:
1031		<i>uucp</i>
1032		<i>uustat</i>
1033		<i>uux</i>

1034 2.2 Application Conformance

1035 For the purposes of POSIX.1-2008, the application conformance requirements given in this
1036 section apply.

1037 All applications claiming conformance to POSIX.1-2008 shall use only language-dependent
1038 services for the C programming language described in [Section 2.3](#) (on page 31), shall use only
1039 the utilities and facilities defined in the Shell and Utilities volume of POSIX.1-2008, and shall fall
1040 within one of the following categories.

1041 2.2.1 Strictly Conforming POSIX Application

1042 A Strictly Conforming POSIX Application is an application that requires only the facilities
1043 described in POSIX.1-2008. Such an application:

- 1044 1. Shall accept any implementation behavior that results from actions it takes in areas
1045 described in POSIX.1-2008 as *implementation-defined* or *unspecified*, or where POSIX.1-2008
1046 indicates that implementations may vary
- 1047 2. Shall not perform any actions that are described as producing *undefined* results
- 1048 3. For symbolic constants, shall accept any value in the range permitted by POSIX.1-2008,
1049 but shall not rely on any value in the range being greater than the minimums listed or
1050 being less than the maximums listed in POSIX.1-2008
- 1051 4. Shall not use facilities designated as *obsolescent*
- 1052 5. Is required to tolerate and permitted to adapt to the presence or absence of optional
1053 facilities whose availability is indicated by [Section 2.1.3](#) (on page 16)
- 1054 6. For the C programming language, shall not produce any output dependent on any
1055 behavior described in the ISO/IEC 9899:1999 standard as *unspecified*, *undefined*, or
1056 *implementation-defined*, unless the System Interfaces volume of POSIX.1-2008 specifies the
1057 behavior
- 1058 7. For the C programming language, shall not exceed any minimum implementation limit
1059 defined in the ISO/IEC 9899:1999 standard, unless the System Interfaces volume of
1060 POSIX.1-2008 specifies a higher minimum implementation limit
- 1061 8. For the C programming language, shall define `_POSIX_C_SOURCE` to be 200809L before
1062 any header is included

1063 Within POSIX.1-2008, any restrictions placed upon a Conforming POSIX Application shall

restrict a Strictly Conforming POSIX Application.

2.2.2 Conforming POSIX Application

2.2.2.1 ISO/IEC Conforming POSIX Application

An ISO/IEC Conforming POSIX Application is an application that uses only the facilities described in POSIX.1-2008 and approved Conforming Language bindings for any ISO or IEC standard. Such an application shall include a statement of conformance that documents all options and limit dependencies, and all other ISO or IEC standards used.

2.2.2.2 <National Body> Conforming POSIX Application

A <National Body> Conforming POSIX Application differs from an ISO/IEC Conforming POSIX Application in that it also may use specific standards of a single ISO/IEC member body referred to here as <National Body>. Such an application shall include a statement of conformance that documents all options and limit dependencies, and all other <National Body> standards used.

2.2.3 Conforming POSIX Application Using Extensions

A Conforming POSIX Application Using Extensions is an application that differs from a Conforming POSIX Application only in that it uses non-standard facilities that are consistent with POSIX.1-2008. Such an application shall fully document its requirements for these extended facilities, in addition to the documentation required of a Conforming POSIX Application. A Conforming POSIX Application Using Extensions shall be either an ISO/IEC Conforming POSIX Application Using Extensions or a <National Body> Conforming POSIX Application Using Extensions (see [Section 2.2.2.1](#) and [Section 2.2.2.2](#)).

2.2.4 Strictly Conforming XSI Application

A Strictly Conforming XSI Application is an application that requires only the facilities described in POSIX.1-2008. Such an application:

1. Shall accept any implementation behavior that results from actions it takes in areas described in POSIX.1-2008 as *implementation-defined* or *unspecified*, or where POSIX.1-2008 indicates that implementations may vary
2. Shall not perform any actions that are described as producing *undefined* results
3. For symbolic constants, shall accept any value in the range permitted by POSIX.1-2008, but shall not rely on any value in the range being greater than the minimums listed or being less than the maximums listed in POSIX.1-2008
4. Shall not use facilities designated as *obsolescent*
5. Is required to tolerate and permitted to adapt to the presence or absence of optional facilities whose availability is indicated by [Section 2.1.4](#) (on page 19)

6. For the C programming language, shall not produce any output dependent on any behavior described in the ISO C standard as *unspecified*, *undefined*, or *implementation-defined*, unless the System Interfaces volume of POSIX.1-2008 specifies the behavior
7. For the C programming language, shall not exceed any minimum implementation limit defined in the ISO C standard, unless the System Interfaces volume of POSIX.1-2008 specifies a higher minimum implementation limit
8. For the C programming language, shall define `_XOPEN_SOURCE` to be 700 before any header is included

Within POSIX.1-2008, any restrictions placed upon a Conforming POSIX Application shall restrict a Strictly Conforming XSI Application.

2.2.5 Conforming XSI Application Using Extensions

A Conforming XSI Application Using Extensions is an application that differs from a Strictly Conforming XSI Application only in that it uses non-standard facilities that are consistent with POSIX.1-2008. Such an application shall fully document its requirements for these extended facilities, in addition to the documentation required of a Strictly Conforming XSI Application.

2.3 Language-Dependent Services for the C Programming Language

Implementors seeking to claim conformance using the ISO C standard shall claim POSIX conformance as described in [Section 2.1.3](#) (on page 16).

2.4 Other Language-Related Specifications

POSIX.1-2008 is currently specified in terms of the shell command language and ISO C. Bindings to other programming languages are being developed.

If conformance to POSIX.1-2008 is claimed for implementation of any programming language, the implementation of that language shall support the use of external symbols distinct to at least 31 bytes in length in the source program text. (That is, identifiers that differ at or before the thirty-first byte shall be distinct.) If a national or international standard governing a language defines a maximum length that is less than this value, the language-defined maximum shall be supported. External symbols that differ only by case shall be distinct when the character set in use distinguishes uppercase and lowercase characters and the language permits (or requires) uppercase and lowercase characters to be distinct in external symbols.

1127

1128

Definitions

1129 For the purposes of POSIX.1-2008, the following terms and definitions apply. The Authoritative
1130 Dictionary of IEEE Standards Terms, Seventh Edition should be referenced for terms not defined
1131 in this section.

1132 **Note:** No shading to denote extensions or options occurs in this chapter. Where the terms and
1133 definitions given in this chapter are used elsewhere in text related to extensions and options,
1134 they are shaded as appropriate.

1135 3.1 Abortive Release

1136 An abrupt termination of a network connection that may result in the loss of data.

1137 3.2 Absolute Pathname

1138 A pathname beginning with a single or more than two <slash> characters; see also [Section 3.267](#)
1139 (on page 75).

1140 **Note:** Pathname Resolution is defined in detail in [Section 4.12](#) (on page 111).

1141 3.3 Access Mode

1142 A particular form of access permitted to a file.

1143 3.4 Additional File Access Control Mechanism

1144 An implementation-defined mechanism that is layered upon the access control mechanisms
1145 defined here, but which do not grant permissions beyond those defined herein, although they
1146 may further restrict them.

1147 **Note:** File Access Permissions are defined in detail in [Section 4.4](#) (on page 108).

1148 3.5 Address Space

1149 The memory locations that can be referenced by a process or the threads of a process.

1150 3.6 Advisory Information

1151 An interface that advises the implementation on (portable) application behavior so that it can
1152 optimize the system.

1153 3.7 Affirmative Response

1154 An input string that matches one of the responses acceptable to the *LC_MESSAGES* category
1155 keyword **yesexpr**, matching an extended regular expression in the current locale.

1156 **Note:** The *LC_MESSAGES* category is defined in detail in [Section 7.3.6](#) (on page 164).

1157 3.8 Alert

1158 To cause the user's terminal to give some audible or visual indication that an error or some other
1159 event has occurred. When the standard output is directed to a terminal device, the method for
1160 alerting the terminal user is unspecified. When the standard output is not directed to a terminal
1161 device, the alert is accomplished by writing the alert to standard output (unless the utility
1162 description indicates that the use of standard output produces undefined results in this case).

1163 3.9 Alert Character (<alert>)

1164 A character that in the output stream should cause a terminal to alert its user via a visual or
1165 audible notification. It is the character designated by '`\a`' in the C language. It is unspecified
1166 whether this character is the exact sequence transmitted to an output device by the system to
1167 accomplish the alert function.

1168 3.10 Alias Name

1169 In the shell command language, a word consisting solely of underscores, digits, and alphabets
1170 from the portable character set and any of the following characters: '`!`', '`%`', '`'`', '`,`', '`@`'.

1171 Implementations may allow other characters within alias names as an extension.

1172 **Note:** The Portable Character Set is defined in detail in [Section 6.1](#) (on page 125).

3.11 Alignment

A requirement that objects of a particular type be located on storage boundaries with addresses that are particular multiples of a byte address.

Note: See also the ISO C standard, Section B3.

3.12 Alternate File Access Control Mechanism

An implementation-defined mechanism that is independent of the access control mechanisms defined herein, and which if enabled on a file may either restrict or extend the permissions of a given user. POSIX.1-2008 defines when such mechanisms can be enabled and when they are disabled.

Note: File Access Permissions are defined in detail in [Section 4.4](#) (on page 108).

3.13 Alternate Signal Stack

Memory associated with a thread, established upon request by the implementation for a thread, separate from the thread signal stack, in which signal handlers responding to signals sent to that thread may be executed.

3.14 Ancillary Data

Protocol-specific, local system-specific, or optional information. The information can be both local or end-to-end significant, header information, part of a data portion, protocol-specific, and implementation or system-specific.

3.15 Angle Brackets

The characters '`<`' (left-angle-bracket) and '`>`' (right-angle-bracket). When used in the phrase “enclosed in angle brackets”, the symbol '`<`' immediately precedes the object to be enclosed, and '`>`' immediately follows it. When describing these characters in the portable character set, the names `<less-than-sign>` and `<greater-than-sign>` are used.

3.16 Apostrophe Character (<apostrophe>)

The character designated by '`\'`' in the C language, also known as the single-quote character.

3.17 Application

A computer program that performs some desired function.

3.18 Application Address

Endpoint address of a specific application.

3.19 Application Program Interface (API)

The definition of syntax and semantics for providing computer system services.

3.20 Appropriate Privileges

An implementation-defined means of associating privileges with a process with regard to the function calls, function call options, and the commands that need special privileges. There may be zero or more such means. These means (or lack thereof) are described in the conformance document.

Note: Function calls are defined in the System Interfaces volume of POSIX.1-2008, and commands are defined in the Shell and Utilities volume of POSIX.1-2008.

3.21 Argument

In the shell command language, a parameter passed to a utility as the equivalent of a single string in the *argv* array created by one of the *exec* functions. An argument is one of the options, option-arguments, or operands following the command name.

Note: The Utility Argument Syntax is defined in detail in [Section 12.1](#) (on page 213) and XCU [Section 2.9.1.1](#) (on page 2339).

In the C language, an expression in a function call expression or a sequence of preprocessing tokens in a function-like macro invocation.

3.22 Arm (a Timer)

To start a timer measuring the passage of time, enabling notifying a process when the specified time or time interval has passed.

3.23 Asterisk Character (<asterisk>)

The character ' * '.

3.24 Async-Cancel-Safe Function

A function that may be safely invoked by an application while the asynchronous form of cancellation is enabled. No function is async-cancel-safe unless explicitly described as such.

1227 **3.25 Asynchronous Events**

1228 Events that occur independently of the execution of the application.

1229 **3.26 Asynchronous Input and Output**

1230 A functionality enhancement to allow an application process to queue data input and output
1231 commands with asynchronous notification of completion.

1232 **3.27 Async-Signal-Safe Function**

1233 A function that may be invoked, without restriction, from signal-catching functions. No function
1234 is async-signal-safe unless explicitly described as such.

1235 **3.28 Asynchronously-Generated Signal**

1236 A signal that is not attributable to a specific thread. Examples are signals sent via *kill()*, signals
1237 sent from the keyboard, and signals delivered to process groups. Being asynchronous is a
1238 property of how the signal was generated and not a property of the signal number. All signals
1239 may be generated asynchronously.

1240 **Note:** The *kill()* function is defined in detail in the System Interfaces volume of POSIX.1-2008.

1241 **3.29 Asynchronous I/O Completion**

1242 For an asynchronous read or write operation, when a corresponding synchronous read or write
1243 would have completed and when any associated status fields have been updated.

1244 **3.30 Asynchronous I/O Operation**

1245 An I/O operation that does not of itself cause the thread requesting the I/O to be blocked from
1246 further use of the processor.

1247 This implies that the process and the I/O operation may be running concurrently.

1248 **3.31 Authentication**

1249 The process of validating a user or process to verify that the user or process is not a counterfeit.

3.32 Authorization

The process of verifying that a user or process has permission to use a resource in the manner requested.

To ensure security, the user or process would also need to be authenticated before granting access.

3.33 Background Job

See *Background Process Group* in [Section 3.35](#).

3.34 Background Process

A process that is a member of a background process group.

3.35 Background Process Group (or Background Job)

Any process group, other than a foreground process group, that is a member of a session that has established a connection with a controlling terminal.

3.36 Backquote Character

The character ' ` ', also known as <grave-accent>.

3.37 Backslash Character (<backslash>)

The character designated by ' \ ' in the C language, also known as reverse solidus.

3.38 Backspace Character (<backspace>)

A character that, in the output stream, should cause printing (or displaying) to occur one column position previous to the position about to be printed. If the position about to be printed is at the beginning of the current line, the behavior is unspecified. It is the character designated by ' \b ' in the C language. It is unspecified whether this character is the exact sequence transmitted to an output device by the system to accomplish the backspace function. The backspace defined here is not necessarily the ERASE special character.

Note: Special Characters are defined in detail in [Section 11.1.9](#) (on page 204).

1274 **3.39 Barrier**

1275 A synchronization object that allows multiple threads to synchronize at a particular point in
1276 their execution.

1277 **3.40 Basename**

1278 The final, or only, filename in a pathname.

1279 **3.41 Basic Regular Expression (BRE)**

1280 A regular expression (see [Section 3.317](#), on page 84) used by the majority of utilities that select
1281 strings from a set of character strings.

1282 **Note:** Basic Regular Expressions are described in detail in [Section 9.3](#) (on page 183).

1283 **3.42 Batch Access List**

1284 A list of user IDs and group IDs of those users and groups authorized to place batch jobs in a
1285 batch queue.

1286 A batch access list is associated with a batch queue. A batch server uses the batch access list of a
1287 batch queue as one of the criteria in deciding to put a batch job in a batch queue.

1288 **3.43 Batch Administrator**

1289 A user that is authorized to modify all the attributes of queues and jobs and to change the status
1290 of a batch server.

1291 **3.44 Batch Client**

1292 A computational entity that utilizes batch services by making requests of batch servers.

1293 Batch clients often provide the means by which users access batch services, although a batch
1294 server may act as a batch client by virtue of making requests of another batch server.

3.45 Batch Destination

The batch server in a batch system to which a batch job should be sent for processing.

Acceptance of a batch job at a batch destination is the responsibility of a receiving batch server. A batch destination may consist of a batch server-specific portion, a network-wide portion, or both. The batch server-specific portion is referred to as the “batch queue”. The network-wide portion is referred to as a “batch server name”.

3.46 Batch Destination Identifier

A string that identifies a specific batch destination.

A string of characters in the portable character set used to specify a particular batch destination.

Note: The Portable Character Set is defined in detail in [Section 6.1](#) (on page 125).

3.47 Batch Directive

A line from a file that is interpreted by the batch server. The line is usually in the form of a comment and is an additional means of passing options to the *qsub* utility.

Note: The *qsub* utility is defined in detail in the Shell and Utilities volume of POSIX.1-2008.

3.48 Batch Job

A set of computational tasks for a computing system.

Batch jobs are managed by batch servers.

Once created, a batch job may be executing or pending execution. A batch job that is executing has an associated session leader (a process) that initiates and monitors the computational tasks of the batch job.

3.49 Batch Job Attribute

A named data type whose value affects the processing of a batch job.

The values of the attributes of a batch job affect the processing of that job by the batch server that manages the batch job.

3.50 Batch Job Identifier

A unique name for a batch job. A name that is unique among all other batch job identifiers in a batch system and that identifies the batch server to which the batch job was originally submitted.

1323 **3.51 Batch Job Name**

1324 A label that is an attribute of a batch job. The batch job name is not necessarily unique.

1325 **3.52 Batch Job Owner**

1326 The *username@hostname* of the user submitting the batch job, where *username* is a user name (see
1327 also [Section 3.431](#), on page 102) and *hostname* is a network host name.

1328 **3.53 Batch Job Priority**

1329 A value specified by the user that may be used by an implementation to determine the order in
1330 which batch jobs are selected to be executed. Job priority has a numeric value in the range
1331 -1 024 to 1 023.

1332 **Note:** The batch job priority is not the execution priority (nice value) of the batch job.

1333 **3.54 Batch Job State**

1334 An attribute of a batch job which determines the types of requests that the batch server that
1335 manages the batch job can accept for the batch job. Valid states include QUEUED, RUNNING,
1336 HELD, WAITING, EXITING, and TRANSITING.

1337 **3.55 Batch Name Service**

1338 A service that assigns batch names that are unique within the batch name space, and that can
1339 translate a unique batch name into the location of the named batch entity.

1340 **3.56 Batch Name Space**

1341 The environment within which a batch name is known to be unique.

3.57 Batch Node

A host containing part or all of a batch system.

A batch node is a host meeting at least one of the following conditions:

- Capable of executing a batch client
- Contains a routing batch queue
- Contains an execution batch queue

3.58 Batch Operator

A user that is authorized to modify some, but not all, of the attributes of jobs and queues, and may change the status of the batch server.

3.59 Batch Queue

A manageable object that represents a set of batch jobs and is managed by a single batch server.

Note: A set of batch jobs is called a batch queue largely for historical reasons. Jobs are selected from the batch queue for execution based on attributes such as priority, resource requirements, and hold conditions.

See also XCU [Section 3.1.2](#) (on page 2398).

3.60 Batch Queue Attribute

A named data type whose value affects the processing of all batch jobs that are members of the batch queue.

A batch queue has attributes that affect the processing of batch jobs that are members of the batch queue.

3.61 Batch Queue Position

The place, relative to other jobs in the batch queue, occupied by a particular job in a batch queue. This is defined in part by submission time and priority; see also [Section 3.62](#).

3.62 Batch Queue Priority

The maximum job priority allowed for any batch job in a given batch queue.

The batch queue priority is set and may be changed by users with appropriate privileges. The priority is bounded in an implementation-defined manner.

3.63 Batch Rerunability

An attribute of a batch job indicating that it may be rerun after an abnormal termination from the beginning without affecting the validity of the results.

3.64 Batch Restart

The action of resuming the processing of a batch job from the point of the last checkpoint. Typically, this is done if the batch job has been interrupted because of a system failure.

3.65 Batch Server

A computational entity that provides batch services.

3.66 Batch Server Name

A string of characters in the portable character set used to specify a particular server in a network.

Note: The Portable Character Set is defined in detail in [Section 6.1](#) (on page 125).

3.67 Batch Service

Computational and organizational services performed by a batch system on behalf of batch jobs. Batch services are of two types: requested and deferred.

Note: Batch Services are listed in XCU [Table 3-5](#) (on page 2412).

3.68 Batch Service Request

A solicitation of services from a batch client to a batch server.

A batch service request may entail the exchange of any number of messages between the batch client and the batch server.

When naming specific types of service requests, the term “request” is qualified by the type of request, as in *Queue Batch Job Request* and *Delete Batch Job Request*.

3.69 Batch Submission

The process by which a batch client requests that a batch server create a batch job via a *Queue Job Request* to perform a specified computational task.

3.70 Batch System

A collection of one or more batch servers.

3.71 Batch Target User

The name of a user on the batch destination batch server.

The target user is the user name under whose account the batch job is to execute on the destination batch server.

3.72 Batch User

A user who is authorized to make use of batch services.

3.73 Bind

The process of assigning a network address to an endpoint.

3.74 Blank Character (<blank>)

One of the characters that belong to the **blank** character class as defined via the *LC_CTYPE* category in the current locale. In the POSIX locale, a <blank> character is either a <tab> or a <space>.

3.75 Blank Line

A line consisting solely of zero or more <blank> characters terminated by a <newline>; see also [Section 3.145](#) (on page 56).

3.76 Blocked Process (or Thread)

A process (or thread) that is waiting for some condition (other than the availability of a processor) to be satisfied before it can continue execution.

3.77 Blocking

A property of an open file description that causes function calls associated with it to wait for the requested action to be performed before returning.

3.78 Block-Mode Terminal

A terminal device operating in a mode incapable of the character-at-a-time input and output operations described by some of the standard utilities.

Note: Output Devices and Terminal Types are defined in detail in [Section 10.2](#) (on page 198).

3.79 Block Special File

A file that refers to a device. A block special file is normally distinguished from a character special file by providing access to the device in a manner such that the hardware characteristics of the device are not visible.

3.80 Braces

The characters ' {' (left-curly-bracket) and ' } ' (right-curly-bracket). When used in the phrase “enclosed in (curly) braces” the symbol ' {' immediately precedes the object to be enclosed, and ' } ' immediately follows it. When describing these characters in the portable character set, the names <left-curly-bracket> and <left-brace> are used for ' { ', and <right-curly-bracket> and <right-brace> are used for ' } '.

3.81 Brackets

The characters ' [' (left-square-bracket) and '] ' (right-square-bracket). When used in the phrase “enclosed in (square) brackets” the symbol ' [' immediately precedes the object to be enclosed, and '] ' immediately follows it. When describing these characters in the portable character set, the names <left-square-bracket> and <right-square-bracket> are used.

3.82 Broadcast

The transfer of data from one endpoint to several endpoints, as described in RFC 919 and RFC 922.

3.83 Built-In Utility (or Built-In)

A utility implemented within a shell. The utilities referred to as special built-ins have special qualities. Unless qualified, the term “built-in” includes the special built-in utilities. Regular built-ins are not required to be actually built into the shell on the implementation, but they do have special command-search qualities.

Note: Special Built-In Utilities are defined in detail in XCU [Section 2.14](#) (on page 2356).

Regular Built-In Utilities are defined in detail in XCU [Section 2.9.1.1](#) (on page 2339).

3.84 Byte

An individually addressable unit of data storage that is exactly an octet, used to store a character or a portion of a character; see also [Section 3.87](#) (on page 47). A byte is composed of a contiguous sequence of 8 bits. The least significant bit is called the “low-order” bit; the most significant is called the “high-order” bit.

Note: The definition of byte from the ISO C standard is broader than the above and might accommodate hardware architectures with different sized addressable units than octets.

3.85 Byte Input/Output Functions

The functions that perform byte-oriented input from streams or byte-oriented output to streams: *fgetc()*, *fgets()*, *fprintf()*, *fputc()*, *fputs()*, *fread()*, *fscanf()*, *fwrite()*, *getc()*, *getchar()*, *getdelim()*, *getline()*, *gets()*, *printf()*, *putc()*, *putchar()*, *puts()*, *scanf()*, *ungetc()*, *vfprintf()*, and *vprintf()*.

Note: Functions are defined in detail in the System Interfaces volume of POSIX.1-2008.

3.86 Carriage-Return Character (<carriage-return>)

A character that in the output stream indicates that printing should start at the beginning of the same physical line in which the carriage-return occurred. It is the character designated by ‘\r’ in the C language. It is unspecified whether this character is the exact sequence transmitted to an output device by the system to accomplish the movement to the beginning of the line.

3.87 Character

A sequence of one or more bytes representing a single graphic symbol or control code.

Note: This term corresponds to the ISO C standard term multi-byte character, where a single-byte character is a special case of a multi-byte character. Unlike the usage in the ISO C standard, *character* here has no necessary relationship with storage space, and *byte* is used when storage space is discussed.

See the definition of the portable character set in [Section 6.1](#) (on page 125) for a further explanation of the graphical representations of (abstract) characters, as opposed to character encodings.

3.88 Character Array

An array of elements of type **char**.

3.89 Character Class

A named set of characters sharing an attribute associated with the name of the class. The classes and the characters that they contain are dependent on the value of the *LC_CTYPE* category in the current locale.

Note: The *LC_CTYPE* category is defined in detail in [Section 7.3.1](#) (on page 139).

3.90 Character Set

A finite set of different characters used for the representation, organization, or control of data.

3.91 Character Special File

A file that refers to a device (such as a terminal device file) or that has special properties (such as */dev/null*).

Note: The General Terminal Interface is defined in detail in [Chapter 11](#) (on page 199).

3.92 Character String

A contiguous sequence of characters terminated by and including the first null byte.

3.93 Child Process

A new process created (by *fork()*, *posix_spawn()*, or *posix_spawnnp()*) by a given process. A child process remains the child of the creating process as long as both processes continue to exist.

Note: The *fork()*, *posix_spawn()*, and *posix_spawnnp()* functions are defined in detail in the System Interfaces volume of POSIX.1-2008.

3.94 Circumflex Character (<circumflex>)

The character ' ^ '.

3.95 Clock

A software or hardware object that can be used to measure the apparent or actual passage of time.

The current value of the time measured by a clock can be queried and, possibly, set to a value within the legal range of the clock.

3.96 Clock Jump

The difference between two successive distinct values of a clock, as observed from the application via one of the “get time” operations.

3.97 Clock Tick

An interval of time; an implementation-defined number of these occur each second. Clock ticks are one of the units that may be used to express a value found in type **clock_t**.

3.98 Coded Character Set

A set of unambiguous rules that establishes a character set and the one-to-one relationship between each character of the set and its bit representation.

3.99 Codeset

The result of applying rules that map a numeric code value to each element of a character set. An element of a character set may be related to more than one numeric code value but the reverse is not true. However, for state-dependent encodings the relationship between numeric code values and elements of a character set may be further controlled by state information. The character set may contain fewer elements than the total number of possible numeric code values; that is, some code values may be unassigned.

Note: Character Encoding is defined in detail in [Section 6.2](#) (on page 128).

3.100 Collating Element

The smallest entity used to determine the logical ordering of character or wide-character strings; see also [Section 3.102](#). A collating element consists of either a single character, or two or more characters collating as a single entity. The value of the *LC_COLLATE* category in the current locale determines the current set of collating elements.

3.101 Collation

The logical ordering of character or wide-character strings according to defined precedence rules. These rules identify a collation sequence between the collating elements, and such additional rules that can be used to order strings consisting of multiple collating elements.

3.102 Collation Sequence

The relative order of collating elements as determined by the setting of the *LC_COLLATE* category in the current locale. The collation sequence is used for sorting and is determined from the collating weights assigned to each collating element. In the absence of weights, the collation sequence is the order in which collating elements are specified between **order_start** and **order_end** keywords in the *LC_COLLATE* category.

Multi-level sorting is accomplished by assigning elements one or more collation weights, up to the limit {*COLL_WEIGHTS_MAX*}. On each level, elements may be given the same weight (at the primary level, called an equivalence class; see also [Section 3.151](#), on page 57) or be omitted from the sequence. Strings that collate equally using the first assigned weight (primary ordering) are then compared using the next assigned weight (secondary ordering), and so on.

Note: {*COLL_WEIGHTS_MAX*} is defined in detail in [<limits.h>](#).

3.103 Column Position

A unit of horizontal measure related to characters in a line.

It is assumed that each character in a character set has an intrinsic column width independent of any output device. Each printable character in the portable character set has a column width of one. The standard utilities, when used as described in POSIX.1-2008, assume that all characters have integral column widths. The column width of a character is not necessarily related to the internal representation of the character (numbers of bits or bytes).

The column position of a character in a line is defined as one plus the sum of the column widths of the preceding characters in the line. Column positions are numbered starting from 1.

3.104 Command

A directive to the shell to perform a particular task.

Note: Shell Commands are defined in detail in XCU [Section 2.9](#) (on page 2338).

3.105 Command Language Interpreter

An interface that interprets sequences of text input as commands. It may operate on an input stream or it may interactively prompt and read commands from a terminal. It is possible for applications to invoke utilities through a number of interfaces, which are collectively considered to act as command interpreters. The most obvious of these are the *sh* utility and the *system()* function, although *popen()* and the various forms of *exec* may also be considered to behave as interpreters.

Note: The *sh* utility is defined in detail in the Shell and Utilities volume of POSIX.1-2008.

The *system()*, *popen()*, and *exec* functions are defined in detail in the System Interfaces volume of POSIX.1-2008.

3.106 Composite Graphic Symbol

A graphic symbol consisting of a combination of two or more other graphic symbols in a single character position, such as a diacritical mark and a base character.

3.107 Condition Variable

A synchronization object which allows a thread to suspend execution, repeatedly, until some associated predicate becomes true. A thread whose execution is suspended on a condition variable is said to be blocked on the condition variable.

3.108 Connected Socket

A connection-mode socket for which a connection has been established, or a connectionless-mode socket for which a peer address has been set. See also [Section 3.109](#), [Section 3.110](#), [Section 3.111](#), and [Section 3.350](#) (on page 89).

3.109 Connection

An association established between two or more endpoints for the transfer of data

3.110 Connection Mode

The transfer of data in the context of a connection; see also [Section 3.111](#).

3.111 Connectionless Mode

The transfer of data other than in the context of a connection; see also [Section 3.110](#) and [Section 3.124](#) (on page 53).

3.112 Control Character

A character, other than a graphic character, that affects the recording, processing, transmission, or interpretation of text.

3.113 Control Operator

In the shell command language, a token that performs a control function. It is one of the following symbols:

`&    &&    (    )    ;    ;;    newline    |    ||`

The end-of-input indicator used internally by the shell is also considered a control operator.

Note: Token Recognition is defined in detail in XCU [Section 2.3](#) (on page 2321).

3.114 Controlling Process

The session leader that established the connection to the controlling terminal. If the terminal subsequently ceases to be a controlling terminal for this session, the session leader ceases to be the controlling process.

3.115 Controlling Terminal

A terminal that is associated with a session. Each session may have at most one controlling terminal associated with it, and a controlling terminal is associated with exactly one session. Certain input sequences from the controlling terminal cause signals to be sent to all processes in the foreground process group associated with the controlling terminal.

Note: The General Terminal Interface is defined in detail in [Chapter 11](#) (on page 199).

3.116 Conversion Descriptor

A per-process unique value used to identify an open codeset conversion.

3.117 Core File

A file of unspecified format that may be generated when a process terminates abnormally.

3.118 CPU Time (Execution Time)

The time spent executing a process or thread, including the time spent executing system services on behalf of that process or thread. If the Threads option is supported, then the value of the CPU-time clock for a process is implementation-defined. With this definition the sum of all the execution times of all the threads in a process might not equal the process execution time, even in a single-threaded process, because implementations may differ in how they account for time during context switches or for other reasons.

3.119 CPU-Time Clock

A clock that measures the execution time of a particular process or thread.

3.120 CPU-Time Timer

A timer attached to a CPU-time clock.

3.121 Current Job

In the context of job control, the job that will be used as the default for the *fg* or *bg* utilities. There is at most one current job; see also [Section 3.204](#) (on page 65).

3.122 Current Working Directory

See *Working Directory* in [Section 3.441](#) (on page 104).

3.123 Cursor Position

The line and column position on the screen denoted by the terminal's cursor.

3.124 Datagram

A unit of data transferred from one endpoint to another in connectionless mode service.

3.125 Data Segment

Memory associated with a process, that can contain dynamically allocated data.

3.126 Deferred Batch Service

A service that is performed as a result of events that are asynchronous with respect to requests.

Note: Once a batch job has been created, it is subject to deferred services.

3.127 Device

A computer peripheral or an object that appears to the application as such.

3.128 Device ID

A non-negative integer used to identify a device.

3.129 Directory

A file that contains directory entries. No two directory entries in the same directory have the same name.

3.130 Directory Entry (or Link)

An object that associates a filename with a file. Several directory entries can associate names

1634 with the same file.

1635 **3.131 Directory Stream**

1636 A sequence of all the directory entries in a particular directory. An open directory stream may be
1637 implemented using a file descriptor.

1638 **3.132 Disarm (a Timer)**

1639 To stop a timer from measuring the passage of time, disabling any future process notifications
1640 (until the timer is armed again).

1641 **3.133 Display**

1642 To output to the user's terminal. If the output is not directed to a terminal, the results are
1643 undefined.

1644 **3.134 Display Line**

1645 A line of text on a physical device or an emulation thereof. Such a line will have a maximum
1646 number of characters which can be presented.

1647 **Note:** This may also be written as "line on the display".

1648 **3.135 Dollar-Sign Character (<dollar-sign>)**

1649 The character '\$'.

1650 **3.136 Dot**

1651 In the context of naming files, the filename consisting of a single <period> character ('.').

1652 **Note:** In the context of shell special built-in utilities, see *dot* in XCU [Section 2.14](#) (on page 2356).

1653 Pathname Resolution is defined in detail in [Section 4.12](#) (on page 111).

3.137 Dot-Dot

The filename consisting solely of two <period> characters (" . ").

Note: Pathname Resolution is defined in detail in [Section 4.12](#) (on page 111).

3.138 Double-Quote Character

The character ' " ', also known as <quotation-mark>.

Note: The “double” adjective in this term refers to the two strokes in the character glyph. POSIX.1-2008 never uses the term “double-quote” to refer to two apostrophes or quotation-marks.

3.139 Downshifting

The conversion of an uppercase character that has a single-character lowercase representation into this lowercase representation.

3.140 Driver

A module that controls data transferred to and received from devices.

Note: Drivers are traditionally written to be a part of the system implementation, although they are frequently written separately from the writing of the implementation. A driver may contain processor-specific code, and therefore be non-portable.

3.141 Effective Group ID

An attribute of a process that is used in determining various permissions, including file access permissions; see also [Section 3.189](#) (on page 63).

3.142 Effective User ID

An attribute of a process that is used in determining various permissions, including file access permissions; see also [Section 3.430](#) (on page 102).

3.143 Eight-Bit Transparency

The ability of a software component to process 8-bit characters without modifying or utilizing any part of the character in a way that is inconsistent with the rules of the current coded character set.

3.144 Empty Directory

A directory that contains, at most, directory entries for dot and dot-dot, and has exactly one link to it (other than its own dot entry, if one exists), in dot-dot. No other links to the directory may exist. It is unspecified whether an implementation can ever consider the root directory to be empty.

3.145 Empty Line

A line consisting of only a <newline>; see also [Section 3.75](#) (on page 44).

3.146 Empty String (or Null String)

A string whose first byte is a null byte.

3.147 Empty Wide-Character String

A wide-character string whose first element is a null wide-character code.

3.148 Encoding Rule

The rules used to convert between wide-character codes and multi-byte character codes.

Note: Stream Orientation and Encoding Rules are defined in detail in XSH [Section 2.5.2](#) (on page 498).

3.149 Entire Regular Expression

The concatenated set of one or more basic regular expressions or extended regular expressions that make up the pattern specified for string selection.

Note: Regular Expressions are defined in detail in [Chapter 9](#) (on page 181).

3.150 Epoch

The time zero hours, zero minutes, zero seconds, on January 1, 1970 Coordinated Universal Time (UTC).

Note: See also *Seconds Since the Epoch* defined in [Section 4.15](#) (on page 113).

3.151 Equivalence Class

A set of collating elements with the same primary collation weight.

Elements in an equivalence class are typically elements that naturally group together, such as all accented letters based on the same base letter.

The collation order of elements within an equivalence class is determined by the weights assigned on any subsequent levels after the primary weight.

3.152 Era

A locale-specific method for counting and displaying years.

Note: The *LC_TIME* category is defined in detail in [Section 7.3.5](#) (on page 159).

3.153 Event Management

The mechanism that enables applications to register for and be made aware of external events such as data becoming available for reading.

3.154 Executable File

A regular file acceptable as a new process image file by the equivalent of the *exec* family of functions, and thus usable as one form of a utility. The standard utilities described as compilers can produce executable files, but other unspecified methods of producing executable files may also be provided. The internal format of an executable file is unspecified, but a conforming application cannot assume an executable file is a text file.

1720 3.155 Execute

1721 To perform command search and execution actions, as defined in the Shell and Utilities volume
1722 of POSIX.1-2008; see also [Section 3.201](#) (on page 65).

1723 **Note:** Command Search and Execution is defined in detail in XCU [Section 2.9.1.1](#) (on page 2339).

1724 3.156 Execution Time

1725 See *CPU Time* in [Section 3.118](#) (on page 52).

1726 3.157 Execution Time Monitoring

1727 A set of execution time monitoring primitives that allow online measuring of thread and process
1728 execution times.

1729 3.158 Expand

1730 In the shell command language, when not qualified, the act of applying word expansions.

1731 **Note:** Word Expansions are defined in detail in XCU [Section 2.6](#) (on page 2327).

1732 3.159 Extended Regular Expression (ERE)

1733 A regular expression (see also [Section 3.317](#), on page 84) that is an alternative to the Basic
1734 Regular Expression using a more extensive syntax, occasionally used by some utilities.

1735 **Note:** Extended Regular Expressions are described in detail in [Section 9.4](#) (on page 188).

1736 3.160 Extended Security Controls

1737 Implementation-defined security controls allowed by the file access permission and appropriate
1738 privileges (see also [Section 3.20](#), on page 36) mechanisms, through which an implementation can
1739 support different security policies from those described in POSIX.1-2008.

1740 **Note:** See also *Extended Security Controls* defined in [Section 4.3](#) (on page 107).

1741 File Access Permissions are defined in detail in [Section 4.4](#) (on page 108).

3.161 Feature Test Macro

A macro used to determine whether a particular set of features is included from a header.

Note: See also XSH [Section 2.2](#) (on page 472).

3.162 Field

In the shell command language, a unit of text that is the result of parameter expansion, arithmetic expansion, command substitution, or field splitting. During command processing, the resulting fields are used as the command name and its arguments.

Note: Parameter Expansion is defined in detail in XCU [Section 2.6.2](#) (on page 2328).

Arithmetic Expansion is defined in detail in XCU [Section 2.6.4](#) (on page 2332).

Command Substitution is defined in detail in XCU [Section 2.6.3](#) (on page 2331).

Field Splitting is defined in detail in XCU [Section 2.6.5](#) (on page 2333).

For further information on command processing, see XCU [Section 2.9.1](#) (on page 2338).

3.163 FIFO Special File (or FIFO)

A type of file with the property that data written to such a file is read on a first-in-first-out basis.

Note: Other characteristics of FIFOs are described in the System Interfaces volume of POSIX.1-2008, *lseek()*, *open()*, *read()*, and *write()*.

3.164 File

An object that can be written to, or read from, or both. A file has certain attributes, including access permissions and type. File types include regular file, character special file, block special file, FIFO special file, symbolic link, socket, and directory. Other types of files may be supported by the implementation.

3.165 File Description

See *Open File Description* in [Section 3.254](#) (on page 73).

3.166 File Descriptor

A per-process unique, non-negative integer used to identify an open file for the purpose of file access. The value of a newly-created file descriptor is from zero to {OPEN_MAX}-1. A file descriptor can have a value greater than or equal to {OPEN_MAX} if the value of {OPEN_MAX} has decreased (see *sysconf()*) since the file descriptor was opened. File descriptors may also be used to implement message catalog descriptors and directory streams; see also [Section 3.254](#) (on page 73).

Note: {OPEN_MAX} is defined in detail in [<limits.h>](#).

3.167 File Group Class

The property of a file indicating access permissions for a process related to the group identification of a process. A process is in the file group class of a file if the process is not in the file owner class and if the effective group ID or one of the supplementary group IDs of the process matches the group ID associated with the file. Other members of the class may be implementation-defined.

3.168 File Mode

An object containing the file mode bits and file type of a file.

Note: File mode bits and file types are defined in detail in [<sys/stat.h>](#).

3.169 File Mode Bits

A file's file permission bits, set-user-ID-on-execution bit (S_ISUID), set-group-ID-on-execution bit (S_ISGID), and, on directories, the restricted deletion flag bit (S_ISVTX).

Note: File Mode Bits are defined in detail in [<sys/stat.h>](#).

3.170 Filename

A sequence of bytes consisting of 1 to {NAME_MAX} bytes used to name a file. The bytes composing the name shall not contain the <NUL> or <slash> characters. In the context of a pathname, each filename shall be followed by a <slash> or a <NUL> character; elsewhere, a filename followed by a <NUL> character forms a string (but not necessarily a character string). The filenames **dot** and **dot-dot** have special meaning. A filename is sometimes referred to as a "pathname component". See also [Section 3.267](#) (on page 75).

Note: Pathname Resolution is defined in detail in [Section 4.12](#) (on page 111).

1794 3.171 Filename String

1795 A string consisting of a filename followed by a <NUL> character.

+

1796 3.172 File Offset

1797 The byte position in the file where the next I/O operation begins. Each open file description
1798 associated with a regular file, block special file, or directory has a file offset. A character special
1799 file that does not refer to a terminal device may have a file offset. There is no file offset specified
1800 for a pipe or FIFO.

1801 3.173 File Other Class

1802 The property of a file indicating access permissions for a process related to the user and group
1803 identification of a process. A process is in the file other class of a file if the process is not in the
1804 file owner class or file group class.

1805 3.174 File Owner Class

1806 The property of a file indicating access permissions for a process related to the user
1807 identification of a process. A process is in the file owner class of a file if the effective user ID of
1808 the process matches the user ID of the file.

1809 3.175 File Permission Bits

1810 Information about a file that is used, along with other information, to determine whether a
1811 process has read, write, or execute/search permission to a file. The bits are divided into three
1812 parts: owner, group, and other. Each part is used with the corresponding file class of processes.
1813 These bits are contained in the file mode.

1814 **Note:** File modes are defined in detail in [<sys/stat.h>](#).

1815 File Access Permissions are defined in detail in [Section 4.4](#) (on page 108).

1816 3.176 File Serial Number

1817 A per-file system unique identifier for a file.

1818 3.177 File System

1819 A collection of files and certain of their attributes. It provides a name space for file serial
1820 numbers referring to those files.

1821 3.178 File Type

1822 See *File* in [Section 3.164](#) (on page 59).

1823 3.179 Filter

1824 A command whose operation consists of reading data from standard input or a list of input files
1825 and writing data to standard output. Typically, its function is to perform some transformation
1826 on the data stream.

1827 3.180 First Open (of a File)

1828 When a process opens a file that is not currently an open file within any process.

1829 3.181 Flow Control

1830 The mechanism employed by a communications provider that constrains a sending entity to
1831 wait until the receiving entities can safely receive additional data without loss.

1832 3.182 Foreground Job

1833 See *Foreground Process Group* in [Section 3.184](#).

1834 3.183 Foreground Process

1835 A process that is a member of a foreground process group.

1836 3.184 Foreground Process Group (or Foreground Job)

1837 A process group whose member processes have certain privileges, denied to processes in
1838 background process groups, when accessing their controlling terminal. Each session that has
1839 established a connection with a controlling terminal has at most one process group of the session
1840 as the foreground process group of that controlling terminal.

1841 **Note:** The General Terminal Interface is defined in detail in [Chapter 11](#).

1842 3.185 Foreground Process Group ID

1843 The process group ID of the foreground process group.

3.186 Form-Feed Character (<form-feed>)

A character that in the output stream indicates that printing should start on the next page of an output device. It is the character designated by '`\f`' in the C language. If the form-feed is not the first character of an output line, the result is unspecified. It is unspecified whether this character is the exact sequence transmitted to an output device by the system to accomplish the movement to the next page.

3.187 Graphic Character

A member of the **graph** character class of the current locale.

Note: The **graph** character class is defined in detail in [Section 7.3.1](#) (on page 139).

3.188 Group Database

A system database that contains at least the following information for each group ID:

- Group name
- Numerical group ID
- List of users allowed in the group

The list of users allowed in the group is used by the *newgrp* utility.

Note: The *newgrp* utility is defined in detail in the Shell and Utilities volume of POSIX.1-2008.

3.189 Group ID

A non-negative integer, which can be contained in an object of type **gid_t**, that is used to identify a group of system users. Each system user is a member of at least one group. When the identity of a group is associated with a process, a group ID value is referred to as a real group ID, an effective group ID, one of the supplementary group IDs, or a saved set-group-ID.

3.190 Group Name

A string that is used to identify a group; see also [Section 3.188](#). To be portable across conforming systems, the value is composed of characters from the portable filename character set. The <hyphen> should not be used as the first character of a portable group name.

3.191 Hard Limit

A system resource limitation that may be reset to a lesser or greater limit by a privileged process. A non-privileged process is restricted to only lowering its hard limit.

1872 3.192 Hard Link

1873 The relationship between two directory entries that represent the same file; see also [Section 3.130](#)
1874 (on page 53). The result of an execution of the *ln* utility (without the *-s* option) or the *link()*
1875 function. This term is contrasted against symbolic link; see also [Section 3.375](#) (on page 94).

1876 3.193 Home Directory

1877 The directory specified by the *HOME* environment variable.

1878 3.194 Host Byte Order

1879 The arrangement of bytes in any integer type when using a specific machine architecture.

1880 **Note:** Two common methods of byte ordering are big-endian and little-endian. Big-endian is a format
1881 for storage of binary data in which the most significant byte is placed first, with the rest in
1882 descending order. Little-endian is a format for storage or transmission of binary data in which
1883 the least significant byte is placed first, with the rest in ascending order. See also [Section 4.9](#) (on
1884 page 110).

1885 3.195 Incomplete Line

1886 A sequence of one or more non-*<newline>* characters at the end of the file.

1887 3.196 Inf

1888 A value representing +infinity or a value representing -infinity that can be stored in a floating
1889 type. Not all systems support the Inf values.

1890 3.197 Instrumented Application

1891 An application that contains at least one call to the trace point function *posix_trace_event()*. Each
1892 process of an instrumented application has a mapping of trace event names to trace event type
1893 identifiers. This mapping is used by the trace stream that is created for that process.

1894 3.198 Interactive Shell

1895 A processing mode of the shell that is suitable for direct user interaction.

3.199 Internationalization

The provision within a computer program of the capability of making itself adaptable to the requirements of different native languages, local customs, and coded character sets.

3.200 Interprocess Communication

A functionality enhancement to add a high-performance, deterministic interprocess communication facility for local communication.

3.201 Invoke

To perform command search and execution actions, except that searching for shell functions and special built-in utilities is suppressed; see also [Section 3.155](#) (on page 58).

Note: Command Search and Execution is defined in detail in XCU [Section 2.9.1.1](#) (on page 2339).

3.202 Job

A set of processes, comprising a shell pipeline, and any processes descended from it, that are all in the same process group.

Note: See also XCU [Section 2.9.2](#) (on page 2341).

3.203 Job Control

A facility that allows users selectively to stop (suspend) the execution of processes and continue (resume) their execution at a later point. The user typically employs this facility via the interactive interface jointly supplied by the terminal I/O driver and a command interpreter.

3.204 Job Control Job ID

A handle that is used to refer to a job. The job control job ID can be any of the forms shown in the following table:

Table 3-1 Job Control Job ID Formats

Job Control Job ID	Meaning
%%	Current job.
%+	Current job.
%-	Previous job.
%n	Job number <i>n</i> .
%string	Job whose command begins with <i>string</i> .
%?string	Job whose command contains <i>string</i> .

3.205 Last Close (of a File)

When a process closes a file, resulting in the file not being an open file within any process.

3.206 Line

A sequence of zero or more non-<newline> characters plus a terminating <newline> character.

3.207 Linger

The period of time before terminating a connection, to allow outstanding data to be transferred.

3.208 Link

See *Directory Entry* in [Section 3.130](#) (on page 53).

3.209 Link Count

The number of directory entries that refer to a particular file.

3.210 Local Customs

The conventions of a geographical area or territory for such things as date, time, and currency formats.

3.211 Local Interprocess Communication (Local IPC)

The transfer of data between processes in the same system.

1941 3.212 Locale

1942 The definition of the subset of a user's environment that depends on language and cultural
1943 conventions.

1944 **Note:** Locales are defined in detail in [Chapter 7](#) (on page 135).

1945 3.213 Localization

1946 The process of establishing information within a computer system specific to the operation of
1947 particular native languages, local customs, and coded character sets.

1948 3.214 Login

1949 The unspecified activity by which a user gains access to the system. Each login is associated
1950 with exactly one login name.

1951 3.215 Login Name

1952 A user name that is associated with a login.

1953 3.216 Map

1954 To create an association between a page-aligned range of the address space of a process and
1955 some memory object, such that a reference to an address in that range of the address space
1956 results in a reference to the associated memory object. The mapped memory object is not
1957 necessarily memory-resident.

1958 3.217 Marked Message

1959 A STREAMS message on which a certain flag is set. Marking a message gives the application
1960 protocol-specific information. An application can use *ioctl()* to determine whether a given
1961 message is marked.

1962 **Note:** The *ioctl()* function is defined in detail in the System Interfaces volume of POSIX.1-2008.

3.218 Matched

A state applying to a sequence of zero or more characters when the characters in the sequence correspond to a sequence of characters defined by a basic regular expression or extended regular expression pattern.

Note: Regular Expressions are defined in detail in [Chapter 9](#) (on page 181).

3.219 Memory Mapped Files

A facility to allow applications to access files as part of the address space.

3.220 Memory Object

One of:

- A file (see [Section 3.164](#), on page 59)
- A shared memory object (see [Section 3.342](#), on page 88)
- A typed memory object (see [Section 3.423](#), on page 101)

When used in conjunction with `mmap()`, a memory object appears in the address space of the calling process.

Note: The `mmap()` function is defined in detail in the System Interfaces volume of POSIX.1-2008.

3.221 Memory-Resident

The process of managing the implementation in such a way as to provide an upper bound on memory access times.

3.222 Message

In the context of programmatic message passing, information that can be transferred between processes or threads by being added to and removed from a message queue. A message consists of a fixed-size message buffer.

3.223 Message Catalog

In the context of providing natural language messages to the user, a file or storage area containing program messages, command prompts, and responses to prompts for a particular native language, territory, and codeset.

1989 3.224 Message Catalog Descriptor

1990 In the context of providing natural language messages to the user, a per-process unique value
1991 used to identify an open message catalog. A message catalog descriptor may be implemented
1992 using a file descriptor.

1993 3.225 Message Queue

1994 In the context of programmatic message passing, an object to which messages can be added and
1995 removed. Messages may be removed in the order in which they were added or in priority order.

1996 3.226 Mode

1997 A collection of attributes that specifies a file's type and its access permissions.

1998 **Note:** File Access Permissions are defined in detail in [Section 4.4](#) (on page 108).

1999 3.227 Monotonic Clock

2000 A clock measuring real time, whose value cannot be set via `clock_settime()` and which cannot
2001 have negative clock jumps.

2002 3.228 Mount Point

2003 Either the system root directory or a directory for which the `st_dev` field of structure **stat** differs
2004 from that of its parent directory.

2005 **Note:** The **stat** structure is defined in detail in [<sys/stat.h>](#).

2006 3.229 Multi-Character Collating Element

2007 A sequence of two or more characters that collate as an entity. For example, in some coded
2008 character sets, an accented character is represented by a non-spacing accent, followed by the
2009 letter. Other examples are the Spanish elements *ch* and *ll*.

2010 3.230 Mutex

2011 A synchronization object used to allow multiple threads to serialize their access to shared data.
2012 The name derives from the capability it provides; namely, mutual-exclusion. The thread that has
2013 locked a mutex becomes its owner and remains the owner until that same thread unlocks the
2014 mutex.

2015 3.231 Name

2016 In the shell command language, a word consisting solely of underscores, digits, and alphabetic
 2017 from the portable character set. The first character of a name is not a digit.

2018 **Note:** The Portable Character Set is defined in detail in [Section 6.1](#) (on page 125).

2019 3.232 Named STREAM

2020 A STREAMS-based file descriptor that is attached to a name in the file system name space. All
 2021 subsequent operations on the named STREAM act on the STREAM that was associated with the
 2022 file descriptor until the name is disassociated from the STREAM.

2023 3.233 NaN (Not a Number)

2024 A set of values that may be stored in a floating type but that are neither Inf nor valid floating-
 2025 point numbers. Not all systems support NaN values.

2026 3.234 Native Language

2027 A computer user's spoken or written language, such as American English, British English,
 2028 Danish, Dutch, French, German, Italian, Japanese, Norwegian, or Swedish.

2029 3.235 Negative Response

2030 An input string that matches one of the responses acceptable to the *LC_MESSAGES* category
 2031 keyword **noexpr**, matching an extended regular expression in the current locale.

2032 **Note:** The *LC_MESSAGES* category is defined in detail in [Section 7.3.6](#) (on page 164).

2033 3.236 Network

2034 A collection of interconnected hosts.

2035 **Note:** The term “network” in POSIX.1-2008 is used to refer to the network of hosts. The term “batch
 2036 system” is used to refer to the network of batch servers.

2037 3.237 Network Address

2038 A network-visible identifier used to designate specific endpoints in a network. Specific
 2039 endpoints on host systems have addresses, and host systems may also have addresses.

3.238 Network Byte Order

The way of representing any integer type such that, when transmitted over a network via a network endpoint, the **int** type is transmitted as an appropriate number of octets with the most significant octet first, followed by any other octets in descending order of significance.

Note: This order is more commonly known as big-endian ordering. See also [Section 4.9](#) (on page 110).

3.239 Newline Character (<newline>)

A character that in the output stream indicates that printing should start at the beginning of the next line. It is the character designated by '`\n`' in the C language. It is unspecified whether this character is the exact sequence transmitted to an output device by the system to accomplish the movement to the next line.

3.240 Nice Value

A number used as advice to the system to alter process scheduling. Numerically smaller values give a process additional preference when scheduling a process to run. Numerically larger values reduce the preference and make a process less likely to run. Typically, a process with a smaller nice value runs to completion more quickly than an equivalent process with a higher nice value. The symbol {NZERO} specifies the default nice value of the system.

3.241 Non-Blocking

A property of an open file description that causes function calls involving it to return without delay when it is detected that the requested action associated with the function call cannot be completed without unknown delay.

Note: The exact semantics are dependent on the type of file associated with the open file description. For data reads from devices such as ttys and FIFOs, this property causes the read to return immediately when no data was available. Similarly, for writes, it causes the call to return immediately when the thread would otherwise be delayed in the write operation; for example, because no space was available. For networking, it causes functions not to await protocol events (for example, acknowledgements) to occur. See also XSH [Section 2.10.7](#) (on page 524).

3.242 Non-Spacing Characters

A character, such as a character representing a diacritical mark in the ISO/IEC 6937:2001 standard coded graphic character set, which is used in combination with other characters to form composite graphic symbols.

2070 3.243 NUL

2071 A character with all bits set to zero.

2072 3.244 Null Byte

2073 A byte with all bits set to zero.

2074 3.245 Null Pointer

2075 A pointer obtained by converting an integer constant expression with the value 0, or such an
2076 expression cast to type **void ***, to a pointer type; for example, **(char *)0**. The C language
2077 guarantees that a null pointer compares unequal to a pointer to any object or function, so it is
2078 used by many functions that return pointers to indicate an error.

2079 3.246 Null String

2080 See *Empty String* in [Section 3.146](#) (on page 56).

2081 3.247 Null Wide-Character Code

2082 A wide-character code with all bits set to zero.

2083 3.248 Number-Sign Character (<number-sign>)

2084 The character ' # ', also known as hash sign.

2085 3.249 Object File

2086 A regular file containing the output of a compiler, formatted as input to a linkage editor for
2087 linking with other object files into an executable form. The methods of linking are unspecified
2088 and may involve the dynamic linking of objects at runtime. The internal format of an object file
2089 is unspecified, but a conforming application cannot assume an object file is a text file.

2090 3.250 Octet

2091 Unit of data representation that consists of eight contiguous bits.

2092 **3.251 Offset Maximum**

2093 An attribute of an open file description representing the largest value that can be used as a file
2094 offset.

2095 **3.252 Opaque Address**

2096 An address such that the entity making use of it requires no details about its contents or format.

2097 **3.253 Open File**

2098 A file that is currently associated with a file descriptor.

2099 **3.254 Open File Description**

2100 A record of how a process or group of processes is accessing a file. Each file descriptor refers to
2101 exactly one open file description, but an open file description can be referred to by more than
2102 one file descriptor. The file offset, file status, and file access modes are attributes of an open file
2103 description.

2104 **3.255 Operand**

2105 An argument to a command that is generally used as an object supplying information to a utility
2106 necessary to complete its processing. Operands generally follow the options in a command line.

2107 **Note:** Utility Argument Syntax is defined in detail in [Section 12.1](#) (on page 213).

2108 **3.256 Operator**

2109 In the shell command language, either a control operator or a redirection operator.

2110 3.257 Option

2111 An argument to a command that is generally used to specify changes in the utility's default
2112 behavior.

2113 **Note:** Utility Argument Syntax is defined in detail in [Section 12.1](#) (on page 213).

2114 3.258 Option-Argument

2115 A parameter that follows certain options. In some cases an option-argument is included within
2116 the same argument string as the option—in most cases it is the next argument.

2117 **Note:** Utility Argument Syntax is defined in detail in [Section 12.1](#) (on page 213).

2118 3.259 Orientation

2119 A stream has one of three orientations: unoriented, byte-oriented, or wide-oriented.

2120 **Note:** For further information, see XSH [Section 2.5.2](#) (on page 498).

2121 3.260 Orphaned Process Group

2122 A process group in which the parent of every member is either itself a member of the group or is
2123 not a member of the group's session.

2124 3.261 Page

2125 The granularity of process memory mapping or locking.

2126 Physical memory and memory objects can be mapped into the address space of a process on
2127 page boundaries and in integral multiples of pages. Process address space can be locked into
2128 memory (made memory-resident) on page boundaries and in integral multiples of pages.

2129 3.262 Page Size

2130 The size, in bytes, of the system unit of memory allocation, protection, and mapping. On
2131 systems that have segment rather than page-based memory architectures, the term "page"
2132 means a segment.

3.263 Parameter

In the shell command language, an entity that stores values. There are three types of parameters: variables (named parameters), positional parameters, and special parameters. Parameter expansion is accomplished by introducing a parameter with the '\$' character.

Note: See also XCU [Section 2.5](#) (on page 2323).

In the C language, an object declared as part of a function declaration or definition that acquires a value on entry to the function, or an identifier following the macro name in a function-like macro definition.

3.264 Parent Directory

When discussing a given directory, the directory that both contains a directory entry for the given directory and is represented by the pathname dot-dot in the given directory.

When discussing other types of files, a directory containing a directory entry for the file under discussion.

This concept does not apply to dot and dot-dot.

3.265 Parent Process

The process which created (or inherited) the process under discussion.

3.266 Parent Process ID

An attribute of a new process identifying the parent of the process. The parent process ID of a process is the process ID of its creator, for the lifetime of the creator. After the creator's lifetime has ended, the parent process ID is the process ID of an implementation-defined system process.

3.267 Pathname

A string that is used to identify a file. In the context of POSIX.1-2008, a pathname may be limited to {PATH_MAX} bytes, including the terminating null byte. It has optional beginning <slash> characters, followed by zero or more filenames separated by <slash> characters. A pathname can optionally contain one or more trailing <slash> characters. Multiple successive <slash> characters are considered to be the same as one <slash>, except for the case of exactly two leading <slash> characters.

Note: If a pathname consists of only bytes corresponding to characters from the portable filename character set (see [Section 3.278](#), on page 78), <slash> characters, and a single terminating <NUL> character, the pathname will be usable as a character string in all supported locales; otherwise, the pathname might only be a string (rather than a character string). Additionally, since the single-byte encoding of the <slash> character is required to be the same across all locales and to not occur within a multi-byte character, references to a <slash> character within a pathname are well-defined even when the pathname is not a character string. However, this property does not necessarily hold for the remaining characters within the portable filename

2168 character set. +
 2169 Pathname Resolution is defined in detail in [Section 4.12](#) (on page 111).

2170 3.268 Pathname Component

2171 See *Filename* in [Section 3.170](#) (on page 60).

2172 3.269 Path Prefix

2173 The part of a pathname up to, but not including, the last component and any trailing <slash>
 2174 characters, unless the pathname consists entirely of <slash> characters, in which case the path
 2175 prefix is ' / ' for a pathname containing either a single <slash> or three or more <slash>
 2176 characters, and ' / / ' for the pathname //. The path prefix of a pathname containing no <slash>
 2177 characters is empty, but is treated as referring to the current working directory.

2178 **Note:** The term is used both in the sense of identifying part of a pathname that forms the prefix and of
 2179 joining a non-empty path prefix to a filename to form a pathname. In the latter case, the path
 2180 prefix need not have a trailing <slash> (in which case the joining is done with a <slash>
 2181 character).

2182 3.270 Pattern

2183 A sequence of characters used either with regular expression notation or for pathname
 2184 expansion, as a means of selecting various character strings or pathnames, respectively.

2185 **Note:** Regular Expressions are defined in detail in [Chapter 9](#) (on page 181).

2186 See also XCU [Section 2.6.6](#) (on page 2333).

2187 The syntaxes of the two types of patterns are similar, but not identical; POSIX.1-2008 always
 2188 indicates the type of pattern being referred to in the immediate context of the use of the term.

2189 3.271 Period Character (<period>)

2190 The character ' . '. The term “period” is contrasted with dot (see also [Section 3.136](#), on page 54),
 2191 which is used to describe a specific directory entry.

2192 3.272 Permissions

2193 Attributes of an object that determine the privilege necessary to access or manipulate the object.

2194 **Note:** File Access Permissions are defined in detail in [Section 4.4](#) (on page 108).

2195 3.273 Persistence

2196 A mode for semaphores, shared memory, and message queues requiring that the object and its
2197 state (including data, if any) are preserved after the object is no longer referenced by any
2198 process.

2199 Persistence of an object does not imply that the state of the object is maintained across a system
2200 crash or a system reboot.

2201 3.274 Pipe

2202 An object identical to a FIFO which has no links in the file hierarchy.

2203 **Note:** The *pipe()* function is defined in detail in the System Interfaces volume of POSIX.1-2008.

2204 3.275 Polling

2205 A scheduling scheme whereby the local process periodically checks until the pre-specified
2206 events (for example, read, write) have occurred.

2207 3.276 Portable Character Set

2208 The collection of characters that are required to be present in all locales supported by
2209 conforming systems.

2210 **Note:** The Portable Character Set is defined in detail in [Section 6.1](#) (on page 125).

2211 This term is contrasted against the smaller portable filename character set; see also [Section 3.278](#)
2212 (on page 78).

2213 3.277 Portable Filename

- 2214 A filename consisting only of characters from the portable filename character set. +
- 2215 **Note:** Applications should avoid using filenames that have the <hyphen> character as the first +
- 2216 character since this may cause problems when filenames are passed as command line +
- 2217 arguments. +

2218 3.278 Portable Filename Character Set

- 2219 The set of characters from which portable filenames are constructed.
- 2220 A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
- 2221 a b c d e f g h i j k l m n o p q r s t u v w x y z
- 2222 0 1 2 3 4 5 6 7 8 9 . _ -
- 2223 The last three characters are the <period>, <underscore>, and <hyphen> characters, respectively. |
- 2224 See also [Section 3.267](#) (on page 75).

2225 3.279 Positional Parameter

- 2226 In the shell command language, a parameter denoted by a single digit or one or more digits in
- 2227 curly braces.
- 2228 **Note:** For further information, see XCU [Section 2.5.1](#) (on page 2323).

2229 3.280 Preallocation

- 2230 The reservation of resources in a system for a particular use.
- 2231 Preallocation does not imply that the resources are immediately allocated to that use, but merely
- 2232 indicates that they are guaranteed to be available in bounded time when needed.

2233 3.281 Preempted Process (or Thread)

- 2234 A running thread whose execution is suspended due to another thread becoming runnable at a
- 2235 higher priority.

2236 3.282 Previous Job

- 2237 In the context of job control, the job that will be used as the default for the *fg* or *bg* utilities if the
- 2238 current job exits. There is at most one previous job; see also [Section 3.204](#) (on page 65).

2239 3.283 Printable Character

2240 One of the characters included in the **print** character classification of the *LC_CTYPE* category in
2241 the current locale.

2242 **Note:** The *LC_CTYPE* category is defined in detail in [Section 7.3.1](#) (on page 139).

2243 3.284 Printable File

2244 A text file consisting only of the characters included in the **print** and **space** character
2245 classifications of the *LC_CTYPE* category and the <backspace>, all in the current locale.

2246 **Note:** The *LC_CTYPE* category is defined in detail in [Section 7.3.1](#) (on page 139).

2247 3.285 Priority

2248 A non-negative integer associated with processes or threads whose value is constrained to a
2249 range defined by the applicable scheduling policy. Numerically higher values represent higher
2250 priorities.

2251 3.286 Priority Band

2252 The queuing order applied to normal priority STREAMS messages. High priority STREAMS
2253 messages are not grouped by priority bands. The only differentiation made by the STREAMS
2254 mechanism is between zero and non-zero bands, but specific protocol modules may differentiate
2255 between priority bands.

2256 3.287 Priority Inversion

2257 A condition in which a thread that is not voluntarily suspended (waiting for an event or time
2258 delay) is not running while a lower priority thread is running. Such blocking of the higher
2259 priority thread is often caused by contention for a shared resource.

2260 3.288 Priority Scheduling

2261 A performance and determinism improvement facility to allow applications to determine the
2262 order in which threads that are ready to run are granted access to processor resources.

2263 3.289 Priority-Based Scheduling

2264 Scheduling in which the selection of a running thread is determined by the priorities of the
2265 runnable processes or threads.

3.290 Privilege

See *Appropriate Privileges* in [Section 3.20](#) (on page 36).

3.291 Process

An address space with one or more threads executing within that address space, and the required system resources for those threads.

Note: Many of the system resources defined by POSIX.1-2008 are shared among all of the threads within a process. These include the process ID, the parent process ID, process group ID, session membership, real, effective, and saved set-user-ID, real, effective, and saved set-group-ID, supplementary group IDs, current working directory, root directory, file mode creation mask, and file descriptors.

3.292 Process Group

A collection of processes that permits the signaling of related processes. Each process in the system is a member of a process group that is identified by a process group ID. A newly created process joins the process group of its creator.

3.293 Process Group ID

The unique positive integer identifier representing a process group during its lifetime.

Note: See also *Process Group ID Reuse* defined in [Section 4.13](#) (on page 112).

3.294 Process Group Leader

A process whose process ID is the same as its process group ID.

3.295 Process Group Lifetime

The period of time that begins when a process group is created and ends when the last remaining process in the group leaves the group, due either to the end of the lifetime of the last process or to the last remaining process calling the *setsid()* or *setpgid()* functions.

Note: The *setsid()* and *setpgid()* functions are defined in detail in the System Interfaces volume of POSIX.1-2008.

3.296 Process ID

The unique positive integer identifier representing a process during its lifetime.

Note: See also *Process ID Reuse* defined in [Section 4.13](#) (on page 112).

3.297 Process Lifetime

The period of time that begins when a process is created and ends when its process ID is returned to the system. After a process is created by *fork()*, *posix_spawn()*, or *posix_spawnnp()*, it is considered active. At least one thread of control and address space exist until it terminates. It then enters an inactive state where certain resources may be returned to the system, although some resources, such as the process ID, are still in use. When another process executes a *wait()*, *waitid()*, or *waitpid()* function for an inactive process, the remaining resources are returned to the system. The last resource to be returned to the system is the process ID. At this time, the lifetime of the process ends.

Note: The *fork()*, *posix_spawn()*, *posix_spawnnp()*, *wait()*, *waitid()*, and *waitpid()* functions are defined in detail in the System Interfaces volume of POSIX.1-2008.

3.298 Process Memory Locking

A performance improvement facility to bind application programs into the high-performance random access memory of a computer system. This avoids potential latencies introduced by the operating system in storing parts of a program that were not recently referenced on secondary memory devices.

3.299 Process Termination

There are two kinds of process termination:

1. Normal termination occurs by a return from *main()*, when requested with the *exit()*, *_exit()*, or *_Exit()* functions; or when the last thread in the process terminates by returning from its start function, by calling the *pthread_exit()* function, or through cancellation.
2. Abnormal termination occurs when requested by the *abort()* function or when some signals are received.

Note: The *_exit()*, *_Exit()*, *abort()*, and *exit()* functions are defined in detail in the System Interfaces volume of POSIX.1-2008.

3.300 Process-To-Process Communication

The transfer of data between processes.

3.301 Process Virtual Time

The measurement of time in units elapsed by the system clock while a process is executing.

3.302 Program

A prepared sequence of instructions to the system to accomplish a defined task. The term “program” in POSIX.1-2008 encompasses applications written in the Shell Command Language, complex utility input languages (for example, *awk*, *lex*, *sed*, and so on), and high-level languages.

3.303 Protocol

A set of semantic and syntactic rules for exchanging information.

3.304 Pseudo-Terminal

A facility that provides an interface that is identical to the terminal subsystem, except where noted otherwise in POSIX.1-2008. A pseudo-terminal is composed of two devices: the “master device” and a “slave device”. The slave device provides processes with an interface that is identical to the terminal interface, although there need not be hardware behind that interface. Anything written on the master device is presented to the slave as an input and anything written on the slave device is presented as an input on the master side.

3.305 Radix Character

The character that separates the integer part of a number from the fractional part.

3.306 Read-Only File System

A file system that has implementation-defined characteristics restricting modifications.

Note: File Times Update is described in detail in [Section 4.8](#) (on page 109).

3.307 Read-Write Lock

Multiple readers, single writer (read-write) locks allow many threads to have simultaneous read-only access to data while allowing only one thread to have write access at any given time. They are typically used to protect data that is read-only more frequently than it is changed.

Read-write locks can be used to synchronize threads in the current process and other processes if they are allocated in memory that is writable and shared among the cooperating processes and have been initialized for this behavior.

2349 **3.308 Real Group ID**

2350 The attribute of a process that, at the time of process creation, identifies the group of the user
2351 who created the process; see also [Section 3.189](#) (on page 63).

2352 **3.309 Real Time**

2353 Time measured as total units elapsed by the system clock without regard to which thread is
2354 executing.

2355 **3.310 Realtime Signal Extension**

2356 A determinism improvement facility to enable asynchronous signal notifications to an
2357 application to be queued without impacting compatibility with the existing signal functions.

2358 **3.311 Real User ID**

2359 The attribute of a process that, at the time of process creation, identifies the user who created the
2360 process; see also [Section 3.430](#) (on page 102).

2361 **3.312 Record**

2362 A collection of related data units or words which is treated as a unit.

2363 **3.313 Redirection**

2364 In the shell command language, a method of associating files with the input or output of
2365 commands.

2366 **Note:** For further information, see XCU [Section 2.7](#) (on page 2334).

2367 **3.314 Redirection Operator**

2368 In the shell command language, a token that performs a redirection function. It is one of the
2369 following symbols:

2370 < > >| << >> <& >& <<- <>

2371 **3.315 Referenced Shared Memory Object**

2372 A shared memory object that is open or has one or more mappings defined on it.

2373 **3.316 Refresh**

2374 To ensure that the information on the user's terminal screen is up-to-date.

2375 **3.317 Regular Expression**

2376 A pattern that selects specific strings from a set of character strings.

2377 **Note:** Regular Expressions are described in detail in [Chapter 9](#) (on page 181).

2378 **3.318 Region**

2379 In the context of the address space of a process, a sequence of addresses.

2380 In the context of a file, a sequence of offsets.

2381 **3.319 Regular File**

2382 A file that is a randomly accessible sequence of bytes, with no further structure imposed by the
2383 system.

2384 **3.320 Relative Pathname**

2385 A pathname not beginning with a <slash> character.

2386 **Note:** Pathname Resolution is defined in detail in [Section 4.12](#) (on page 111).

2387 **3.321 Relocatable File**

2388 A file holding code or data suitable for linking with other object files to create an executable or a
2389 shared object file.

2390 **3.322 Relocation**

2391 The process of connecting symbolic references with symbolic definitions. For example, when a
2392 program calls a function, the associated call instruction transfers control to the proper
2393 destination address at execution.

2394 **3.323 Requested Batch Service**

2395 A service that is either rejected or performed prior to a response from the service to the
2396 requester.

2397 **3.324 (Time) Resolution**

2398 The minimum time interval that a clock can measure or whose passage a timer can detect.

2399 **3.325 Robust Mutex**

2400 A mutex with the *robust* attribute set.

2401 **Note:** The *robust* attribute is defined in detail by the `pthread_mutexattr_getrobust()` function.

2402 **3.326 Root Directory**

2403 A directory, associated with a process, that is used in pathname resolution for pathnames that
2404 begin with a <slash> character.

2405 **3.327 Runnable Process (or Thread)**

2406 A thread that is capable of being a running thread, but for which no processor is available.

3.328 Running Process (or Thread)

A thread currently executing on a processor. On multi-processor systems there may be more than one such thread in a system at a time.

3.329 Saved Resource Limits

An attribute of a process that provides some flexibility in the handling of unrepresentable resource limits, as described in the *exec* family of functions and *setrlimit()*.

Note: The *exec* and *setrlimit()* functions are defined in detail in the System Interfaces volume of POSIX.1-2008.

3.330 Saved Set-Group-ID

An attribute of a process that allows some flexibility in the assignment of the effective group ID attribute, as described in the *exec* family of functions and *setgid()*.

Note: The *exec* and *setgid()* functions are defined in detail in the System Interfaces volume of POSIX.1-2008.

3.331 Saved Set-User-ID

An attribute of a process that allows some flexibility in the assignment of the effective user ID attribute, as described in the *exec* family of functions and *setuid()*.

Note: The *exec* and *setuid()* functions are defined in detail in the System Interfaces volume of POSIX.1-2008.

3.332 Scheduling

The application of a policy to select a runnable process or thread to become a running process or thread, or to alter one or more of the thread lists.

3.333 Scheduling Allocation Domain

The set of processors on which an individual thread can be scheduled at any given time.

3.334 Scheduling Contention Scope

A property of a thread that defines the set of threads against which that thread competes for resources.

2433 For example, in a scheduling decision, threads sharing scheduling contention scope compete for
2434 processor resources. In POSIX.1-2008, a thread has scheduling contention scope of either
2435 PTHREAD_SCOPE_SYSTEM or PTHREAD_SCOPE_PROCESS.

2436 3.335 Scheduling Policy

2437 A set of rules that is used to determine the order of execution of processes or threads to achieve
2438 some goal.

2439 **Note:** Scheduling Policy is defined in detail in [Section 4.14](#) (on page 112).

2440 3.336 Screen

2441 A rectangular region of columns and lines on a terminal display. A screen may be a portion of a
2442 physical display device or may occupy the entire physical area of the display device.

2443 3.337 Scroll

2444 To move the representation of data vertically or horizontally relative to the terminal screen.
2445 There are two types of scrolling:

- 2446 1. The cursor moves with the data.
- 2447 2. The cursor remains stationary while the data moves.

2448 3.338 Semaphore

2449 A minimum synchronization primitive to serve as a basis for more complex synchronization
2450 mechanisms to be defined by the application program.

2451 **Note:** Semaphores are defined in detail in [Section 4.16](#) (on page 113).

3.339 Session

A collection of process groups established for job control purposes. Each process group is a member of a session. A process is considered to be a member of the session of which its process group is a member. A newly created process joins the session of its creator. A process can alter its session membership; see *setsid()*. There can be multiple process groups in the same session.

Note: The *setsid()* function is defined in detail in the System Interfaces volume of POSIX.1-2008.

3.340 Session Leader

A process that has created a session.

Note: For further information, see the *setsid()* function defined in the System Interfaces volume of POSIX.1-2008.

3.341 Session Lifetime

The period between when a session is created and the end of the lifetime of all the process groups that remain as members of the session.

3.342 Shared Memory Object

An object that represents memory that can be mapped concurrently into the address space of more than one process.

3.343 Shell

A program that interprets sequences of text input as commands. It may operate on an input stream or it may interactively prompt and read commands from a terminal.

3.344 Shell, the

The Shell Command Language Interpreter; a specific instance of a shell.

Note: For further information, see the *sh* utility defined in the Shell and Utilities volume of POSIX.1-2008.

3.345 Shell Script

A file containing shell commands. If the file is made executable, it can be executed by specifying its name as a simple command. Execution of a shell script causes a shell to execute the commands within the script. Alternatively, a shell can be requested to execute the commands in a shell script by specifying the name of the shell script as the operand to the *sh* utility.

Note: Simple Commands are defined in detail in XCU [Section 2.9.1](#) (on page 2338).

The *sh* utility is defined in detail in the Shell and Utilities volume of POSIX.1-2008.

3.346 Signal

A mechanism by which a process or thread may be notified of, or affected by, an event occurring in the system. Examples of such events include hardware exceptions and specific actions by processes. The term signal is also used to refer to the event itself.

3.347 Signal Stack

Memory established for a thread, in which signal handlers catching signals sent to that thread are executed.

3.348 Single-Quote Character

The character designated by `'\''` in the C language, also known as <apostrophe>.

3.349 Slash Character (<slash>)

The character ' / ', also known as solidus.

3.350 Socket

A file of a particular type that is used as a communications endpoint for process-to-process communication as described in the System Interfaces volume of POSIX.1-2008.

3.351 Socket Address

An address associated with a socket or remote endpoint, including an address family identifier and addressing information specific to that address family. The address may include multiple parts, such as a network address associated with a host system and an identifier for a specific endpoint.

2501 3.352 Soft Limit

2502 A resource limitation established for each process that the process may set to any value less than
2503 or equal to the hard limit.

2504 3.353 Source Code

2505 When dealing with the Shell Command Language, input to the command language interpreter.
2506 The term “shell script” is synonymous with this meaning.

2507 When dealing with an ISO/IEC-conforming programming language, source code is input to a
2508 compiler conforming to that ISO/IEC standard.

2509 Source code also refers to the input statements prepared for the following standard utilities: *awk*,
2510 *bc*, *ed*, *lex*, *localedef*, *make*, *sed*, and *yacc*.

2511 Source code can also refer to a collection of sources meeting any or all of these meanings.

2512 **Note:** The *awk*, *bc*, *ed*, *lex*, *localedef*, *make*, *sed*, and *yacc* utilities are defined in detail in the Shell and
2513 Utilities volume of POSIX.1-2008.

2514 3.354 Space Character (<space>)

2515 The character defined in the portable character set as <space>. The <space> character is a
2516 member of the **space** character class of the current locale, but represents the single character, and
2517 not all of the possible members of the class; see also [Section 3.436](#) (on page 103).

2518 3.355 Spawn

2519 A process creation primitive useful for systems that have difficulty with *fork()* and as an efficient
2520 replacement for *fork()/exec*.

2521 3.356 Special Built-In

2522 See *Built-In Utility* in [Section 3.83](#) (on page 46).

2523 **3.357 Special Parameter**

2524 In the shell command language, a parameter named by a single character from the following list:

2525 * @ # ? ! - \$ 0

2526 **Note:** For further information, see XCU [Section 2.5.2](#) (on page 2324).

2527 **3.358 Spin Lock**

2528 A synchronization object used to allow multiple threads to serialize their access to shared data.

2529 **3.359 Sporadic Server**

2530 A scheduling policy for threads and processes that reserves a certain amount of execution
2531 capacity for processing aperiodic events at a given priority level.

2532 **3.360 Standard Error**

2533 An output stream usually intended to be used for diagnostic messages.

2534 **3.361 Standard Input**

2535 An input stream usually intended to be used for primary data input.

2536 **3.362 Standard Output**

2537 An output stream usually intended to be used for primary data output.

2538 **3.363 Standard Utilities**

2539 The utilities described in the Shell and Utilities volume of POSIX.1-2008.

3.364 Stream

Appearing in lowercase, a stream is a file access object that allows access to an ordered sequence of characters, as described by the ISO C standard. Such objects can be created by the *fdopen()*, *fmemopen()*, *fopen()*, *open_memstream()*, or *popen()* functions, and are associated with a file descriptor. A stream provides the additional services of user-selectable buffering and formatted input and output; see also [Section 3.365](#).

Note: For further information, see XSH [Section 2.5](#) (on page 495).

The *fdopen()*, *fmemopen()*, *fopen()*, *open_memstream()*, and *popen()* functions are defined in detail in the System Interfaces volume of POSIX.1-2008.

3.365 STREAM

Appearing in uppercase, STREAM refers to a full-duplex connection between a process and an open device or pseudo-device. It optionally includes one or more intermediate processing modules that are interposed between the process end of the STREAM and the device driver (or pseudo-device driver) end of the STREAM; see also [Section 3.364](#).

Note: For further information, see XSH [Section 2.6](#) (on page 499).

3.366 STREAM End

The STREAM end is the driver end of the STREAM and is also known as the downstream end of the STREAM.

3.367 STREAM Head

The STREAM head is the beginning of the STREAM and is at the boundary between the system and the application process. This is also known as the upstream end of the STREAM.

3.368 STREAMS Multiplexor

A driver with multiple STREAMS connected to it. Multiplexing with STREAMS connected above is referred to as N-to-1, or “upper multiplexing”. Multiplexing with STREAMS connected below is referred to as 1-to-N or “lower multiplexing”.

3.369 String

A contiguous sequence of bytes terminated by and including the first null byte.

3.370 Subshell

A shell execution environment, distinguished from the main or current shell execution environment.

Note: For further information, see XCU [Section 2.12](#) (on page 2353).

3.371 Successfully Transferred

For a write operation to a regular file, when the system ensures that all data written is readable on any subsequent open of the file (even one that follows a system or power failure) in the absence of a failure of the physical storage medium.

For a read operation, when an image of the data on the physical storage medium is available to the requesting process.

3.372 Supplementary Group ID

An attribute of a process used in determining file access permissions. A process has up to {NGROUPS_MAX} supplementary group IDs in addition to the effective group ID. The supplementary group IDs of a process are set to the supplementary group IDs of the parent process when the process is created.

3.373 Suspended Job

A job that has received a SIGSTOP, SIGTSTP, SIGTTIN, or SIGTTOU signal that caused the process group to stop. A suspended job is a background job, but a background job is not necessarily a suspended job.

3.374 Symbolic Constant

An object-like macro defined with a constant value.

Unless stated otherwise, the following shall apply to every symbolic constant:

- It expands to a compile-time constant expression with an integer type.
- It may be defined as another type of constant—e.g., an enumeration constant—as well as being a macro.
- It need not be usable in **#if** preprocessing directives.

3.375 Symbolic Link

A type of file with the property that when the file is encountered during pathname resolution, a string stored by the file is used to modify the pathname resolution. The stored string has a length of {SYMLINK_MAX} bytes or fewer.

Note: Pathname Resolution is defined in detail in [Section 4.12](#) (on page 111).

3.376 Synchronized Input and Output

A determinism and robustness improvement mechanism to enhance the data input and output mechanisms, so that an application can ensure that the data being manipulated is physically present on secondary mass storage devices.

3.377 Synchronized I/O Completion

The state of an I/O operation that has either been successfully transferred or diagnosed as unsuccessful.

3.378 Synchronized I/O Data Integrity Completion

For read, when the operation has been completed or diagnosed if unsuccessful. The read is complete only when an image of the data has been successfully transferred to the requesting process. If there were any pending write requests affecting the data to be read at the time that the synchronized read operation was requested, these write requests are successfully transferred prior to reading the data.

For write, when the operation has been completed or diagnosed if unsuccessful. The write is complete only when the data specified in the write request is successfully transferred and all file system information required to retrieve the data is successfully transferred.

File attributes that are not necessary for data retrieval (access time, modification time, status change time) need not be successfully transferred prior to returning to the calling process.

3.379 Synchronized I/O File Integrity Completion

Identical to a synchronized I/O data integrity completion with the addition that all file attributes relative to the I/O operation (including access time, modification time, status change time) are successfully transferred prior to returning to the calling process.

3.380 Synchronized I/O Operation

An I/O operation performed on a file that provides the application assurance of the integrity of its data and files.

3.381 Synchronous I/O Operation

An I/O operation that causes the thread requesting the I/O to be blocked from further use of the processor until that I/O operation completes.

Note: A synchronous I/O operation does not imply synchronized I/O data integrity completion or synchronized I/O file integrity completion.

3.382 Synchronously-Generated Signal

A signal that is attributable to a specific thread.

For example, a thread executing an illegal instruction or touching invalid memory causes a synchronously-generated signal. Being synchronous is a property of how the signal was generated and not a property of the signal number.

3.383 System

An implementation of POSIX.1-2008.

3.384 System Boot

An unspecified sequence of events that may result in the loss of transitory data; that is, data that is not saved in permanent storage. For example, message queues, shared memory, semaphores, and processes.

3.385 System Clock

A clock with at least one second resolution that contains seconds since the Epoch.

3.386 System Console

A device that receives messages sent by the *syslog()* function, and the *fntmsg()* function when the MM_CONSOLE flag is set.

Note: The *syslog()* and *fntmsg()* functions are defined in detail in the System Interfaces volume of POSIX.1-2008.

3.387 System Crash

An interval initiated by an unspecified circumstance that causes all processes (possibly other than special system processes) to be terminated in an undefined manner, after which any

2649 changes to the state and contents of files created or written to by an application prior to the
2650 interval are undefined, except as required elsewhere in POSIX.1-2008.

2651 **3.388 System Databases**

2652 An implementation provides two system databases: the “group database” (see also [Section](#)
2653 [3.188](#), on page 63) and the “user database” (see also [Section 3.429](#), on page 101).

2654 **3.389 System Documentation**

2655 All documentation provided with an implementation except for the conformance document.
2656 Electronically distributed documents for an implementation are considered part of the system
2657 documentation.

2658 **3.390 System Process**

2659 An object other than a process executing an application, that is provided by the system and has a
2660 process ID.

2661 **3.391 System Reboot**

2662 See *System Boot* defined in [Section 3.384](#) (on page 95).

2663 **3.392 System Trace Event**

2664 A trace event that is generated by the implementation, in response either to a system-initiated
2665 action or to an application-requested action, except for a call to *posix_trace_event()*. When
2666 supported by the implementation, a system-initiated action generates a process-independent
2667 system trace event and an application-requested action generates a process-dependent system
2668 trace event. For a system trace event not defined by POSIX.1-2008, the associated trace event
2669 type identifier is derived from the implementation-defined name for this trace event, and the
2670 associated data is of implementation-defined content and length.

2671 **3.393 System-Wide**

2672 Pertaining to events occurring in all processes existing in an implementation at a given point in
2673 time.

3.394 Tab Character (<tab>)

A character that in the output stream indicates that printing or displaying should start at the next horizontal tabulation position on the current line. It is the character designated by ‘\t’ in the C language. If the current position is at or past the last defined horizontal tabulation position, the behavior is unspecified. It is unspecified whether this character is the exact sequence transmitted to an output device by the system to accomplish the tabulation.

3.395 Terminal (or Terminal Device)

A character special file that obeys the specifications of the general terminal interface.

Note: The General Terminal Interface is defined in detail in [Chapter 11](#) (on page 199).

3.396 Text Column

A roughly rectangular block of characters capable of being laid out side-by-side next to other text columns on an output page or terminal screen. The widths of text columns are measured in column positions.

3.397 Text File

A file that contains characters organized into zero or more lines. The lines do not contain NUL characters and none can exceed {LINE_MAX} bytes in length, including the <newline> character. Although POSIX.1-2008 does not distinguish between text files and binary files (see the ISO C standard), many utilities only produce predictable or meaningful output when operating on text files. The standard utilities that have such restrictions always specify “text files” in their STDIN or INPUT FILES sections.

3.398 Thread

A single flow of control within a process. Each thread has its own thread ID, scheduling priority and policy, *errno* value, floating point environment, thread-specific key/value bindings, and the required system resources to support a flow of control. Anything whose address may be determined by a thread, including but not limited to static variables, storage obtained via *malloc()*, directly addressable storage obtained through implementation-defined functions, and automatic variables, are accessible to all threads in the same process.

Note: The *malloc()* function is defined in detail in the System Interfaces volume of POSIX.1-2008.

3.399 Thread ID

Each thread in a process is uniquely identified during its lifetime by a value of type **pthread_t** called a thread ID.

3.400 Thread List

An ordered set of runnable threads that all have the same ordinal value for their priority.

The ordering of threads on the list is determined by a scheduling policy or policies. The set of thread lists includes all runnable threads in the system.

3.401 Thread-Safe

A thread-safe function can be safely invoked concurrently with other calls to the same function, or with calls to any other thread-safe functions, by multiple threads. Each function defined in the System Interfaces volume of POSIX.1-2008 is thread-safe unless explicitly stated otherwise. Examples are any “pure” function, a function which holds a mutex locked while it is accessing static storage, or objects shared among threads.

3.402 Thread-Specific Data Key

A process global handle of type `pthread_key_t` which is used for naming thread-specific data.

Although the same key value may be used by different threads, the values bound to the key by `pthread_setspecific()` and accessed by `pthread_getspecific()` are maintained on a per-thread basis and persist for the life of the calling thread.

Note: The `pthread_getspecific()` and `pthread_setspecific()` functions are defined in detail in the System Interfaces volume of POSIX.1-2008.

3.403 Tilde Character (<tilde>)

The character ‘~’.

3.404 Timeouts

A method of limiting the length of time an interface will block; see also [Section 3.76](#) (on page 44).

3.405 Timer

A mechanism that can notify a thread when the time as measured by a particular clock has reached or passed a specified value, or when a specified amount of time has passed.

3.406 Timer Overrun

A condition that occurs each time a timer, for which there is already an expiration signal queued to the process, expires.

2732 3.407 Token

2733 In the shell command language, a sequence of characters that the shell considers as a single unit
2734 when reading input. A token is either an operator or a word.

2735 **Note:** The rules for reading input are defined in detail in XCU [Section 2.3](#) (on page 2321).

2736 3.408 Trace Analyzer Process

2737 A process that extracts trace events from a trace stream to retrieve information about the
2738 behavior of an application.

2739 3.409 Trace Controller Process

2740 A process that creates a trace stream for tracing a process.

2741 3.410 Trace Event

2742 A data object that represents an action executed by the system, and that is recorded in a trace
2743 stream.

2744 3.411 Trace Event Type

2745 A data object type that defines a class of trace event.

2746 3.412 Trace Event Type Mapping

2747 A one-to-one mapping between trace event types and trace event names.

2748 3.413 Trace Filter

2749 A filter that allows the trace controller process to specify those trace event types that are to be
2750 ignored; that is, not generated.

2751 3.414 Trace Generation Version

2752 A data object that is an implementation-defined character string, generated by the trace system
2753 and describing the origin and version of the trace system.

2754 3.415 Trace Log

2755 The flushed image of a trace stream, if the trace stream is created with a trace log.

2756 3.416 Trace Point

2757 An action that may cause a trace event to be generated.

2758 3.417 Trace Stream

2759 An opaque object that contains trace events plus internal data needed to interpret those trace
2760 events.

2761 3.418 Trace Stream Identifier

2762 A handle to manage tracing operations in a trace stream.

2763 3.419 Trace System

2764 A system that allows both system and user trace events to be generated into a trace stream.
2765 These trace events can be retrieved later.

2766 3.420 Traced Process

2767 A process for which at least one trace stream has been created. A traced process is also called a
2768 target process.

2769 3.421 Tracing Status of a Trace Stream

2770 A status that describes the state of an active trace stream. The tracing status of a trace stream can
2771 be retrieved from the trace stream attributes. An active trace stream can be in one of two states:
2772 running or suspended.

2773 3.422 Typed Memory Name Space

2774 A system-wide name space that contains the names of the typed memory objects present in the
2775 system. It is configurable for a given implementation.

2776 3.423 Typed Memory Object

2777 A combination of a typed memory pool and a typed memory port. The entire contents of the
2778 pool are accessible from the port. The typed memory object is identified through a name that
2779 belongs to the typed memory name space.

2780 3.424 Typed Memory Pool

2781 An extent of memory with the same operational characteristics. Typed memory pools may be
2782 contained within each other.

2783 3.425 Typed Memory Port

2784 A hardware access path to one or more typed memory pools.

2785 3.426 Unbind

2786 Remove the association between a network address and an endpoint.

2787 3.427 Unit Data

2788 See *Datagram* in [Section 3.124](#) (on page 53).

2789 3.428 Upshifting

2790 The conversion of a lowercase character that has a single-character uppercase representation into
2791 this uppercase representation.

2792 3.429 User Database

2793 A system database that contains at least the following information for each user ID:

- 2794 • User name
- 2795 • Numerical user ID
- 2796 • Initial numerical group ID
- 2797 • Initial working directory
- 2798 • Initial user program

2799 The initial numerical group ID is used by the *newgrp* utility. Any other circumstances under
2800 which the initial values are operative are implementation-defined.

- 2801 If the initial user program field is null, an implementation-defined program is used.
- 2802 If the initial working directory field is null, the interpretation of that field is implementation-
- 2803 defined.
- 2804 **Note:** The *newgrp* utility is defined in detail in the Shell and Utilities volume of POSIX.1-2008.

2805 3.430 User ID

- 2806 A non-negative integer that is used to identify a system user. When the identity of a user is
- 2807 associated with a process, a user ID value is referred to as a real user ID, an effective user ID, or
- 2808 a saved set-user-ID.

2809 3.431 User Name

- 2810 A string that is used to identify a user; see also [Section 3.429](#) (on page 101). To be portable across
- 2811 systems conforming to POSIX.1-2008, the value is composed of characters from the portable
- 2812 filename character set. The <hyphen> character should not be used as the first character of a
- 2813 portable user name.

2814 3.432 User Trace Event

- 2815 A trace event that is generated explicitly by the application as a result of a call to
- 2816 *posix_trace_event()*.

2817 3.433 Utility

- 2818 A program, excluding special built-in utilities provided as part of the Shell Command Language,
- 2819 that can be called by name from a shell to perform a specific task, or related set of tasks.
- 2820 **Note:** For further information on special built-in utilities, see XCU [Section 2.14](#) (on page 2356).

3.434 Variable

In the shell command language, a named parameter.

Note: For further information, see XCU [Section 2.5](#) (on page 2323).

3.435 Vertical-Tab Character (<vertical-tab>)

A character that in the output stream indicates that printing should start at the next vertical tabulation position. It is the character designated by '`\v`' in the C language. If the current position is at or past the last defined vertical tabulation position, the behavior is unspecified. It is unspecified whether this character is the exact sequence transmitted to an output device by the system to accomplish the tabulation.

3.436 White Space

A sequence of one or more characters that belong to the **space** character class as defined via the `LC_CTYPE` category in the current locale.

In the POSIX locale, white space consists of one or more `<blank>` (`<space>` and `<tab>` characters), `<newline>`, `<carriage-return>`, `<form-feed>`, and `<vertical-tab>` characters.

3.437 Wide-Character Code (C Language)

An integer value corresponding to a single graphic symbol or control code.

Note: C Language Wide-Character Codes are defined in detail in [Section 6.3](#) (on page 129).

3.438 Wide-Character Input/Output Functions

The functions that perform wide-oriented input from streams or wide-oriented output to streams: `fgetwc()`, `fgetws()`, `fputwc()`, `fputws()`, `fwprintf()`, `fwscanf()`, `getwc()`, `getwchar()`, `putwc()`, `putwchar()`, `ungetwc()`, `vfwprintf()`, `vfwscanf()`, `vwpprintf()`, `vwscanf()`, `wprintf()`, and `wscanf()`.

Note: These functions are defined in detail in the System Interfaces volume of POSIX.1-2008.

3.439 Wide-Character String

A contiguous sequence of wide-character codes terminated by and including the first null wide-character code.

2846 3.440 Word

2847 In the shell command language, a token other than an operator. In some cases a word is also a
2848 portion of a word token: in the various forms of parameter expansion, such as `${name-word}`,
2849 and variable assignment, such as `name=word`, the word is the portion of the token depicted by
2850 *word*. The concept of a word is no longer applicable following word expansions—only fields
2851 remain.

2852 **Note:** For further information, see XCU [Section 2.6.2](#) (on page 2328) and [Section 2.6](#) (on page 2327).

2853 3.441 Working Directory (or Current Working Directory)

2854 A directory, associated with a process, that is used in pathname resolution for pathnames that do
2855 not begin with a <slash> character.

2856 3.442 Worldwide Portability Interface

2857 Functions for handling characters in a codeset-independent manner.

2858 3.443 Write

2859 To output characters to a file, such as standard output or standard error. Unless otherwise stated,
2860 standard output is the default output destination for all uses of the term “write”; see the
2861 distinction between display and write in [Section 3.133](#) (on page 54).

2862 3.444 XSI

2863 The X/Open System Interfaces (XSI) option is the core application programming interface for C
2864 and *sh* programming for systems conforming to the Single UNIX Specification. This is a
2865 superset of the mandatory requirements for conformance to POSIX.1-2008.

2866 3.445 XSI-Conformant

2867 A system which allows an application to be built using a set of services that are consistent across
2868 all systems that conform to POSIX.1-2008 and that support the XSI option.

2869 **Note:** See also [Chapter 2](#) (on page 15).

2870 3.446 Zombie Process

2871 A process that has terminated and that is deleted when its exit status has been reported to
2872 another process which is waiting for that process to terminate.

2873 3.447 ± 0

2874 The algebraic sign provides additional information about any variable that has the value zero
2875 when the representation allows the sign to be determined.

2876

2877

General Concepts

2878

For the purposes of POSIX.1-2008, the general concepts given in [Chapter 4](#) apply.

2879

Note: No shading to denote extensions or options occurs in this chapter. Where the terms and definitions given in this chapter are used elsewhere in text related to extensions and options, they are shaded as appropriate.

2880

2881

4.1 Concurrent Execution

2882

2883

2884

Functions that suspend the execution of the calling thread shall not cause the execution of other threads to be indefinitely suspended.

4.2 Directory Protection

2885

2886

2887

If a directory is writable and the mode bit S_ISVTX is set on the directory, a process may remove or rename files within that directory only if one or more of the following is true:

2888

- The effective user ID of the process is the same as that of the owner ID of the file.

2889

- The effective user ID of the process is the same as that of the owner ID of the directory.

2890

- The process has appropriate privileges.

2891

- Optionally, the file is writable by the process. Whether or not files that are writable by the process can be removed or renamed is implementation-defined.

2892

2893

If the S_ISVTX bit is set on a non-directory file, the behavior is unspecified.

4.3 Extended Security Controls

2894

2895

2896

2897

2898

An implementation may provide implementation-defined extended security controls (see [Section 3.160](#), on page 58). These permit an implementation to provide security mechanisms to implement different security policies than those described in POSIX.1-2008. These mechanisms shall not alter or override the defined semantics of any of the interfaces in POSIX.1-2008.

4.4 File Access Permissions

The standard file access control mechanism uses the file permission bits, as described below.

Implementations may provide *additional* or *alternate* file access control mechanisms, or both. An additional access control mechanism shall only further restrict the access permissions defined by the file permission bits. An alternate file access control mechanism shall:

- Specify file permission bits for the file owner class, file group class, and file other class of that file, corresponding to the access permissions.
- Be enabled only by explicit user action, on a per-file basis by the file owner or a user with appropriate privileges.
- Be disabled for a file after the file permission bits are changed for that file with *chmod()*. The disabling of the alternate mechanism need not disable any additional mechanisms supported by an implementation.

Whenever a process requests file access permission for read, write, or execute/search, if no additional mechanism denies access, access shall be determined as follows:

- If a process has appropriate privileges:
 - If read, write, or directory search permission is requested, access shall be granted.
 - If execute permission is requested, access shall be granted if execute permission is granted to at least one user by the file permission bits or by an alternate access control mechanism; otherwise, access shall be denied.
- Otherwise:
 - The file permission bits of a file contain read, write, and execute/search permissions for the file owner class, file group class, and file other class.
 - Access shall be granted if an alternate access control mechanism is not enabled and the requested access permission bit is set for the class (file owner class, file group class, or file other class) to which the process belongs, or if an alternate access control mechanism is enabled and it allows the requested access; otherwise, access shall be denied.

4.5 File Hierarchy

Files in the system are organized in a hierarchical structure in which all of the non-terminal nodes are directories and all of the terminal nodes are any other type of file. Since multiple directory entries may refer to the same file, the hierarchy is properly described as a “directed graph”.

4.6 Filenames

Uppercase and lowercase letters shall retain their unique identities between conforming implementations.

4.7 Filename Portability

For a filename to be portable across implementations conforming to POSIX.1-2008, it shall consist only of the portable filename character set as defined in [Section 3.278](#) (on page 78).

Note: Applications should avoid using filenames that have the <hyphen> character as the first character since this may cause problems when filenames are passed as command line arguments.

4.8 File Times Update

Each file has three distinct associated timestamps: the time of last data access, the time of last data modification, and the time the file status last changed. These values are returned in the file characteristics structure **struct stat**, as described in [<sys/stat.h>](#) (on page 391).

Each function or utility in POSIX.1-2008 that reads or writes data (even if the data does not change) or performs an operation to change file status (even if the file status does not change) indicates which of the appropriate timestamps shall be marked for update. If an implementation of such a function or utility marks for update one of these timestamps in a place or time not specified by POSIX.1-2008, this shall be documented, except that any changes caused by pathname resolution need not be documented. For the other functions or utilities in POSIX.1-2008 (those that are not explicitly required to read or write file data or change file status, but that in some implementations happen to do so), the effect is unspecified.

An implementation may update timestamps that are marked for update immediately, or it may update such timestamps periodically. At the point in time when an update occurs, any marked timestamps shall be set to the current time and the update marks shall be cleared. All timestamps that are marked for update shall be updated when the file ceases to be open by any process or before a *fstat()*, *fstatat()*, *fsync()*, *futimens()*, *lstat()*, *stat()*, *utime()*, *utimensat()*, or *utimes()* is successfully performed on the file. Other times at which updates are done are unspecified. Marks for update, and updates themselves, shall not be done for files on read-only file systems; see [Section 3.306](#) (on page 82).

The resolution of timestamps of files in a file system is implementation-defined, but shall be no coarser than one-second resolution. The three timestamps shall always have values that are supported by the file system. Whenever any of a file's timestamps are to be set to a value *V* according to the rules of the preceding paragraphs of this section, the implementation shall immediately set the timestamp to the greatest value supported by the file system that is not greater than *V*.

4.9 Host and Network Byte Orders

When data is transmitted over the network, it is sent as a sequence of octets (8-bit unsigned values). If an entity (such as an address or a port number) can be larger than 8 bits, it needs to be stored in several octets. The convention is that all such values are stored with 8 bits in each octet, and with the first (lowest-addressed) octet holding the most-significant bits. This is called “network byte order”.

Network byte order may not be convenient for processing actual values. For this, it is more sensible for values to be stored as ordinary integers. This is known as “host byte order”. In host byte order:

- The most significant bit might not be stored in the first byte in address order.
- Bits might not be allocated to bytes in any obvious order at all.

8-bit values stored in `uint8_t` objects do not require conversion to or from host byte order, as they have the same representation. 16 and 32-bit values can be converted using the `htonl()`, `htons()`, `ntohl()`, and `ntohs()` functions. When reading data that is to be converted to host byte order, it should either be received directly into a `uint16_t` or `uint32_t` object or should be copied from an array of bytes using `memcpy()` or similar. Passing the data through other types could cause the byte order to be changed. Similar considerations apply when sending data.

4.10 Measurement of Execution Time

The mechanism used to measure execution time shall be implementation-defined. The implementation shall also define to whom the CPU time that is consumed by interrupt handlers and system services on behalf of the operating system will be charged. See [Section 3.118](#) (on page 52).

4.11 Memory Synchronization

Applications shall ensure that access to any memory location by more than one thread of control (threads or processes) is restricted such that no thread of control can read or modify a memory location while another thread of control may be modifying it. Such access is restricted using functions that synchronize thread execution and also synchronize memory with respect to other threads. The following functions synchronize memory with respect to other threads:

<code>fork()</code>	<code>pthread_mutex_trylock()</code>	<code>pthread_rwlock_unlock()</code>
<code>pthread_barrier_wait()</code>	<code>pthread_mutex_unlock()</code>	<code>pthread_rwlock_wrlock()</code>
<code>pthread_cond_broadcast()</code>	<code>pthread_spin_lock()</code>	<code>sem_post()</code>
<code>pthread_cond_signal()</code>	<code>pthread_spin_trylock()</code>	<code>sem_timedwait()</code>
<code>pthread_cond_timedwait()</code>	<code>pthread_spin_unlock()</code>	<code>sem_trywait()</code>
<code>pthread_cond_wait()</code>	<code>pthread_rwlock_rdlock()</code>	<code>sem_wait()</code>
<code>pthread_create()</code>	<code>pthread_rwlock_timedrdlock()</code>	<code>semctl()</code>
<code>pthread_join()</code>	<code>pthread_rwlock_timedwrlock()</code>	<code>semop()</code>
<code>pthread_mutex_lock()</code>	<code>pthread_rwlock_tryrdlock()</code>	<code>wait()</code>
<code>pthread_mutex_timedlock()</code>	<code>pthread_rwlock_trywrlock()</code>	<code>waitpid()</code>

The `pthread_once()` function shall synchronize memory for the first call in each thread for a given `pthread_once_t` object.

The `pthread_mutex_lock()` function need not synchronize memory if the mutex type is `PTHREAD_MUTEX_RECURSIVE` and the calling thread already owns the mutex. The `pthread_mutex_unlock()` function need not synchronize memory if the mutex type is `PTHREAD_MUTEX_RECURSIVE` and the mutex has a lock count greater than one.

Unless explicitly stated otherwise, if one of the above functions returns an error, it is unspecified whether the invocation causes memory to be synchronized.

Applications may allow more than one thread of control to read a memory location simultaneously.

4.12 Pathname Resolution

Pathname resolution is performed for a process to resolve a pathname to a particular directory entry for a file in the file hierarchy. There may be multiple pathnames that resolve to the same directory entry, and multiple directory entries for the same file. When a process resolves a pathname of an existing directory entry, the entire pathname shall be resolved as described below. When a process resolves a pathname of a directory entry that is to be created immediately after the pathname is resolved, pathname resolution terminates when all components of the path prefix of the last component have been resolved. It is then the responsibility of the process to create the final component.

Each filename in the pathname is located in the directory specified by its predecessor (for example, in the pathname fragment `a/b`, file `b` is located in directory `a`). Pathname resolution shall fail if this cannot be accomplished. If the pathname begins with a `<slash>`, the predecessor of the first filename in the pathname shall be taken to be the root directory of the process (such pathnames are referred to as “absolute pathnames”). If the pathname does not begin with a `<slash>`, the predecessor of the first filename of the pathname shall be taken to be either the current working directory of the process or for certain interfaces the directory identified by a file descriptor passed to the interface (such pathnames are referred to as “relative pathnames”).

The interpretation of a pathname component is dependent on the value of `{NAME_MAX}` and `_POSIX_NO_TRUNC` associated with the path prefix of that component. If any pathname component is longer than `{NAME_MAX}`, the implementation shall consider this an error.

A pathname that contains at least one non-`<slash>` character and that ends with one or more trailing `<slash>` characters shall not be resolved successfully unless the last pathname component before the trailing `<slash>` characters names an existing directory or a directory entry that is to be created for a directory immediately after the pathname is resolved. Interfaces using pathname resolution may specify additional constraints⁶ when a pathname that does not name an existing directory contains at least one non-`<slash>` character and contains one or more trailing `<slash>` characters.

If a symbolic link is encountered during pathname resolution, the behavior shall depend on whether the pathname component is at the end of the pathname and on the function being performed. If all of the following are true, then pathname resolution is complete:

1. This is the last pathname component of the pathname.
2. The pathname has no trailing `<slash>`.

6. The only interfaces that further constrain pathnames in POSIX.1-2008 are the `rename()` and `renameat()` functions (see XSH [rename\(\)](#)) and the `mv` utility (see XCU [mv](#)).

3. The function is required to act on the symbolic link itself, or certain arguments direct that the function act on the symbolic link itself.

In all other cases, the system shall prefix the remaining pathname, if any, with the contents of the symbolic link. If the combined length exceeds {PATH_MAX}, and the implementation considers this to be an error, *errno* shall be set to [ENAMETOOLONG] and an error indication shall be returned. Otherwise, the resolved pathname shall be the resolution of the pathname just created. If the resulting pathname does not begin with a <slash>, the predecessor of the first filename of the pathname is taken to be the directory containing the symbolic link.

If the system detects a loop in the pathname resolution process, it shall set *errno* to [ELOOP] and return an error indication. The same may happen if during the resolution process more symbolic links were followed than the implementation allows. This implementation-defined limit shall not be smaller than {SYMLOOP_MAX}.

The special filename dot shall refer to the directory specified by its predecessor. The special filename dot-dot shall refer to the parent directory of its predecessor directory. As a special case, in the root directory, dot-dot may refer to the root directory itself.

A pathname consisting of a single <slash> shall resolve to the root directory of the process. A null pathname shall not be successfully resolved. If a pathname begins with two successive <slash> characters, the first component following the leading <slash> characters may be interpreted in an implementation-defined manner, although more than two leading <slash> characters shall be treated as a single <slash> character.

Pathname resolution for a given pathname shall yield the same results when used by any interface in POSIX.1-2008 as long as there are no changes to any files evaluated during pathname resolution for the given pathname between resolutions.

4.13 Process ID Reuse

A process group ID shall not be reused by the system until the process group lifetime ends.

A process ID shall not be reused by the system until the process lifetime ends. In addition, if there exists a process group whose process group ID is equal to that process ID, the process ID shall not be reused by the system until the process group lifetime ends. A process that is not a system process shall not have a process ID of 1.

4.14 Scheduling Policy

A scheduling policy affects process or thread ordering:

- When a process or thread is a running thread and it becomes a blocked thread
- When a process or thread is a running thread and it becomes a preempted thread
- When a process or thread is a blocked thread and it becomes a runnable thread
- When a running thread calls a function that can change the priority or scheduling policy of a process or thread
- In other scheduling policy-defined circumstances

Conforming implementations shall define the manner in which each of the scheduling policies may modify the priorities or otherwise affect the ordering of processes or threads at each of the

occurrences listed above. Additionally, conforming implementations shall define in what other circumstances and in what manner each scheduling policy may modify the priorities or affect the ordering of processes or threads.

4.15 Seconds Since the Epoch

A value that approximates the number of seconds that have elapsed since the Epoch. A Coordinated Universal Time name (specified in terms of seconds (*tm_sec*), minutes (*tm_min*), hours (*tm_hour*), days since January 1 of the year (*tm_yday*), and calendar year minus 1900 (*tm_year*)) is related to a time represented as seconds since the Epoch, according to the expression below.

If the year is <1970 or the value is negative, the relationship is undefined. If the year is ≥1970 and the value is non-negative, the value is related to a Coordinated Universal Time name according to the C-language expression, where *tm_sec*, *tm_min*, *tm_hour*, *tm_yday*, and *tm_year* are all integer types:

$$tm\_sec + tm\_min*60 + tm\_hour*3600 + tm\_yday*86400 + \\ (tm\_year-70)*31536000 + ((tm\_year-69)/4)*86400 - \\ ((tm\_year-1)/100)*86400 + ((tm\_year+299)/400)*86400$$

The relationship between the actual time of day and the current value for seconds since the Epoch is unspecified.

How any changes to the value of seconds since the Epoch are made to align to a desired relationship with the current actual time is implementation-defined. As represented in seconds since the Epoch, each and every day shall be accounted for by exactly 86 400 seconds.

Note: The last three terms of the expression add in a day for each year that follows a leap year starting with the first leap year since the Epoch. The first term adds a day every 4 years starting in 1973, the second subtracts a day back out every 100 years starting in 2001, and the third adds a day back in every 400 years starting in 2001. The divisions in the formula are integer divisions; that is, the remainder is discarded leaving only the integer quotient.

4.16 Semaphore

A minimum synchronization primitive to serve as a basis for more complex synchronization mechanisms to be defined by the application program.

For the semaphores associated with the Semaphores option, a semaphore is represented as a shareable resource that has a non-negative integer value. When the value is zero, there is a (possibly empty) set of threads awaiting the availability of the semaphore.

For the semaphores associated with the X/Open System Interfaces (XSI) option, a semaphore is a positive integer (0 through 32767). The *semget()* function can be called to create a set or array of semaphores. A semaphore set can contain one or more semaphores up to an implementation-defined value.

Semaphore Lock Operation

An operation that is applied to a semaphore. If, prior to the operation, the value of the semaphore is zero, the semaphore lock operation shall cause the calling thread to be blocked and added to the set of threads awaiting the semaphore; otherwise, the value shall be decremented.

Semaphore Unlock Operation

An operation that is applied to a semaphore. If, prior to the operation, there are any threads in the set of threads awaiting the semaphore, then some thread from that set shall be removed from the set and becomes unblocked; otherwise, the semaphore value shall be incremented.

4.17 Thread-Safety

Refer to XSH [Section 2.9](#) (on page 512).

4.18 Tracing

The trace system allows a traced process to have a selection of events created for it. Traces consist of streams of trace event types.

A trace event type is identified on the one hand by a trace event type name, also referenced as a trace event name, and on the other hand by a trace event type identifier. A trace event name is a human-readable string. A trace event type identifier is an opaque identifier used by the trace system. There shall be a one-to-one relationship between trace event type identifiers and trace event names for a given trace stream and also for a given traced process. The trace event type identifier shall be generated automatically from a trace event name by the trace system either when a trace controller process invokes *posix_trace_trid_eventid_open()* or when an instrumented application process invokes *posix_trace_eventid_open()*. Trace event type identifiers are used to filter trace event types, to allow interpretation of user data, and to identify the kind of trace point that generated a trace event.

Each trace event shall be of a particular trace event type, and associated with a trace event type identifier. The execution of a trace point shall generate a trace event if a trace stream has been created and started for the process that executed the trace point and if the corresponding trace event type identifier is not ignored by filtering.

A generated trace event shall be recorded in a trace stream, and optionally also in a trace log if a trace log is associated with the trace stream, except that:

- For a trace stream, if no resources are available for the event, the event is lost.
- For a trace log, if no resources are available for the event, or a flush operation does not succeed, the event is lost.

A trace event recorded in an active trace stream may be retrieved by an application having appropriate privileges.

A trace event recorded in a trace log may be retrieved by an application having appropriate privileges after opening the trace log as a pre-recorded trace stream, with the function *posix_trace_open()*.

When a trace event is reported it is possible to retrieve the following:

- A trace event type identifier
- A timestamp
- The process ID of the traced process, if the trace event is process-dependent
- Any optional trace event data including its length
- If the Threads option is supported, the thread ID, if the trace event is process-dependent
- The program address at which the trace point was invoked

Trace events may be mapped from trace event types to trace event names. One such mapping shall be associated with each trace stream. An active trace stream is associated with a traced process, and also with its children if the Trace Inherit option is supported and also the inheritance policy is set to `_POSIX_TRACE_INHERIT`. Therefore each traced process has a mapping of the trace event names to trace event type identifiers that have been defined for that process.

Traces can be recorded into either trace streams or trace logs.

The implementation and format of a trace stream are unspecified. A trace stream need not be and generally is not persistent. A trace stream may be either active or pre-recorded:

- An active trace stream is a trace stream that has been created and has not yet been shut down. It can be of one of the two following classes:
 1. An active trace stream without a trace log that was created with the `posix_trace_create()` function
 2. If the Trace Log option is supported, an active trace stream with a trace log that was created with the `posix_trace_create_withlog()` function
- A pre-recorded trace stream is a trace stream that was opened from a trace log object using the `posix_trace_open()` function.

An active trace stream can loop. This behavior means that when the resources allocated by the trace system for the trace stream are exhausted, the trace system reuses the resources associated with the oldest recorded trace events to record new trace events.

If the Trace Log option is supported, an active trace stream with a trace log can be flushed. This operation causes the trace system to write trace events from the trace stream to the associated trace log, following the defined policies or using an explicit function call. After this operation, the trace system may reuse the resources associated with the flushed trace events.

An active trace stream with or without a trace log can be cleared. This operation shall cause all the resources associated with this trace stream to be reinitialized. The trace stream shall behave as if it was returning from its creation, except that the mapping of trace event type identifiers to trace event names shall not be cleared. If a trace log was associated with this trace stream, the trace log shall also be reinitialized.

A trace log shall be recorded when the `posix_trace_shutdown()` operation is invoked or during tracing, depending on the tracing strategy which is defined by a log policy. After the trace stream has been shut down, the trace information can be retrieved from the associated trace log using the same interface used to retrieve information from an active trace stream.

For a traced process, if the Trace Inherit option is supported and the trace stream's inheritance attribute is `_POSIX_TRACE_INHERIT`, the initial targeted traced process shall be traced together with all of its future children. The `posix_pid` member of each trace event in a trace stream shall be the process ID of the traced process.

Each trace point may be an implementation-defined action such as a context switch, or an application-programmed action such as a call to a specific operating system service (for example, *fork()*) or a call to *posix_trace_event()*.

Trace points may be filtered. The operation of the filter is to filter out (ignore) selected trace events. By default, no trace events are filtered.

The results of the tracing operations can be analyzed and monitored by a trace controller process or a trace analyzer process.

Only the trace controller process has control of the trace stream it has created. The control of the operation of a trace stream is done using its corresponding trace stream identifier. The trace controller process is able to:

- Initialize the attributes of a trace stream
- Create the trace stream
- Start and stop tracing
- Know the mapping of the traced process
- If the Trace Event Filter option is supported, filter the type of trace events to be recorded
- Shut the trace stream down

A traced process may also be a trace controller process. Only the trace controller process can control its trace stream(s). A trace stream created by a trace controller process shall be shut down if its controller process terminates or executes another file.

A trace controller process may also be a trace analyzer process. Trace analysis can be done concurrently with the traced process or can be done off-line, in the same or in a different platform.

4.19 Treatment of Error Conditions for Mathematical Functions

For all the functions in the **<math.h>** header, an application wishing to check for error situations should set *errno* to 0 and call *feclearexcept*(FE_ALL_EXCEPT) before calling the function. On return, if *errno* is non-zero or *fetestexcept*(FE_INVALID | FE_DIVBYZERO | FE_OVERFLOW | FE_UNDERFLOW) is non-zero, an error has occurred.

The following error conditions are defined for all functions in the **<math.h>** header.

4.19.1 Domain Error

A “domain error” shall occur if an input argument is outside the domain over which the mathematical function is defined. The description of each function lists any required domain errors; an implementation may define additional domain errors, provided that such errors are consistent with the mathematical definition of the function.

On a domain error, the function shall return an implementation-defined value; if the integer expression (math_errhandling & MATH_ERRNO) is non-zero, *errno* shall be set to [EDOM]; if the integer expression (math_errhandling & MATH_ERREXCEPT) is non-zero, the “invalid” floating-point exception shall be raised.

4.19.2 Pole Error

A “pole error” occurs if the mathematical result of the function is an exact infinity (for example, $\log(0.0)$).

On a pole error, the function shall return the value of the macro `HUGE_VAL`, `HUGE_VALF`, or `HUGE_VALL` according to the return type, with the same sign as the correct value of the function; if the integer expression `(math_errhandling & MATH_ERRNO)` is non-zero, *errno* shall be set to `[ERANGE]`; if the integer expression `(math_errhandling & MATH_ERREXCEPT)` is non-zero, the “divide-by-zero” floating-point exception shall be raised.

4.19.3 Range Error

A “range error” shall occur if the finite mathematical result of the function cannot be represented in an object of the specified type, due to extreme magnitude.

4.19.3.1 Result Overflows

A floating result overflows if the magnitude of the mathematical result is finite but so large that the mathematical result cannot be represented without extraordinary roundoff error in an object of the specified type. If a floating result overflows and default rounding is in effect, then the function shall return the value of the macro `HUGE_VAL`, `HUGE_VALF`, or `HUGE_VALL` according to the return type, with the same sign as the correct value of the function; if the integer expression `(math_errhandling & MATH_ERRNO)` is non-zero, *errno* shall be set to `[ERANGE]`; if the integer expression `(math_errhandling & MATH_ERREXCEPT)` is non-zero, the “overflow” floating-point exception shall be raised.

4.19.3.2 Result Underflows

The result underflows if the magnitude of the mathematical result is so small that the mathematical result cannot be represented, without extraordinary roundoff error, in an object of the specified type. If the result underflows, the function shall return an implementation-defined value whose magnitude is no greater than the smallest normalized positive number in the specified type; if the integer expression `(math_errhandling & MATH_ERRNO)` is non-zero, whether *errno* is set to `[ERANGE]` is implementation-defined; if the integer expression `(math_errhandling & MATH_ERREXCEPT)` is non-zero, whether the “underflow” floating-point exception is raised is implementation-defined.

4.20 Treatment of NaN Arguments for the Mathematical Functions

For functions called with a NaN argument, no errors shall occur and a NaN shall be returned, except where stated otherwise.

If a function with one or more NaN arguments returns a NaN result, the result should be the same as one of the NaN arguments (after possible type conversion), except perhaps for the sign.

On implementations that support the IEC 60559:1989 standard floating point, functions with signaling NaN argument(s) shall be treated as if the function were called with an argument that is a required domain error and shall return a quiet NaN result, except where stated otherwise.

Note: The function might never see the signaling NaN, since it might trigger when the arguments are evaluated during the function call.

On implementations that support the IEC 60559:1989 standard floating point, for those functions that do not have a documented domain error, the following shall apply:

These functions shall fail if:

Domain Error Any argument is a signaling NaN.

Either, the integer expression (`math_errhandling & MATH_ERRNO`) is non-zero and *errno* shall be set to [EDOM], or the integer expression (`math_errhandling & MATH_ERREXCEPT`) is non-zero and the invalid floating-point exception shall be raised.

4.21 Utility

A utility program shall be either an executable file, such as might be produced by a compiler or linker system from computer source code, or a file of shell source code, directly interpreted by the shell. The program may have been produced by the user, provided by the system implementor, or acquired from an independent distributor.

The system may implement certain utilities as shell functions (see XCU [Section 2.9.5](#), on page 2346) or built-in utilities, but only an application that is aware of the command search order (as described in XCU [Section 2.9.1.1](#), on page 2339) or of performance characteristics can discern differences between the behavior of such a function or built-in utility and that of an executable file.

4.22 Variable Assignment

In the shell command language, a word consisting of the following parts:

varname=value

When used in a context where assignment is defined to occur and at no other time, the *value* (representing a word or field) shall be assigned as the value of the variable denoted by *varname*.

Note: For further information, see XCU [Section 2.9.1](#) (on page 2338).

The *varname* and *value* parts shall meet the requirements for a name and a word, respectively, except that they are delimited by the embedded unquoted <equals-sign>, in addition to other delimiters.

3306 **Note:** Additional delimiters are described in XCU [Section 2.3](#) (on page 2321).

3307 When a variable assignment is done, the variable shall be created if it did not already exist. If

3308 *value* is not specified, the variable shall be given a null value.

3309 **Note:** An alternative form of variable assignment:

3310 *symbol=value*

3311 (where *symbol* is a valid word delimited by an <equals-sign>, but not a valid name) produces

3312 unspecified results. The form *symbol=value* is used by the KornShell *name[expression]=value*

3313 syntax.

The STDIN, STDOUT, STDERR, INPUT FILES, and OUTPUT FILES sections of the utility descriptions use a syntax to describe the data organization within the files, when that organization is not otherwise obvious. The syntax is similar to that used by the System Interfaces volume of POSIX.1-2008 `printf()` function, as described in this chapter. When used in STDIN or INPUT FILES sections of the utility descriptions, this syntax describes the format that could have been used to write the text to be read, not a format that could be used by the System Interfaces volume of POSIX.1-2008 `scanf()` function to read the input file.

The description of an individual record is as follows:

"<format>", [*<arg1>*, *<arg2>*, . . . , *<argn>*]

The *format* is a character string that contains three types of objects defined below:

1. *Characters* that are not “escape sequences” or “conversion specifications”, as described below, shall be copied to the output.
2. *Escape Sequences* represent non-graphic characters and the escape character (<backslash>).
3. *Conversion Specifications* specify the output format of each argument; see below.

The following characters have the following special meaning in the format string:

' ' (An empty character position.) Represents one or more <blank> characters.

Δ Represents exactly one <space> character.

Table 5-1 lists escape sequences and associated actions on display devices capable of the action.

Table 5-1 Escape Sequences and Associated Actions

Escape Sequence	Represents Character	Terminal Action
\\	<backslash>	Print the <backslash> character.
\a	<alert>	Attempt to alert the user through audible or visible notification.
\b	<backspace>	Move the printing position to one column before the current position, unless the current position is the start of a line.
\f	<form-feed>	Move the printing position to the initial printing position of the next logical page.
\n	<newline>	Move the printing position to the start of the next line.
\r	<carriage-return>	Move the printing position to the start of the current line.
\t	<tab>	Move the printing position to the next tab position on the current line. If there are no more tab positions remaining on the line, the behavior is undefined.
\v	<vertical-tab>	Move the printing position to the start of the next <vertical-tab> position. If there are no more <vertical-tab> positions left on the page, the behavior is undefined.

3351	Each conversion specification is introduced by the <percent-sign> character ('%'). After the	
3352	character '%', the following shall appear in sequence:	
3353	<i>flags</i>	Zero or more <i>flags</i> , in any order, that modify the meaning of the conversion
3354		specification.
3355	<i>field width</i>	An optional string of decimal digits to specify a minimum field width. For an
3356		output field, if the converted value has fewer bytes than the field width, it shall be
3357		padded on the left (or right, if the left-adjustment flag ('-'), described below, has
3358		been given) to the field width.
3359	<i>precision</i>	Gives the minimum number of digits to appear for the d, o, i, u, x, or X conversion
3360		specifiers (the field is padded with leading zeros), the number of digits to appear
3361		after the radix character for the e and f conversion specifiers, the maximum
3362		number of significant digits for the g conversion specifier; or the maximum
3363		number of bytes to be written from a string in the s conversion specifier. The
3364		precision shall take the form of a <period> ('.') followed by a decimal digit
3365		string; a null digit string is treated as zero.
3366	<i>conversion specifier characters</i>	
3367		A conversion specifier character (see below) that indicates the type of conversion
3368		to be applied.
3369	The <i>flag</i> characters and their meanings are:	
3370	-	The result of the conversion shall be left-justified within the field.
3371	+	The result of a signed conversion shall always begin with a sign ('+' or '-').
3372	<space>	If the first character of a signed conversion is not a sign, a <space> shall be
3373		prefixed to the result. This means that if the <space> and '+' flags both appear,
3374		the <space> flag shall be ignored.
3375	#	The value shall be converted to an alternative form. For c, d, i, u, and s
3376		conversion specifiers, the behavior is undefined. For the o conversion specifier, it
3377		shall increase the precision to force the first digit of the result to be a zero. For x or
3378		X conversion specifiers, a non-zero result has 0x or 0X prefixed to it, respectively.
3379		For a, A, e, E, f, F, g, and G conversion specifiers, the result shall always contain a
3380		radix character, even if no digits follow the radix character. For g and G conversion
3381		specifiers, trailing zeros shall not be removed from the result as they usually are.
3382	0	For a, A, d, e, E, f, F, g, G, i, o, u, x, and X conversion specifiers, leading zeros
3383		(following any indication of sign or base) shall be used to pad to the field width
3384		rather than performing space padding, except when converting an infinity or NaN.
3385		If the '0' and '-' flags both appear, the '0' flag shall be ignored. For d, i, o, u,
3386		x, and X conversion specifiers, if a precision is specified, the '0' flag shall be
3387		ignored. For other conversion specifiers, the behavior is undefined.
3388	Each conversion specifier character shall result in fetching zero or more arguments. The results	
3389	are undefined if there are insufficient arguments for the format. If the format is exhausted while	
3390	arguments remain, the excess arguments shall be ignored.	
3391	The conversion specifiers and their meanings are:	
3392	a,A	The floating-point number argument representing a floating-point number shall be
3393		converted in the style "[-] 0xh.hhhhp±d", where there is one hexadecimal digit
3394		(which shall be non-zero if the argument is a normalized floating-point number
3395		and is otherwise unspecified) before the decimal-point character and the number
3396		of hexadecimal digits after it is equal to the precision; if the precision is missing

3397		and FLT_RADIX is a power of 2, then the precision shall be sufficient for an exact
3398		representation of the value; if the precision is missing and FLT_RADIX is not a
3399		power of 2, then the precision shall be sufficient to distinguish different floating-
3400		point values in the internal representation used by the utility, except that trailing
3401		zeros may be omitted; if the precision is zero and the # flag is not specified, no
3402		decimal-point character shall appear. The letters "abcdef" shall be used for a
3403		conversion and the letters "ABCDEF" for A conversion. The A conversion specifier
3404		produces a number with X and P instead of x and p. The exponent shall always
3405		contain at least one digit, and only as many more digits as necessary to represent
3406		the decimal exponent of 2. If the value is zero, the exponent shall be zero. A
3407		floating-point number argument representing an infinity or NaN shall be
3408		converted in the style of an f or F conversion specifier.
3409	d,i,o,u,x,X	The integer argument shall be written as signed decimal (d or i), unsigned octal
3410		(o), unsigned decimal (u), or unsigned hexadecimal notation (x and X). The d and
3411		i specifiers shall convert to signed decimal in the style "[-]dddd". The x
3412		conversion specifier shall use the numbers and letters "0123456789abcdef" and
3413		the X conversion specifier shall use the numbers and letters
3414		"0123456789ABCDEF". The <i>precision</i> component of the argument shall specify
3415		the minimum number of digits to appear. If the value being converted can be
3416		represented in fewer digits than the specified minimum, it shall be expanded with
3417		leading zeros. The default precision shall be 1. The result of converting a zero
3418		value with a precision of 0 shall be no characters. If both the field width and
3419		precision are omitted, the implementation may precede, follow, or precede and
3420		follow numeric arguments of types d, i, and u with <blank> characters; arguments
3421		of type o (octal) may be preceded with leading zeros.
3422	f,F	The floating-point number argument shall be written in decimal notation in the
3423		style [-]ddd.ddd, where the number of digits after the radix character (shown here
3424		as a decimal point) shall be equal to the <i>precision</i> specification. The LC_NUMERIC
3425		locale category shall determine the radix character to use in this format. If the
3426		<i>precision</i> is omitted from the argument, six digits shall be written after the radix
3427		character; if the <i>precision</i> is explicitly 0, no radix character shall appear.
3428		A floating-point number argument representing an infinity shall be converted in
3429		one of the styles "[-]inf" or "[-]infinity"; which style is implementation-
3430		defined. A floating-point number argument representing a NaN shall be converted
3431		in one of the styles "[-]nan(<i>n-char-sequence</i>)" or "[-]nan"; which style,
3432		and the meaning of any <i>n-char-sequence</i> , is implementation-defined. The F
3433		conversion specifier produces "INF", "INFINITY", or "NAN" instead of "inf",
3434		"infinity", or "nan", respectively.
3435	e,E	The floating-point number argument shall be written in the style [-]d.ddde±dd (the
3436		symbol '±' indicates either a <plus-sign> or minus-sign), where there is one digit
3437		before the radix character (shown here as a decimal point) and the number of
3438		digits after it is equal to the precision. The LC_NUMERIC locale category shall
3439		determine the radix character to use in this format. When the precision is missing,
3440		six digits shall be written after the radix character; if the precision is 0, no radix
3441		character shall appear. The E conversion specifier shall produce a number with E
3442		instead of e introducing the exponent. The exponent shall always contain at least
3443		two digits. However, if the value to be written requires an exponent greater than
3444		two digits, additional exponent digits shall be written as necessary.
3445		A floating-point number argument representing an infinity or NaN shall be
3446		converted in the style of an f or F conversion specifier.

3447	g,G	The floating-point number argument shall be written in style <code>f</code> or <code>e</code> (or in style <code>F</code> or <code>E</code> in the case of a <code>G</code> conversion specifier), with the precision specifying the number of significant digits. The style used depends on the value converted: style <code>e</code> (or <code>E</code>) shall be used only if the exponent resulting from the conversion is less than -4 or greater than or equal to the precision. Trailing zeros are removed from the result. A radix character shall appear only if it is followed by a digit.
3448		
3449		
3450		
3451		
3452		
3453		A floating-point number argument representing an infinity or NaN shall be converted in the style of an <code>f</code> or <code>F</code> conversion specifier.
3454		
3455	c	The single-byte character argument shall be written.
3456	s	The argument shall be taken to be a string and bytes from the string shall be written until the end of the string or the number of bytes indicated by the <i>precision</i> specification of the argument is reached. If the precision is omitted from the argument, it shall be taken to be infinite, so all bytes up to the end of the string shall be written.
3457		
3458		
3459		
3460		
3461	%	Write a ' % ' character; no argument is converted.
3462		In no case does a nonexistent or insufficient field width cause truncation of a field; if the result of a conversion is wider than the field width, the field is simply expanded to contain the conversion result. The term “field width” should not be confused with the term “precision” used in the description of <code>%s</code> .
3463		
3464		
3465		

Examples

To represent the output of a program that prints a date and time in the form Sunday, July 3, 10:02, where *weekday* and *month* are strings:

```
"%s,%s,%d,%d:%.2d\n" <weekday>, <month>, <day>, <hour>, <min>
```

To show ' π ' written to 5 decimal places:

```
"pi=%f\n", <value of  $\pi$ >
```

To show an input file format consisting of five <colon>-separated fields:

```
"%s:%s:%s:%s:%s\n", <arg1>, <arg2>, <arg3>, <arg4>, <arg5>
```

6.1 Portable Character Set

Conforming implementations shall support one or more coded character sets. Each supported locale shall include the *portable character set*, which is the set of symbolic names for characters in [Table 6-1](#). This is used to describe characters within the text of POSIX.1-2008. The first eight entries in [Table 6-1](#) are defined in the ISO/IEC 6429: 1992 standard and the rest of the characters are defined in the ISO/IEC 10646-1: 2000 standard.

Table 6-1 Portable Character Set

Symbolic Name	Glyph	UCS	Description
<NUL>		<U0000>	NULL (NUL)
<alert>		<U0007>	BELL (BEL)
<backspace>		<U0008>	BACKSPACE (BS)
<tab>		<U0009>	CHARACTER TABULATION (HT)
<carriage-return>		<U000D>	CARRIAGE RETURN (CR)
<newline>		<U000A>	LINE FEED (LF)
<vertical-tab>		<U000B>	LINE TABULATION (VT)
<form-feed>		<U000C>	FORM FEED (FF)
<space>		<U0020>	SPACE
<exclamation-mark>	!	<U0021>	EXCLAMATION MARK
<quotation-mark>	"	<U0022>	QUOTATION MARK
<number-sign>	#	<U0023>	NUMBER SIGN
<dollar-sign>	\$	<U0024>	DOLLAR SIGN
<percent-sign>	%	<U0025>	PERCENT SIGN
<ampersand>	&	<U0026>	AMPERSAND
<apostrophe>	'	<U0027>	APOSTROPHE
<left-parenthesis>	(	<U0028>	LEFT PARENTHESIS
<right-parenthesis>	)	<U0029>	RIGHT PARENTHESIS
<asterisk>	*	<U002A>	ASTERISK
<plus-sign>	+	<U002B>	PLUS SIGN
<comma>	,	<U002C>	COMMA
<hyphen-minus>	–	<U002D>	HYPHEN-MINUS
<hyphen>	–	<U002D>	HYPHEN-MINUS
<full-stop>	.	<U002E>	FULL STOP
<period>	.	<U002E>	FULL STOP
<slash>	/	<U002F>	SOLIDUS
<solidus>	/	<U002F>	SOLIDUS
<zero>	0	<U0030>	DIGIT ZERO
<one>	1	<U0031>	DIGIT ONE
<two>	2	<U0032>	DIGIT TWO

	Symbolic Name	Glyph	UCS	Description
3514				
3515	<three>	3	<U0033>	DIGIT THREE
3516	<four>	4	<U0034>	DIGIT FOUR
3517	<five>	5	<U0035>	DIGIT FIVE
3518	<six>	6	<U0036>	DIGIT SIX
3519	<seven>	7	<U0037>	DIGIT SEVEN
3520	<eight>	8	<U0038>	DIGIT EIGHT
3521	<nine>	9	<U0039>	DIGIT NINE
3522	<colon>	:	<U003A>	COLON
3523	<semicolon>	;	<U003B>	SEMICOLON
3524	<less-than-sign>	<	<U003C>	LESS-THAN SIGN
3525	<equals-sign>	=	<U003D>	EQUALS SIGN
3526	<greater-than-sign>	>	<U003E>	GREATER-THAN SIGN
3527	<question-mark>	?	<U003F>	QUESTION MARK
3528	<commercial-at>	@	<U0040>	COMMERCIAL AT
3529	<A>	A	<U0041>	LATIN CAPITAL LETTER A
3530		B	<U0042>	LATIN CAPITAL LETTER B
3531	<C>	C	<U0043>	LATIN CAPITAL LETTER C
3532	<D>	D	<U0044>	LATIN CAPITAL LETTER D
3533	<E>	E	<U0045>	LATIN CAPITAL LETTER E
3534	<F>	F	<U0046>	LATIN CAPITAL LETTER F
3535	<G>	G	<U0047>	LATIN CAPITAL LETTER G
3536	<H>	H	<U0048>	LATIN CAPITAL LETTER H
3537	<I>	I	<U0049>	LATIN CAPITAL LETTER I
3538	<J>	J	<U004A>	LATIN CAPITAL LETTER J
3539	<K>	K	<U004B>	LATIN CAPITAL LETTER K
3540	<L>	L	<U004C>	LATIN CAPITAL LETTER L
3541	<M>	M	<U004D>	LATIN CAPITAL LETTER M
3542	<N>	N	<U004E>	LATIN CAPITAL LETTER N
3543	<O>	O	<U004F>	LATIN CAPITAL LETTER O
3544	<P>	P	<U0050>	LATIN CAPITAL LETTER P
3545	<Q>	Q	<U0051>	LATIN CAPITAL LETTER Q
3546	<R>	R	<U0052>	LATIN CAPITAL LETTER R
3547	<S>	S	<U0053>	LATIN CAPITAL LETTER S
3548	<T>	T	<U0054>	LATIN CAPITAL LETTER T
3549	<U>	U	<U0055>	LATIN CAPITAL LETTER U
3550	<V>	V	<U0056>	LATIN CAPITAL LETTER V
3551	<W>	W	<U0057>	LATIN CAPITAL LETTER W
3552	<X>	X	<U0058>	LATIN CAPITAL LETTER X
3553	<Y>	Y	<U0059>	LATIN CAPITAL LETTER Y
3554	<Z>	Z	<U005A>	LATIN CAPITAL LETTER Z
3555	<left-square-bracket>	[	<U005B>	LEFT SQUARE BRACKET
3556	<backslash>	\	<U005C>	REVERSE SOLIDUS
3557	<reverse-solidus>	\	<U005C>	REVERSE SOLIDUS
3558	<right-square-bracket>	]	<U005D>	RIGHT SQUARE BRACKET
3559	<circumflex-accent>	^	<U005E>	CIRCUMFLEX ACCENT
3560	<circumflex>	^	<U005E>	CIRCUMFLEX ACCENT
3561	<low-line>	—	<U005F>	LOW LINE
3562	<underscore>	—	<U005F>	LOW LINE
3563	<grave-accent>	`	<U0060>	GRAVE ACCENT
3564	<a>	a	<U0061>	LATIN SMALL LETTER A
3565		b	<U0062>	LATIN SMALL LETTER B

Symbolic Name	Glyph	UCS	Description
<c>	c	<U0063>	LATIN SMALL LETTER C
<d>	d	<U0064>	LATIN SMALL LETTER D
<e>	e	<U0065>	LATIN SMALL LETTER E
<f>	f	<U0066>	LATIN SMALL LETTER F
<g>	g	<U0067>	LATIN SMALL LETTER G
<h>	h	<U0068>	LATIN SMALL LETTER H
<i>	i	<U0069>	LATIN SMALL LETTER I
<j>	j	<U006A>	LATIN SMALL LETTER J
<k>	k	<U006B>	LATIN SMALL LETTER K
<l>	l	<U006C>	LATIN SMALL LETTER L
<m>	m	<U006D>	LATIN SMALL LETTER M
<n>	n	<U006E>	LATIN SMALL LETTER N
<o>	o	<U006F>	LATIN SMALL LETTER O
<p>	p	<U0070>	LATIN SMALL LETTER P
<q>	q	<U0071>	LATIN SMALL LETTER Q
<r>	r	<U0072>	LATIN SMALL LETTER R
<s>	s	<U0073>	LATIN SMALL LETTER S
<t>	t	<U0074>	LATIN SMALL LETTER T
<u>	u	<U0075>	LATIN SMALL LETTER U
<v>	v	<U0076>	LATIN SMALL LETTER V
<w>	w	<U0077>	LATIN SMALL LETTER W
<x>	x	<U0078>	LATIN SMALL LETTER X
<y>	y	<U0079>	LATIN SMALL LETTER Y
<z>	z	<U007A>	LATIN SMALL LETTER Z
<left-brace>	{	<U007B>	LEFT CURLY BRACKET
<left-curly-bracket>	{	<U007B>	LEFT CURLY BRACKET
<vertical-line>		<U007C>	VERTICAL LINE
<right-brace>	}	<U007D>	RIGHT CURLY BRACKET
<right-curly-bracket>	}	<U007D>	RIGHT CURLY BRACKET
<tilde>	~	<U007E>	TILDE

POSIX.1-2008 uses character names other than the above, but only in an informative way; for example, in examples to illustrate the use of characters beyond the portable character set with the facilities of POSIX.1-2008.

Table 6-1 (on page 125) defines the characters in the portable character set and the corresponding symbolic character names used to identify each character in a character set description file. The table contains more than one symbolic character name for characters whose traditional name differs from the chosen name. Characters defined in **Table 6-2** (on page 130) may also be used in character set description files.

POSIX.1-2008 places only the following requirements on the encoded values of the characters in the portable character set:

- If the encoded values associated with each member of the portable character set are not invariant across all locales supported by the implementation, if an application uses any pair of locales where the character encodings differ, or accesses data from an application using a locale which has different encodings from the locales used by the application, the results are unspecified.
- The encoded values associated with the digits 0 to 9 shall be such that the value of each character after 0 shall be one greater than the value of the previous character.

- A null character, NUL, which has all bits set to zero, shall be in the set of characters.
- The encoded values associated with `<slash>` and `<period>` shall be invariant across all locales supported by the implementation.
- The encoded values associated with the members of the portable character set are each represented in a single byte. Moreover, if the value is stored in an object of C-language type `char`, it is guaranteed to be positive (except the NUL, which is always zero).

Conforming implementations shall support certain character and character set attributes, as defined in [Section 7.2](#) (on page 136).

6.2 Character Encoding

The POSIX locale contains the characters in [Table 6-1](#) (on page 125) and [Table 6-2](#) (on page 130) which have the properties listed in [Section 7.3.1](#) (on page 139). In other locales, the presence, meaning, and representation of any additional characters are locale-specific.

In locales other than the POSIX locale, a character may have a state-dependent encoding. There are two types of these encodings:

- A single-shift encoding (where each character not in the initial shift state is preceded by a shift code) can be defined if each shift-code and character sequence is considered a multi-byte character. This is done using the concatenated-constant format in a character set description file, as described in [Section 6.4](#) (on page 129). If the implementation supports a character encoding of this type, all of the standard utilities in the Shell and Utilities volume of POSIX.1-2008 shall support it. Use of a single-shift encoding with any of the functions in the System Interfaces volume of POSIX.1-2008 that do not specifically mention the effects of state-dependent encoding is implementation-defined.
- A locking-shift encoding (where the state of the character is determined by a shift code that may affect more than the single character following it) cannot be defined with the current character set description file format. Use of a locking-shift encoding with any of the standard utilities in the Shell and Utilities volume of POSIX.1-2008 or with any of the functions in the System Interfaces volume of POSIX.1-2008 that do not specifically mention the effects of state-dependent encoding is implementation-defined.

While in the initial shift state, all characters in the portable character set shall retain their usual interpretation and shall not alter the shift state. The interpretation for subsequent bytes in the sequence shall be a function of the current shift state. A byte with all bits zero shall be interpreted as the null character independent of shift state. Such a byte shall not occur as part of any other character. Likewise, the byte values used to encode `<period>` and `<slash>` shall not occur as part of any other character in any locale.

The maximum allowable number of bytes in a character in the current locale shall be indicated by `{MB_CUR_MAX}`, defined in the `<stdlib.h>` header and by the `<mb_cur_max>` value in a character set description file; see [Section 6.4](#) (on page 129). The implementation's maximum number of bytes in a character shall be defined by the C-language macro `{MB_LEN_MAX}`.

6.3 C Language Wide-Character Codes

In the shell, the standard utilities are written so that the encodings of characters are described by the locale's `LC_CTYPE` definition (see [Section 7.3.1](#), on page 139) and there is no differentiation between characters consisting of single octets (8-bit bytes) or multiple bytes. However, in the C language, a differentiation is made. To ease the handling of variable length characters, the C language has introduced the concept of wide-character codes.

All wide-character codes in a given process consist of an equal number of bits. This is in contrast to characters, which can consist of a variable number of bytes. The byte or byte sequence that represents a character can also be represented as a wide-character code. Wide-character codes thus provide a uniform size for manipulating text data. A wide-character code having all bits zero is the null wide-character code (see [Section 3.247](#), on page 72), and terminates wide-character strings (see [Section 3.437](#), on page 103). The wide-character value for each member of the portable character set shall equal its value when used as the lone character in an integer character constant. Wide-character codes for other characters are locale and implementation-defined. State shift bytes shall not have a wide-character code representation. POSIX.1-2008 provides no means of defining a wide-character codeset.

6.4 Character Set Description File

Implementations shall provide a character set description file for at least one coded character set supported by the implementation. These files are referred to elsewhere in POSIX.1-2008 as *charmap* files. It is implementation-defined whether or not users or applications can provide additional character set description files.

POSIX.1-2008 does not require that multiple character sets or codesets be supported. Although multiple charmap files are supported, it is the responsibility of the implementation to provide the file or files; if only one is provided, only that one is accessible using the *localedef* utility's `-f` option.

Each character set description file, except those that use the ISO/IEC 10646-1:2000 standard position values as the encoding values, shall define characteristics for the coded character set and the encoding for the characters specified in [Table 6-1](#) (on page 125), and may define encoding for additional characters supported by the implementation. Other information about the coded character set may also be in the file. Coded character set character values shall be defined using symbolic character names followed by character encoding values.

Each symbolic name specified in [Table 6-1](#) (on page 125) shall be included in the file and shall be mapped to a unique coding value, except as noted below. The glyphs represented by the C character constants `'{', '}', ' ', '_, ' -', ' /', ' \', ' .',` and `'^'` have more than one symbolic name; all symbolic names for each such glyph shall be included, each with identical encoding. If some or all of the control characters identified in [Table 6-2](#) (on page 130) are supported by the implementation, the symbolic names and their corresponding encoding values shall be included in the file. The encoding values shall each be represented in a single byte. Some of the encodings associated with the symbolic names in [Table 6-2](#) (on page 130) may be the same as characters found in [Table 6-1](#) (on page 125); both names shall be provided for each encoding.

Table 6-2 Control Character Set

<ACK>	<DC2>	<ENQ>	<FS>	<IS4>	<SOH>
<BEL>	<DC3>	<EOT>	<GS>	<LF>	<STX>
<BS>	<DC4>	<ESC>	<HT>	<NAK>	<SUB>
<CAN>	<DEL>	<ETB>	<IS1>	<RS>	<SYN>
<CR>	<DLE>	<ETX>	<IS2>	<SI>	<US>
<DC1>		<FF>	<IS3>	<SO>	<VT>

The following declarations can precede the character definitions. Each shall consist of the symbol shown in the following list, starting in column 1, including the surrounding brackets, followed by one or more <blank> characters, followed by the value to be assigned to the symbol.

<code_set_name> The name of the coded character set for which the character set description file is defined. The characters of the name shall be taken from the set of characters with visible glyphs defined in Table 6-1 (on page 125).

<mb_cur_max> The maximum number of bytes in a multi-byte character. This shall default to 1.

<mb_cur_min> An unsigned positive integer value that defines the minimum number of bytes in a character for the encoded character set. On XSI-conformant systems, <mb_cur_min> shall always be 1.

<escape_char> The character used to indicate that the characters following shall be interpreted in a special way, as defined later in this section. This shall default to <backslash> ('\\'), which is the character used in all the following text and examples, unless otherwise noted.

<comment_char> The character that, when placed in column 1 of a charmap line, is used to indicate that the line shall be ignored. The default character shall be the <number-sign> ('#').

The character set mapping definitions shall be all the lines immediately following an identifier line containing the string "CHARMAP" starting in column 1, and preceding a trailer line containing the string "END CHARMAP" starting in column 1. Empty lines and lines containing a <comment_char> in the first column shall be ignored. Each non-comment line of the character set mapping definition (that is, between the "CHARMAP" and "END CHARMAP" lines of the file) shall be in either of two forms:

"%s %s %s\n", <symbolic-name>, <encoding>, <comments>

or:

"%s...%s %s %s\n", <symbolic-name>, <symbolic-name>,
<encoding>, <comments>

In the first format, the line in the character set mapping definition shall define a single symbolic name and a corresponding encoding. A symbolic name is one or more characters from the set shown with visible glyphs in Table 6-1 (on page 125), enclosed between angle brackets. A character following an escape character is interpreted as itself; for example, the sequence "<\\>" represents the symbolic name "<\\>" enclosed between angle brackets.

In the second format, the line in the character set mapping definition shall define a range of one or more symbolic names. In this form, the symbolic names shall consist of zero or more non-numeric characters from the set shown with visible glyphs in Table 6-1 (on page 125), followed by an integer formed by one or more decimal digits. Both integers shall contain the same

number of digits. The characters preceding the integer shall be identical in the two symbolic names, and the integer formed by the digits in the second symbolic name shall be equal to or greater than the integer formed by the digits in the first name. This shall be interpreted as a series of symbolic names formed from the common part and each of the integers between the first and the second integer, inclusive. As an example, `<j0101>...<j0104>` is interpreted as the symbolic names `<j0101>`, `<j0102>`, `<j0103>`, and `<j0104>`, in that order.

A character set mapping definition line shall exist for all symbolic names specified in [Table 6-1](#) (on page 125), and shall define the coded character value that corresponds to the character indicated in the table, or the coded character value that corresponds to the control character symbolic name. If the control characters commonly associated with the symbolic names in [Table 6-2](#) (on page 130) are supported by the implementation, the symbolic name and the corresponding encoding value shall be included in the file. Additional unique symbolic names may be included. A coded character value can be represented by more than one symbolic name.

The encoding part is expressed as one (for single-byte character values) or more concatenated decimal, octal, or hexadecimal constants in the following formats:

```
%cd%u", <escape_char>, <decimal byte value>
%cx%x", <escape_char>, <hexadecimal byte value>
%c%o", <escape_char>, <octal byte value>
```

Decimal constants shall be represented by two or three decimal digits, preceded by the escape character and the lowercase letter 'd'; for example, `"\d05"`, `"\d97"`, or `"\d143"`. Hexadecimal constants shall be represented by two hexadecimal digits, preceded by the escape character and the lowercase letter 'x'; for example, `"\x05"`, `"\x61"`, or `"\x8f"`. Octal constants shall be represented by two or three octal digits, preceded by the escape character; for example, `"\05"`, `"\141"`, or `"\217"`. In a portable charmap file, each constant represents an 8-bit byte. When constants are concatenated for multi-byte character values, they shall be of the same type, and interpreted in sequence from first to last with the first byte of the multi-byte character specified by the first byte in the sequence. The manner in which these constants are represented in the character stored in the system is implementation-defined. (This notation was chosen for reasons of portability. There is no requirement that the internal representation in the computer memory be in this same order.) Omitting bytes from a multi-byte character definition produces undefined results.

In lines defining ranges of symbolic names, the encoded value shall be the value for the first symbolic name in the range (the symbolic name preceding the ellipsis). Subsequent symbolic names defined by the range shall have encoding values in increasing order. Bytes shall be treated as unsigned octets, and carry shall be propagated between the bytes as necessary to represent the range. However, because this causes a null byte in the second or subsequent bytes of a character, such a declaration should not be specified. For example, the line:

```
<j0101>...<j0104>  \d129\d254
```

is interpreted as:

```
<j0101>          \d129\d254
<j0102>          \d129\d255
<j0103>          \d130\d00
<j0104>          \d130\d01
```

The expanded declaration of the symbol `<j0103>` in the above example is an invalid specification, because it contains a null byte in the second byte of a character.

The comment is optional.

POSIX.1-2008 provides no means of defining a wide-character codeset.

The following declarations can follow the character set mapping definitions (after the "END CHARMAP" statement). Each shall consist of the keyword shown in the following list, starting in column 1, followed by the value(s) to be associated to the keyword, as defined below.

WIDTH A non-negative integer value defining the column width (see [Section 3.103](#), on page 50) for the printable characters in the coded character set specified in [Table 6-1](#) (on page 125) and [Table 6-2](#) (on page 130). Coded character set character values shall be defined using symbolic character names followed by column width values. Defining a character with more than one **WIDTH** produces undefined results. The **END WIDTH** keyword shall be used to terminate the **WIDTH** definitions. Specifying the width of a non-printable character in a **WIDTH** declaration produces undefined results.

WIDTH_DEFAULT A non-negative integer value defining the default column width for any printable character not listed by one of the **WIDTH** keywords. If no **WIDTH_DEFAULT** keyword is included in the charmap, the default character width shall be 1.

Example

After the "END CHARMAP" statement, a syntax for a width definition would be:

```
WIDTH
<A> 1
<B> 1
<C>...<Z> 1
...
<fool>...<foon> 2
...
END WIDTH
```

In this example, the numerical code point values represented by the symbols **<A>** and **** are assigned a width of 1. The code point values **<C>** to **<Z>** inclusive (**<C>**, **<D>**, **<E>**, and so on) are also assigned a width of 1. Using **<A>...<Z>** would have required fewer lines, but the alternative was shown to demonstrate flexibility. The keyword **WIDTH_DEFAULT** could have been added as appropriate.

6.4.1 State-Dependent Character Encodings

This section addresses the use of state-dependent character encodings (that is, those in which the encoding of a character is dependent on one or more shift codes that may precede it).

A single-shift encoding (where each character not in the initial shift state is preceded by a shift code) can be defined in the charmap format if each shift-code/character sequence is considered a multi-byte character, defined using the concatenated-constant format described in [Section 6.4](#) (on page 129). If the implementation supports a character encoding of this type, all of the standard utilities shall support it. A locking-shift encoding (where the state of the character is determined by a shift code that may affect more than the single character following it) could be defined with an extension to the charmap format described in [Section 6.4](#) (on page 129).

3824 If the implementation supports a character encoding of this type, any of the standard utilities
3825 that describe character (*versus* byte) or text-file manipulation shall have the following
3826 characteristics:

- 3827 1. The utility shall process the statefully encoded data as a concatenation of state-
3828 independent characters. The presence of redundant locking shifts shall not affect the
3829 comparison of two statefully encoded strings.
- 3830 2. A utility that divides, truncates, or extracts substrings from statefully encoded data shall
3831 produce output that contains locking shifts at the beginning or end of the resulting data,
3832 if appropriate, to retain correct state information.

Locale

7.1 General

A locale is the definition of the subset of a user's environment that depends on language and cultural conventions. It is made up from one or more categories. Each category is identified by its name and controls specific aspects of the behavior of components of the system. Category names correspond to the following environment variable names:

LC_CTYPE Character classification and case conversion.

LC_COLLATE Collation order.

LC_MONETARY Monetary formatting.

LC_NUMERIC Numeric, non-monetary formatting.

LC_TIME Date and time formats.

LC_MESSAGES Formats of informative and diagnostic messages and interactive responses.

The standard utilities in the Shell and Utilities volume of POSIX.1-2008 shall base their behavior on the current locale, as defined in the ENVIRONMENT VARIABLES section for each utility. The behavior of some of the C-language functions defined in the System Interfaces volume of POSIX.1-2008 shall also be modified based on a locale selection. The locale to be used by these functions can be selected in the following ways:

1. For functions such as *isalnum_l()* that take a locale object as an argument, a locale object can be obtained from *newlocale()* or *duplocale()* and passed to the function.

2. For functions that do not take a locale object as an argument, the current locale for the thread can be set by calling *uselocale()* or the global locale for the process can be set by calling *setlocale()*. Such functions shall use the current locale of the calling thread if one has been set for that thread; otherwise, they shall use the global locale.

Locales other than those supplied by the implementation can be created via the *localedef* utility, provided that the *_POSIX2_LOCALEDEF* symbol is defined on the system. Even if *localedef* is not provided, all implementations conforming to the System Interfaces volume of POSIX.1-2008 shall provide one or more locales that behave as described in this chapter. The input to the utility is described in [Section 7.3](#) (on page 136). The value that is used to specify a locale when using environment variables shall be the string specified as the *name* operand to the *localedef* utility when the locale was created. The strings "C" and "POSIX" are reserved as identifiers for the POSIX locale (see [Section 7.2](#), on page 136). When the value of a locale environment variable begins with a <slash> (' / '), it shall be interpreted as the pathname of the locale definition; the type of file (regular, directory, and so on) used to store the locale definition is implementation-defined. If the value does not begin with a <slash>, the mechanism used to locate the locale is implementation-defined.

If different character sets are used by the locale categories, the results achieved by an application utilizing these categories are undefined. Likewise, if different codesets are used for the data

being processed by interfaces whose behavior is dependent on the current locale, or the codeset is different from the codeset assumed when the locale was created, the result is also undefined.

Applications can select the desired locale by calling the *newlocale()* or *setlocale()* function with the appropriate value. If the function is invoked with an empty string, such as:

```
newlocale(LC_ALL_MASK, "", (locale_t)0);
```

or:

```
setlocale(LC_ALL, "");
```

the value of the corresponding environment variable is used. If the environment variable is unset or is set to the empty string, the implementation shall set the appropriate environment as defined in [Chapter 8](#) (on page 173).

7.2 POSIX Locale

Conforming systems shall provide a POSIX locale, also known as the C locale. The behavior of standard utilities and functions in the POSIX locale shall be as if the locale was defined via the *localedef* utility with input data from the POSIX locale tables in [Section 7.3](#).

The tables in [Section 7.3](#) describe the characteristics and behavior of the POSIX locale for data consisting entirely of characters from the portable character set and the control character set. For other characters, the behavior is unspecified. For C-language programs, the POSIX locale shall be the default locale when the *setlocale()* function is not called.

The POSIX locale can be specified by assigning to the appropriate environment variables the values "C" or "POSIX".

All implementations shall define a locale as the default locale, to be invoked when no environment variables are set, or set to the empty string. This default locale can be the POSIX locale or any other implementation-defined locale. Some implementations may provide facilities for local installation administrators to set the default locale, customizing it for each location. POSIX.1-2008 does not require such a facility.

7.3 Locale Definition

The capability to specify additional locales to those provided by an implementation is optional, denoted by the `_POSIX2_LOCALEDEF` symbol. If the option is not supported, only implementation-supplied locales are available. Such locales shall be documented using the format specified in this section.

Locales can be described with the file format presented in this section. The file format is that accepted by the *localedef* utility. For the purposes of this section, the file is referred to as the "locale definition file", but no locales shall be affected by this file unless it is processed by *localedef* or some similar mechanism. Any requirements in this section imposed upon the utility shall apply to *localedef* or to any other similar utility used to install locale information using the locale definition file format described here.

The locale definition file shall contain one or more locale category source definitions, and shall not contain more than one definition for the same locale category. If the file contains source definitions for more than one category, implementation-defined categories, if present, shall appear after the categories defined by [Section 7.1](#) (on page 135). A category source definition

contains either the definition of a category or a **copy** directive. For a description of the **copy** directive, see *localedef*. In the event that some of the information for a locale category, as specified in this volume of POSIX.1-2008, is missing from the locale source definition, the behavior of that category, if it is referenced, is unspecified.

A category source definition shall consist of a category header, a category body, and a category trailer. A category header shall consist of the character string naming of the category, beginning with the characters *LC_*. The category trailer shall consist of the string "END", followed by one or more <blank> characters and the string used in the corresponding category header.

The category body shall consist of one or more lines of text. Each line shall contain an identifier, optionally followed by one or more operands. Identifiers shall be either keywords, identifying a particular locale element, or collating elements. In addition to the keywords defined in this volume of POSIX.1-2008, the source can contain implementation-defined keywords. Each keyword within a locale shall have a unique name (that is, two categories cannot have a commonly-named keyword); no keyword shall start with the characters *LC_*. Identifiers shall be separated from the operands by one or more <blank> characters.

Operands shall be characters, collating elements, or strings of characters. Strings shall be enclosed in double-quotes. Literal double-quotes within strings shall be preceded by the <escape character>, described below. When a keyword is followed by more than one operand, the operands shall be separated by <semicolon> characters; <blank> characters shall be allowed both before and after a <semicolon>.

The first category header in the file can be preceded by a line modifying the comment character. It shall have the following format, starting in column 1:

```
"comment_char %c\n", <comment character>
```

The comment character shall default to the <number-sign> ('#'). Blank lines and lines containing the <comment character> in the first position shall be ignored.

The first category header in the file can be preceded by a line modifying the escape character to be used in the file. It shall have the following format, starting in column 1:

```
"escape_char %c\n", <escape character>
```

The escape character shall default to <backslash>, which is the character used in all examples shown in this volume of POSIX.1-2008.

A line can be continued by placing an escape character as the last character on the line; this continuation character shall be discarded from the input. Although the implementation need not accept any one portion of a continued line with a length exceeding {LINE_MAX} bytes, it shall place no limits on the accumulated length of the continued line. Comment lines shall not be continued on a subsequent line using an escaped <newline>.

Individual characters, characters in strings, and collating elements shall be represented using symbolic names, as defined below. In addition, characters can be represented using the characters themselves or as octal, hexadecimal, or decimal constants. When non-symbolic notation is used, the resultant locale definitions are in many cases not portable between systems. The left angle bracket ('<') is a reserved symbol, denoting the start of a symbolic name; when used to represent itself it shall be preceded by the escape character. The following rules apply to character representation:

1. A character can be represented via a symbolic name, enclosed within angle brackets '<' and '>'. The symbolic name, including the angle brackets, shall exactly match a symbolic name defined in the charmap file specified via the *localedef* -f option, and it shall be replaced by a character value determined from the value associated with the symbolic name in the charmap file. The use of a symbolic name not found in the charmap file shall

constitute an error, unless the category is *LC_CTYPE* or *LC_COLLATE*, in which case it shall constitute a warning condition (see *localedef* for a description of actions resulting from errors and warnings). The specification of a symbolic name in a **collating-element** or **collating-symbol** section that duplicates a symbolic name in the charmap file (if present) shall be an error. Use of the escape character or a right angle bracket within a symbolic name is invalid unless the character is preceded by the escape character.

For example:

```
<c>;<c-cedilla>  "<M><a><y>"
```

2. A character in the portable character set can be represented by the character itself, in which case the value of the character is implementation-defined. (Implementations may allow other characters to be represented as themselves, but such locale definitions are not portable.) Within a string, the double-quote character, the escape character, and the right angle bracket character shall be escaped (preceded by the escape character) to be interpreted as the character itself. Outside strings, the characters:

```
,      ;      <      >      escape_char
```

shall be escaped to be interpreted as the character itself.

For example:

```
c      "May"
```

3. A character can be represented as an octal constant. An octal constant shall be specified as the escape character followed by two or three octal digits. Each constant shall represent a byte value. Multi-byte values can be represented by concatenated constants specified in byte order with the last constant specifying the least significant byte of the character.

For example:

```
\143;\347;\143\150  "\115\141\171"
```

4. A character can be represented as a hexadecimal constant. A hexadecimal constant shall be specified as the escape character followed by an 'x' followed by two hexadecimal digits. Each constant shall represent a byte value. Multi-byte values can be represented by concatenated constants specified in byte order with the last constant specifying the least significant byte of the character.

For example:

```
\x63;\xe7;\x63\x68  "\x4d\x61\x79"
```

5. A character can be represented as a decimal constant. A decimal constant shall be specified as the escape character followed by a 'd' followed by two or three decimal digits. Each constant represents a byte value. Multi-byte values can be represented by concatenated constants specified in byte order with the last constant specifying the least significant byte of the character.

For example:

```
\d99;\d231;\d99\d104  "\d77\d97\d121"
```

Implementations may accept single-digit octal, decimal, or hexadecimal constants following the escape character. Only characters existing in the character set for which the locale definition is created shall be specified, whether using symbolic names, the characters themselves, or octal, decimal, or hexadecimal constants. If a charmap file is present, only characters defined in the charmap can be specified using octal, decimal, or hexadecimal constants. Symbolic names not present in the charmap file can be specified and shall be ignored, as specified under item 1

4002 above.

4003 7.3.1 LC_CTYPE

4004 The *LC_CTYPE* category shall define character classification, case conversion, and other
 4005 character attributes. In addition, a series of characters can be represented by three adjacent
 4006 <period> characters representing an ellipsis symbol (" . . . "). The ellipsis specification shall be
 4007 interpreted as meaning that all values between the values preceding and following it represent
 4008 valid characters. The ellipsis specification shall be valid only within a single encoded character
 4009 set; that is, within a group of characters of the same size. An ellipsis shall be interpreted as
 4010 including in the list all characters with an encoded value higher than the encoded value of the
 4011 character preceding the ellipsis and lower than the encoded value of the character following the
 4012 ellipsis.

4013 For example:

4014 \x30; . . . ; \x39;

4015 includes in the character class all characters with encoded values between the endpoints.

4016 The following keywords shall be recognized. In the descriptions, the term “automatically
 4017 included” means that it shall not be an error either to include or omit any of the referenced
 4018 characters; the implementation provides them if missing (even if the entire keyword is missing)
 4019 and accepts them silently if present. When the implementation automatically includes a missing
 4020 character, it shall have an encoded value dependent on the charmap file in effect (see the
 4021 description of the *localedef* -f option); otherwise, it shall have a value derived from an
 4022 implementation-defined character mapping.

4023 The character classes **digit**, **xdigit**, **lower**, **upper**, and **space** have a set of automatically included
 4024 characters. These only need to be specified if the character values (that is, encoding) differ from
 4025 the implementation default values. It is not possible to define a locale without these
 4026 automatically included characters unless some implementation extension is used to prevent
 4027 their inclusion. Such a definition would not be a proper superset of the C or POSIX locale and,
 4028 thus, it might not be possible for conforming applications to work properly.

4029 **copy** Specify the name of an existing locale which shall be used as the definition of
 4030 this category. If this keyword is specified, no other keyword shall be specified.

4031 **upper** Define characters to be classified as uppercase letters.
 4032 In the POSIX locale, the 26 uppercase letters shall be included:

4033 A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

4034 In a locale definition file, no character specified for the keywords **cntrl**, **digit**,
 4035 **punct**, or **space** shall be specified. The uppercase letters <A> to <Z>, as
 4036 defined in [Section 6.4](#) (on page 129) (the portable character set), are
 4037 automatically included in this class.

4038 **lower** Define characters to be classified as lowercase letters.
 4039 In the POSIX locale, the 26 lowercase letters shall be included:

4040 a b c d e f g h i j k l m n o p q r s t u v w x y z

4041 In a locale definition file, no character specified for the keywords **cntrl**, **digit**,
 4042 **punct**, or **space** shall be specified. The lowercase letters <a> to <z> of the
 4043 portable character set are automatically included in this class.

4044	alpha	Define characters to be classified as letters.
4045		In the POSIX locale, all characters in the classes upper and lower shall be
4046		included.
4047		In a locale definition file, no character specified for the keywords cntrl , digit ,
4048		punct , or space shall be specified. Characters classified as either upper or
4049		lower are automatically included in this class.
4050	digit	Define the characters to be classified as numeric digits.
4051		In the POSIX locale, only:
4052		0 1 2 3 4 5 6 7 8 9
4053		shall be included.
4054		In a locale definition file, only the digits <zero>, <one>, <two>, <three>,
4055		<four>, <five>, <six>, <seven>, <eight>, and <nine> shall be specified, and in
4056		contiguous ascending sequence by numerical value. The digits <zero> to
4057		<nine> of the portable character set are automatically included in this class.
4058	alnum	Define characters to be classified as letters and numeric digits. Only the
4059		characters specified for the alpha and digit keywords shall be specified.
4060		Characters specified for the keywords alpha and digit are automatically
4061		included in this class.
4062	space	Define characters to be classified as white-space characters.
4063		In the POSIX locale, exactly <space>, <form-feed>, <newline>, <carriage-
4064		return>, <tab>, and <vertical-tab> shall be included.
4065		In a locale definition file, no character specified for the keywords upper ,
4066		lower , alpha , digit , graph , or xdigit shall be specified. The <space>, <form-
4067		feed>, <newline>, <carriage-return>, <tab>, and <vertical-tab> of the portable
4068		character set, and any characters included in the class blank are automatically
4069		included in this class.
4070	cntrl	Define characters to be classified as control characters.
4071		In the POSIX locale, no characters in classes alpha or print shall be included.
4072		In a locale definition file, no character specified for the keywords upper ,
4073		lower , alpha , digit , punct , graph , print , or xdigit shall be specified.
4074	punct	Define characters to be classified as punctuation characters.
4075		In the POSIX locale, neither the <space> nor any characters in classes alpha ,
4076		digit , or cntrl shall be included.
4077		In a locale definition file, no character specified for the keywords upper ,
4078		lower , alpha , digit , cntrl , xdigit , or as the <space> shall be specified.
4079	graph	Define characters to be classified as printable characters, not including the
4080		<space>.
4081		In the POSIX locale, all characters in classes alpha , digit , and punct shall be
4082		included; no characters in class cntrl shall be included.
4083		In a locale definition file, characters specified for the keywords upper , lower ,
4084		alpha , digit , xdigit , and punct are automatically included in this class. No
4085		character specified for the keyword cntrl shall be specified.

4086	print	Define characters to be classified as printable characters, including the
4087		<space>.
4088		In the POSIX locale, all characters in class graph shall be included; no
4089		characters in class cntrl shall be included.
4090		In a locale definition file, characters specified for the keywords upper , lower ,
4091		alpha , digit , xdigit , punct , graph , and the <space> are automatically included
4092		in this class. No character specified for the keyword cntrl shall be specified.
4093	xdigit	Define the characters to be classified as hexadecimal digits.
4094		In the POSIX locale, only:
4095		0 1 2 3 4 5 6 7 8 9 A B C D E F a b c d e f
4096		shall be included.
4097		In a locale definition file, only the characters defined for the class digit shall be
4098		specified, in contiguous ascending sequence by numerical value, followed by
4099		one or more sets of six characters representing the hexadecimal digits 10 to 15
4100		inclusive, with each set in ascending order (for example, <A>, , <C>, <D>,
4101		<E>, <F>, <a>, , <c>, <d>, <e>, <f>). The digits <zero> to <nine>, the
4102		uppercase letters <A> to <F>, and the lowercase letters <a> to <f> of the
4103		portable character set are automatically included in this class.
4104	blank	Define characters to be classified as <blank> characters.
4105		In the POSIX locale, only the <space> and <tab> shall be included.
4106		In a locale definition file, the <space> and <tab> are automatically included in
4107		this class.
4108	charclass	Define one or more locale-specific character class names as strings separated
4109		by <semicolon> characters. Each named character class can then be defined
4110		subsequently in the <i>LC_CTYPE</i> definition. A character class name shall consist
4111		of at least one and at most {CHARCLASS_NAME_MAX} bytes of
4112		alphanumeric characters from the portable filename character set. The first
4113		character of a character class name shall not be a digit. The name shall not
4114		match any of the <i>LC_CTYPE</i> keywords defined in this volume of
4115		POSIX.1-2008. Future versions of this standard will not specify any <i>LC_CTYPE</i>
4116		keywords containing uppercase letters.
4117	<i>charclass-name</i>	Define characters to be classified as belonging to the named locale-specific
4118		character class. In the POSIX locale, locale-specific named character classes
4119		need not exist.
4120		If a class name is defined by a charclass keyword, but no characters are
4121		subsequently assigned to it, this is not an error; it represents a class without
4122		any characters belonging to it.
4123		The <i>charclass-name</i> can be used as the <i>property</i> argument to the <i>wctype()</i>
4124		function, in regular expression and shell pattern-matching bracket
4125		expressions, and by the <i>tr</i> command.
4126	toupper	Define the mapping of lowercase letters to uppercase letters.
4127		In the POSIX locale, at a minimum, the 26 lowercase characters:
4128		a b c d e f g h i j k l m n o p q r s t u v w x y z
4129		shall be mapped to the corresponding 26 uppercase characters:

4130 A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

4131 In a locale definition file, the operand shall consist of character pairs,
 4132 separated by <semicolon> characters. The characters in each character pair
 4133 shall be separated by a <comma> and the pair enclosed by parentheses. The
 4134 first character in each pair is the lowercase letter, the second the corresponding
 4135 uppercase letter. Only characters specified for the keywords **lower** and **upper**
 4136 shall be specified. The lowercase letters <a> to <z>, and their corresponding
 4137 uppercase letters <A> to <Z>, of the portable character set are automatically
 4138 included in this mapping, but only when the **toupper** keyword is omitted
 4139 from the locale definition.

4140 **tolower** Define the mapping of uppercase letters to lowercase letters.

4141 In the POSIX locale, at a minimum, the 26 uppercase characters:

4142 A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

4143 shall be mapped to the corresponding 26 lowercase characters:

4144 a b c d e f g h i j k l m n o p q r s t u v w x y z

4145 In a locale definition file, the operand shall consist of character pairs,
 4146 separated by <semicolon> characters. The characters in each character pair
 4147 shall be separated by a <comma> and the pair enclosed by parentheses. The
 4148 first character in each pair is the uppercase letter, the second the
 4149 corresponding lowercase letter. Only characters specified for the keywords
 4150 **lower** and **upper** shall be specified. If the **tolower** keyword is omitted from
 4151 the locale definition, the mapping is the reverse mapping of the one specified
 4152 for **toupper**.

4153 The following table shows the character class combinations allowed:

4154 **Table 7-1** Valid Character Class Combinations

In Class	Can Also Belong To										
	upper	lower	alpha	digit	space	cntrl	punct	graph	print	xdigit	blank
upper		—	A	x	x	x	x	A	A	—	x
lower	—		A	x	x	x	x	A	A	—	x
alpha	—	—		x	x	x	x	A	A	—	x
digit	x	x	x		x	x	x	A	A	A	x
space	x	x	x	x		—	*	*	*	x	—
cntrl	x	x	x	x	—		x	x	x	x	—
punct	x	x	x	x	—	x		A	A	x	—
graph	—	—	—	—	—	x	—		A	—	—
print	—	—	—	—	—	x	—	—		—	—
xdigit	—	—	—	—	x	x	x	A	A		x
blank	x	x	x	x	A	—	*	*	*	x	

Notes:

1. Explanation of codes:
 - A Automatically included; see text.
 - Permitted.
 - x Mutually-exclusive.
 - * See note 2.
2. The <space>, which is part of the **space** and **blank** classes, cannot belong to **punct** or **graph**, but shall automatically belong to the **print** class. Other **space** or **blank** characters can be classified as any of **punct**, **graph**, or **print**.

7.3.1.1 LC_CTYPE Category in the POSIX Locale

The character classifications for the POSIX locale follow; the code listing depicts the *localedef* input, and the table represents the same information, sorted by character.

```
LC_CTYPE
# The following is the POSIX locale LC_CTYPE.
# "alpha" is by default "upper" and "lower"
# "alnum" is by definition "alpha" and "digit"
# "print" is by default "alnum", "punct", and the <space>
# "graph" is by default "alnum" and "punct"
#
upper    <A>;<B>;<C>;<D>;<E>;<F>;<G>;<H>;<I>;<J>;<K>;<L>;<M>;\
         <N>;<O>;<P>;<Q>;<R>;<S>;<T>;<U>;<V>;<W>;<X>;<Y>;<Z>
#
lower    <a>;<b>;<c>;<d>;<e>;<f>;<g>;<h>;<i>;<j>;<k>;<l>;<m>;\
         <n>;<o>;<p>;<q>;<r>;<s>;<t>;<u>;<v>;<w>;<x>;<y>;<z>
#
digit    <zero>;<one>;<two>;<three>;<four>;<five>;<six>;\
         <seven>;<eight>;<nine>
#
space    <tab>;<newline>;<vertical-tab>;<form-feed>;\
         <carriage-return>;<space>
#
cntrl    <alert>;<backspace>;<tab>;<newline>;<vertical-tab>;\
         <form-feed>;<carriage-return>;\
         <NUL>;<SOH>;<STX>;<ETX>;<EOT>;<ENQ>;<ACK>;<SO>;\
         <SI>;<DLE>;<DC1>;<DC2>;<DC3>;<DC4>;<NAK>;<SYN>;\
         <ETB>;<CAN>;<EM>;<SUB>;<ESC>;<IS4>;<IS3>;<IS2>;\
         <IS1>;<DEL>
#
punct    <exclamation-mark>;<quotation-mark>;<number-sign>;\
         <dollar-sign>;<percent-sign>;<ampersand>;<apostrophe>;\
         <left-parenthesis>;<right-parenthesis>;<asterisk>;\
         <plus-sign>;<comma>;<hyphen>;<period>;<slash>;\
         <colon>;<semicolon>;<less-than-sign>;<equals-sign>;\
         <greater-than-sign>;<question-mark>;<commercial-at>;\
         <left-square-bracket>;<backslash>;<right-square-bracket>;\
         <circumflex>;<underscore>;<grave-accent>;<left-curly-bracket>;\
         <vertical-line>;<right-curly-bracket>;<tilde>
```

```

4215      #
4216      xdigit    <zero>;<one>;<two>;<three>;<four>;<five>;<six>;<seven>;\
4217              <eight>;<nine>;<A>;<B>;<C>;<D>;<E>;<F>;<a>;<b>;<c>;<d>;<e>;<f>
4218      #
4219      blank     <space>;<tab>
4220      #
4221      toupper   (<a>, <A>); (<b>, <B>); (<c>, <C>); (<d>, <D>); (<e>, <E>); \
4222              (<f>, <F>); (<g>, <G>); (<h>, <H>); (<i>, <I>); (<j>, <J>); \
4223              (<k>, <K>); (<l>, <L>); (<m>, <M>); (<n>, <N>); (<o>, <O>); \
4224              (<p>, <P>); (<q>, <Q>); (<r>, <R>); (<s>, <S>); (<t>, <T>); \
4225              (<u>, <U>); (<v>, <V>); (<w>, <W>); (<x>, <X>); (<y>, <Y>); (<z>, <Z>)
4226      #
4227      tolower   (<A>, <a>); (<B>, <b>); (<C>, <c>); (<D>, <d>); (<E>, <e>); \
4228              (<F>, <f>); (<G>, <g>); (<H>, <h>); (<I>, <i>); (<J>, <j>); \
4229              (<K>, <k>); (<L>, <l>); (<M>, <m>); (<N>, <n>); (<O>, <o>); \
4230              (<P>, <p>); (<Q>, <q>); (<R>, <r>); (<S>, <s>); (<T>, <t>); \
4231              (<U>, <u>); (<V>, <v>); (<W>, <w>); (<X>, <x>); (<Y>, <y>); (<Z>, <z>)
4232      END LC_CTYPE

```

4233

4234

4235

4236

4237

4238

4239

4240

4241

4242

4243

4244

4245

4246

4247

4248

4249

4250

4251

4252

4253

4254

4255

4256

4257

4258

4259

4260

4261

4262

4263

4264

4265

Symbolic Name	Other Case	Character Classes
<NUL>		cntrl
<SOH>		cntrl
<STX>		cntrl
<ETX>		cntrl
<EOT>		cntrl
<ENQ>		cntrl
<ACK>		cntrl
<alert>		cntrl
<backspace>		cntrl
<tab>		cntrl, space, blank
<newline>		cntrl, space
<vertical-tab>		cntrl, space
<form-feed>		cntrl, space
<carriage-return>		cntrl, space
<SO>		cntrl
<SI>		cntrl
<DLE>		cntrl
<DC1>		cntrl
<DC2>		cntrl
<DC3>		cntrl
<DC4>		cntrl
<NAK>		cntrl
<SYN>		cntrl
<ETB>		cntrl
<CAN>		cntrl
		cntrl
<SUB>		cntrl
<ESC>		cntrl
<IS4>		cntrl
<IS3>		cntrl
<IS2>		cntrl
<IS1>		cntrl

	Symbolic Name	Other Case	Character Classes
4266	<space>		space, print, blank
4267	<exclamation-mark>		punct, print, graph
4268	<quotation-mark>		punct, print, graph
4269	<number-sign>		punct, print, graph
4270	<dollar-sign>		punct, print, graph
4271	<percent-sign>		punct, print, graph
4272	<ampersand>		punct, print, graph
4273	<apostrophe>		punct, print, graph
4274	<left-parenthesis>		punct, print, graph
4275	<right-parenthesis>		punct, print, graph
4276	<asterisk>		punct, print, graph
4277	<plus-sign>		punct, print, graph
4278	<comma>		punct, print, graph
4279	<hyphen>		punct, print, graph
4280	<period>		punct, print, graph
4281	<slash>		punct, print, graph
4282	<zero>		digit, xdigit, print, graph
4283	<one>		digit, xdigit, print, graph
4284	<two>		digit, xdigit, print, graph
4285	<three>		digit, xdigit, print, graph
4286	<four>		digit, xdigit, print, graph
4287	<five>		digit, xdigit, print, graph
4288	<six>		digit, xdigit, print, graph
4289	<seven>		digit, xdigit, print, graph
4290	<eight>		digit, xdigit, print, graph
4291	<nine>		digit, xdigit, print, graph
4292	<colon>		punct, print, graph
4293	<semicolon>		punct, print, graph
4294	<less-than-sign>		punct, print, graph
4295	<equals-sign>		punct, print, graph
4296	<greater-than-sign>		punct, print, graph
4297	<question-mark>		punct, print, graph
4298	<commercial-at>		punct, print, graph
4299	<A>	<a>	upper, xdigit, alpha, print, graph
4300			upper, xdigit, alpha, print, graph
4301	<C>	<c>	upper, xdigit, alpha, print, graph
4302	<D>	<d>	upper, xdigit, alpha, print, graph
4303	<E>	<e>	upper, xdigit, alpha, print, graph
4304	<F>	<f>	upper, xdigit, alpha, print, graph
4305	<G>	<g>	upper, alpha, print, graph
4306	<H>	<h>	upper, alpha, print, graph
4307	<I>	<i>	upper, alpha, print, graph
4308	<J>	<j>	upper, alpha, print, graph
4309	<K>	<k>	upper, alpha, print, graph
4310	<L>	<l>	upper, alpha, print, graph
4311	<M>	<m>	upper, alpha, print, graph
4312	<N>	<n>	upper, alpha, print, graph
4313	<O>	<o>	upper, alpha, print, graph
4314	<P>	<p>	upper, alpha, print, graph
4315	<Q>	<q>	upper, alpha, print, graph
4316	<R>	<r>	upper, alpha, print, graph
4317			

	Symbolic Name	Other Case	Character Classes
4318			
4319	<S>	<s>	upper, alpha, print, graph
4320	<T>	<t>	upper, alpha, print, graph
4321	<U>	<u>	upper, alpha, print, graph
4322	<V>	<v>	upper, alpha, print, graph
4323	<W>	<w>	upper, alpha, print, graph
4324	<X>	<x>	upper, alpha, print, graph
4325	<Y>	<y>	upper, alpha, print, graph
4326	<Z>	<z>	upper, alpha, print, graph
4327	<left-square-bracket>		punct, print, graph
4328	<backslash>		punct, print, graph
4329	<right-square-bracket>		punct, print, graph
4330	<circumflex>		punct, print, graph
4331	<underscore>		punct, print, graph
4332	<grave-accent>		punct, print, graph
4333	<a>	<A>	lower, xdigit, alpha, print, graph
4334			lower, xdigit, alpha, print, graph
4335	<c>	<C>	lower, xdigit, alpha, print, graph
4336	<d>	<D>	lower, xdigit, alpha, print, graph
4337	<e>	<E>	lower, xdigit, alpha, print, graph
4338	<f>	<F>	lower, xdigit, alpha, print, graph
4339	<g>	<G>	lower, alpha, print, graph
4340	<h>	<H>	lower, alpha, print, graph
4341	<i>	<I>	lower, alpha, print, graph
4342	<j>	<J>	lower, alpha, print, graph
4343	<k>	<K>	lower, alpha, print, graph
4344	<l>	<L>	lower, alpha, print, graph
4345	<m>	<M>	lower, alpha, print, graph
4346	<n>	<N>	lower, alpha, print, graph
4347	<o>	<O>	lower, alpha, print, graph
4348	<p>	<P>	lower, alpha, print, graph
4349	<q>	<Q>	lower, alpha, print, graph
4350	<r>	<R>	lower, alpha, print, graph
4351	<s>	<S>	lower, alpha, print, graph
4352	<t>	<T>	lower, alpha, print, graph
4353	<u>	<U>	lower, alpha, print, graph
4354	<v>	<V>	lower, alpha, print, graph
4355	<w>	<W>	lower, alpha, print, graph
4356	<x>	<X>	lower, alpha, print, graph
4357	<y>	<Y>	lower, alpha, print, graph
4358	<z>	<Z>	lower, alpha, print, graph
4359	<left-curly-bracket>		punct, print, graph
4360	<vertical-line>		punct, print, graph
4361	<right-curly-bracket>		punct, print, graph
4362	<tilde>		punct, print, graph
4363	<DEL>		cntrl

7.3.2 LC_COLLATE

The *LC_COLLATE* category provides a collation sequence definition for numerous utilities in the Shell and Utilities volume of POSIX.1-2008 (*sort*, *uniq*, and so on), regular expression matching (see [Chapter 9](#), on page 181), and the *strcoll()*, *strxfrm()*, *wscoll()*, and *wcsxfrm()* functions in the System Interfaces volume of POSIX.1-2008.

A collation sequence definition shall define the relative order between collating elements (characters and multi-character collating elements) in the locale. This order is expressed in terms of collation values; that is, by assigning each element one or more collation values (also known as collation weights). This does not imply that implementations shall assign such values, but that ordering of strings using the resultant collation definition in the locale behaves as if such assignment is done and used in the collation process. At least the following capabilities are provided:

1. **Multi-character collating elements.** Specification of multi-character collating elements (that is, sequences of two or more characters to be collated as an entity).
2. **User-defined ordering of collating elements.** Each collating element shall be assigned a collation value defining its order in the character (or basic) collation sequence. This ordering is used by regular expressions and pattern matching and, unless collation weights are explicitly specified, also as the collation weight to be used in sorting.
3. **Multiple weights and equivalence classes.** Collating elements can be assigned one or more (up to the limit {*COLL_WEIGHTS_MAX*}, as defined in [limits.h](#)) collating weights for use in sorting. The first weight is hereafter referred to as the primary weight.
4. **One-to-many mapping.** A single character is mapped into a string of collating elements.
5. **Equivalence class definition.** Two or more collating elements have the same collation value (primary weight).
6. **Ordering by weights.** When two strings are compared to determine their relative order, the two strings are first broken up into a series of collating elements; the elements in each successive pair of elements are then compared according to the relative primary weights for the elements. If equal, and more than one weight has been assigned, then the pairs of collating elements are re-compared according to the relative subsequent weights, until either a pair of collating elements compare unequal or the weights are exhausted.

The following keywords shall be recognized in a collation sequence definition. They are described in detail in the following sections.

copy	Specify the name of an existing locale which shall be used as the definition of this category. If this keyword is specified, no other keyword shall be specified.
collating-element	Define a collating-element symbol representing a multi-character collating element. This keyword is optional.
collating-symbol	Define a collating symbol for use in collation order statements. This keyword is optional.
order_start	Define collation rules. This statement shall be followed by one or more collation order statements, assigning character collation values and collation weights to collating elements.
order_end	Specify the end of the collation-order statements.

4407 7.3.2.1 *The collating-element Keyword*

4408 In addition to the collating elements in the character set, the **collating-element** keyword can be
 4409 used to define multi-character collating elements. The syntax is as follows:

4410 "collating-element %s from \"%s\\n\", <collating-symbol>, <string>

4411 The <collating-symbol> operand shall be a symbolic name, enclosed between angle brackets ('<' and '>'), and shall not duplicate any symbolic name in the current charmap file (if any), or any
 4412 other symbolic name defined in this collation definition. The string operand is a string of two or
 4413 more characters that collates as an entity. A <collating-element> defined via this keyword is only
 4414 recognized with the LC_COLLATE category.
 4415

4416 For example:

4417 collating-element <ch> from "<c><h>"
 4418 collating-element <e-acute> from "<acute><e>"
 4419 collating-element <ll> from "ll"

4420 7.3.2.2 *The collating-symbol Keyword*

4421 This keyword shall be used to define symbols for use in collation sequence statements; that is,
 4422 between the **order_start** and the **order_end** keywords. The syntax is as follows:

4423 "collating-symbol %s\\n\", <collating-symbol>

4424 The <collating-symbol> shall be a symbolic name, enclosed between angle brackets ('<' and '>'), and shall not duplicate any symbolic name in the current charmap file (if any), or any
 4425 other symbolic name defined in this collation definition. A <collating-symbol> defined via this
 4426 keyword is only recognized within the LC_COLLATE category.
 4427

4428 For example:

4429 collating-symbol <UPPER_CASE>
 4430 collating-symbol <HIGH>

4431 The **collating-symbol** keyword defines a symbolic name that can be associated with a relative
 4432 position in the character order sequence. While such a symbolic name does not represent any
 4433 collating element, it can be used as a weight.

4434 7.3.2.3 *The order_start Keyword*

4435 The **order_start** keyword shall precede collation order entries and also define the number of
 4436 weights for this collation sequence definition and other collation rules. The syntax is as follows:

4437 "order_start %s;%s;...;%s\\n\", <sort-rules>, <sort-rules> ...

4438 The operands to the **order_start** keyword are optional. If present, the operands define rules to be
 4439 applied when strings are compared. The number of operands define how many weights each
 4440 element is assigned; if no operands are present, one **forward** operand is assumed. If present, the
 4441 first operand defines rules to be applied when comparing strings using the first (primary)
 4442 weight; the second when comparing strings using the second weight, and so on. Operands shall
 4443 be separated by <semicolon> characters (';'). Each operand shall consist of one or more
 4444 collation directives, separated by <comma> characters (','). If the number of operands exceeds
 4445 the {COLL_WEIGHTS_MAX} limit, the utility shall issue a warning message. The following
 4446 directives shall be supported:

4447 **forward** Specifies that comparison operations for the weight level shall proceed from start
4448 of string towards the end of string.

4449 **backward** Specifies that comparison operations for the weight level shall proceed from end of
4450 string towards the beginning of string.

4451 **position** Specifies that comparison operations for the weight level shall consider the relative
4452 position of elements in the strings not subject to **IGNORE**. The string containing
4453 an element not subject to **IGNORE** after the fewest collating elements subject to
4454 **IGNORE** from the start of the compare shall collate first. If both strings contain a
4455 character not subject to **IGNORE** in the same relative position, the collating values
4456 assigned to the elements shall determine the ordering. In case of equality,
4457 subsequent characters not subject to **IGNORE** shall be considered in the same
4458 manner.

4459 The directives **forward** and **backward** are mutually-exclusive.

4460 If no operands are specified, a single **forward** operand shall be assumed.

4461 For example:

4462 order_start forward;backward

4463 7.3.2.4 Collation Order

4464 The **order_start** keyword shall be followed by collating identifier entries. The syntax for the
4465 collating element entries is as follows:

4466 "%s %s;%s;...;%s\n", <collating-identifier>, <weight>, <weight>, ...

4467 Each *collating-identifier* shall consist of either a character (in any of the forms defined in [Section](#)
4468 7.3, on page 136), a <collating-element>, a <collating-symbol>, an ellipsis, or the special symbol
4469 **UNDEFINED**. The order in which collating elements are specified determines the character
4470 order sequence, such that each collating element shall compare less than the elements following
4471 it.

4472 A <collating-element> shall be used to specify multi-character collating elements, and indicates
4473 that the character sequence specified via the <collating-element> is to be collated as a unit and in
4474 the relative order specified by its place.

4475 A <collating-symbol> can be used to define a position in the relative order for use in weights. No
4476 weights shall be specified with a <collating-symbol>.

4477 The ellipsis symbol specifies that a sequence of characters shall collate according to their
4478 encoded character values. It shall be interpreted as indicating that all characters with a coded
4479 character set value higher than the value of the character in the preceding line, and lower than
4480 the coded character set value for the character in the following line, in the current coded
4481 character set, shall be placed in the character collation order between the previous and the
4482 following character in ascending order according to their coded character set values. An initial
4483 ellipsis shall be interpreted as if the preceding line specified the NUL character, and a trailing
4484 ellipsis as if the following line specified the highest coded character set value in the current
4485 coded character set. An ellipsis shall be treated as invalid if the preceding or following lines do
4486 not specify characters in the current coded character set. The use of the ellipsis symbol ties the
4487 definition to a specific coded character set and may preclude the definition from being portable
4488 between implementations.

4489 The symbol **UNDEFINED** shall be interpreted as including all coded character set values not
4490 specified explicitly or via the ellipsis symbol. Such characters shall be inserted in the character

collation order at the point indicated by the symbol, and in ascending order according to their coded character set values. If no **UNDEFINED** symbol is specified, and the current coded character set contains characters not specified in this section, the utility shall issue a warning message and place such characters at the end of the character collation order.

The optional operands for each collation-element shall be used to define the primary, secondary, or subsequent weights for the collating element. The first operand specifies the relative primary weight, the second the relative secondary weight, and so on. Two or more collation-elements can be assigned the same weight; they belong to the same “equivalence class” if they have the same primary weight. Collation shall behave as if, for each weight level, elements subject to **IGNORE** are removed, unless the **position** collation directive is specified for the corresponding level with the **order_start** keyword. Then each successive pair of elements shall be compared according to the relative weights for the elements. If the two strings compare equal, the process shall be repeated for the next weight level, up to the limit {**COLL_WEIGHTS_MAX**}.

Weights shall be expressed as characters (in any of the forms specified in [Section 7.3](#), on page 136), *<collating-symbol>*s, *<collating-element>*s, an ellipsis, or the special symbol **IGNORE**. A single character, a *<collating-symbol>*, or a *<collating-element>* shall represent the relative position in the character collating sequence of the character or symbol, rather than the character or characters themselves. Thus, rather than assigning absolute values to weights, a particular weight is expressed using the relative order value assigned to a collating element based on its order in the character collation sequence.

One-to-many mapping is indicated by specifying two or more concatenated characters or symbolic names. For example, if the *<eszet>* is given the string “*<s><s>*” as a weight, comparisons are performed as if all occurrences of the *<eszet>* are replaced by “*<s><s>*” (assuming that “*<s>*” has the collating weight “*<s>*”). If it is necessary to define *<eszet>* and “*<s><s>*” as an equivalence class, then a collating element must be defined for the string “*ss*”.

All characters specified via an ellipsis shall by default be assigned unique weights, equal to the relative order of characters. Characters specified via an explicit or implicit **UNDEFINED** special symbol shall by default be assigned the same primary weight (that is, they belong to the same equivalence class). An ellipsis symbol as a weight shall be interpreted to mean that each character in the sequence shall have unique weights, equal to the relative order of their character in the character collation sequence. The use of the ellipsis as a weight shall be treated as an error if the collating element is neither an ellipsis nor the special symbol **UNDEFINED**.

The special keyword **IGNORE** as a weight shall indicate that when strings are compared using the weights at the level where **IGNORE** is specified, the collating element shall be ignored; that is, as if the string did not contain the collating element. In regular expressions and pattern matching, all characters that are subject to **IGNORE** in their primary weight form an equivalence class.

An empty operand shall be interpreted as the collating element itself.

For example, the order statement:

```
<a>      <a>; <a>
```

is equal to:

```
<a>
```

An ellipsis can be used as an operand if the collating element was an ellipsis, and shall be interpreted as the value of each character defined by the ellipsis.

The collation order as defined in this section affects the interpretation of bracket expressions in regular expressions (see [Section 9.3.5](#), on page 184).

```

4537     For example:
4538     order_start    forward;backward
4539     UNDEFINED      IGNORE; IGNORE
4540     <LOW>
4541     <space>        <LOW>; <space>
4542     ...            <LOW>; ...
4543     <a>             <a>; <a>
4544     <a-acute>       <a>; <a-acute>
4545     <a-grave>       <a>; <a-grave>
4546     <A>            <a>; <A>
4547     <A-acute>       <a>; <A-acute>
4548     <A-grave>       <a>; <A-grave>
4549     <ch>           <ch>; <ch>
4550     <Ch>           <ch>; <Ch>
4551     <s>            <s>; <s>
4552     <eszet>        "<s><s>"; "<eszet><eszet>"
4553     order_end

```

4554 This example is interpreted as follows:

- 4555 1. The **UNDEFINED** means that all characters not specified in this definition (explicitly or
4556 via the ellipsis) shall be ignored for collation purposes.
- 4557 2. All characters between <space> and 'a' shall have the same primary equivalence class
4558 and individual secondary weights based on their ordinal encoded values.
- 4559 3. All characters based on the uppercase or lowercase character 'a' belong to the same
4560 primary equivalence class.
- 4561 4. The multi-character collating element <ch> is represented by the collating symbol <ch>
4562 and belongs to the same primary equivalence class as the multi-character collating
4563 element <Ch>.

4564 7.3.2.5 The order_end Keyword

4565 The collating order entries shall be terminated with an **order_end** keyword.

4566 7.3.2.6 LC_COLLATE Category in the POSIX Locale

4567 The collation sequence definition of the POSIX locale follows; the code listing depicts the
4568 *localedef* input.

```

4569 LC_COLLATE
4570 # This is the POSIX locale definition for the LC_COLLATE category.
4571 # The order is the same as in the ASCII codeset.
4572 order_start forward
4573 <NUL>
4574 <SOH>
4575 <STX>
4576 <ETX>
4577 <EOT>
4578 <ENQ>
4579 <ACK>
4580 <alert>
4581 <backspace>

```

4582 <tab>
4583 <newline>
4584 <vertical-tab>
4585 <form-feed>
4586 <carriage-return>
4587 <SO>
4588 <SI>
4589 <DLE>
4590 <DC1>
4591 <DC2>
4592 <DC3>
4593 <DC4>
4594 <NAK>
4595 <SYN>
4596 <ETB>
4597 <CAN>
4598
4599 <SUB>
4600 <ESC>
4601 <IS4>
4602 <IS3>
4603 <IS2>
4604 <IS1>
4605 <space>
4606 <exclamation-mark>
4607 <quotation-mark>
4608 <number-sign>
4609 <dollar-sign>
4610 <percent-sign>
4611 <ampersand>
4612 <apostrophe>
4613 <left-parenthesis>
4614 <right-parenthesis>
4615 <asterisk>
4616 <plus-sign>
4617 <comma>
4618 <hyphen>
4619 <period>
4620 <slash>
4621 <zero>
4622 <one>
4623 <two>
4624 <three>
4625 <four>
4626 <five>
4627 <six>
4628 <seven>
4629 <eight>
4630 <nine>
4631 <colon>
4632 <semicolon>
4633 <less-than-sign>
4634 <equals-sign>

4635	<greater-than-sign>
4636	<question-mark>
4637	<commercial-at>
4638	<A>
4639	
4640	<C>
4641	<D>
4642	<E>
4643	<F>
4644	<G>
4645	<H>
4646	<I>
4647	<J>
4648	<K>
4649	<L>
4650	<M>
4651	<N>
4652	<O>
4653	<P>
4654	<Q>
4655	<R>
4656	<S>
4657	<T>
4658	<U>
4659	<V>
4660	<W>
4661	<X>
4662	<Y>
4663	<Z>
4664	<left-square-bracket>
4665	<backslash>
4666	<right-square-bracket>
4667	<circumflex>
4668	<underscore>
4669	<grave-accent>
4670	<a>
4671	
4672	<c>
4673	<d>
4674	<e>
4675	<f>
4676	<g>
4677	<h>
4678	<i>
4679	<j>
4680	<k>
4681	<l>
4682	<m>
4683	<n>
4684	<o>
4685	<p>
4686	<q>
4687	<r>

```

4688      <s>
4689      <t>
4690      <u>
4691      <v>
4692      <w>
4693      <x>
4694      <y>
4695      <z>
4696      <left-curly-bracket>
4697      <vertical-line>
4698      <right-curly-bracket>
4699      <tilde>
4700      <DEL>
4701      order_end
4702      #
4703      END LC_COLLATE

```

4704 7.3.3 LC_MONETARY

4705 The *LC_MONETARY* category shall define the rules and symbols that are used to format
 4706 monetary numeric information.

4707 This information is available through the *localeconv()* function and is used by the *strfmon()*
 4708 function.

4709 Some of the information is also available in an alternative form via the *nl_langinfo()* function
 4710 (see CRNCYSTR in [<langinfo.h>](#)).

4711 The following items are defined in this category of the locale. The item names are the keywords
 4712 recognized by the *localedef* utility when defining a locale. They are also similar to the member
 4713 names of the *lconv* structure defined in [<locale.h>](#); see [<locale.h>](#) for the exact symbols in the
 4714 header. The *localeconv()* function returns {CHAR_MAX} for unspecified integer items and the
 4715 empty string (" ") for unspecified or size zero string items.

4716 In a locale definition file, the operands are strings, formatted as indicated by the grammar in
 4717 [Section 7.4](#) (on page 165). For some keywords, the strings can contain only integers. Keywords
 4718 that are not provided, string values set to the empty string (" "), or integer keywords set to -1,
 4719 are used to indicate that the value is not available in the locale. The following keywords shall be
 4720 recognized:

4721 **copy** Specify the name of an existing locale which shall be used as the
 4722 definition of this category. If this keyword is specified, no other keyword
 4723 shall be specified.

4724 **Note:** This is a *localedef* utility keyword, unavailable through *localeconv()*.

4725 **int_curr_symbol** The international currency symbol. The operand shall be a four-character
 4726 string, with the first three characters containing the alphabetic
 4727 international currency symbol. The international currency symbol should
 4728 be chosen in accordance with those specified in the ISO 4217 standard.
 4729 The fourth character shall be the character used to separate the
 4730 international currency symbol from the monetary quantity.

4731 **currency_symbol** The string that shall be used as the local currency symbol.

4732	mon_decimal_point	The operand is a string containing the symbol that shall be used as the decimal delimiter (radix character) in monetary formatted quantities.
4733		
4734	mon_thousands_sep	The operand is a string containing the symbol that shall be used as a separator for groups of digits to the left of the decimal delimiter in formatted monetary quantities.
4735		
4736		
4737	mon_grouping	Define the size of each group of digits in formatted monetary quantities. The operand is a sequence of integers separated by <semicolon> characters. Each integer specifies the number of digits in each group, with the initial integer defining the size of the group immediately preceding the decimal delimiter, and the following integers defining the preceding groups. If the last integer is not -1, then the size of the previous group (if any) shall be repeatedly used for the remainder of the digits. If the last integer is -1, then no further grouping shall be performed.
4738		
4739		
4740		
4741		
4742		
4743		
4744		
4745	positive_sign	A string that shall be used to indicate a non-negative-valued formatted monetary quantity.
4746		
4747	negative_sign	A string that shall be used to indicate a negative-valued formatted monetary quantity.
4748		
4749	int_frac_digits	An integer representing the number of fractional digits (those to the right of the decimal delimiter) to be written in a formatted monetary quantity using int_curr_symbol .
4750		
4751		
4752	frac_digits	An integer representing the number of fractional digits (those to the right of the decimal delimiter) to be written in a formatted monetary quantity using currency_symbol .
4753		
4754		
4755	p_cs_precedes	An integer set to 1 if the currency_symbol precedes the value for a monetary quantity with a non-negative value, and set to 0 if the symbol succeeds the value.
4756		
4757		
4758	p_sep_by_space	Set to a value indicating the separation of the currency_symbol , the sign string, and the value for a non-negative formatted monetary quantity.
4759		
4760		The values of p_sep_by_space , n_sep_by_space , int_p_sep_by_space , and int_n_sep_by_space are interpreted according to the following:
4761		
4762		0 No <space> separates the currency symbol and value.
4763		1 If the currency symbol and sign string are adjacent, a <space> separates them from the value; otherwise, a <space> separates the currency symbol from the value.
4764		
4765		
4766		2 If the currency symbol and sign string are adjacent, a <space> separates them; otherwise, a <space> separates the sign string from the value.
4767		
4768		
4769	n_cs_precedes	An integer set to 1 if the currency_symbol precedes the value for a monetary quantity with a negative value, and set to 0 if the symbol succeeds the value.
4770		
4771		
4772	n_sep_by_space	Set to a value indicating the separation of the currency_symbol , the sign string, and the value for a negative formatted monetary quantity.
4773		
4774	p_sign_posn	An integer set to a value indicating the positioning of the positive_sign for a monetary quantity with a non-negative value. The following integer values shall be recognized for int_n_sign_posn , int_p_sign_posn ,
4775		
4776		

4777		n_sign_posn , and p_sign_posn :
4778	0	Parentheses enclose the quantity and the currency_symbol .
4779	1	The sign string precedes the quantity and the currency_symbol .
4780	2	The sign string succeeds the quantity and the currency_symbol .
4781	3	The sign string precedes the currency_symbol .
4782	4	The sign string succeeds the currency_symbol .
4783	n_sign_posn	An integer set to a value indicating the positioning of the negative_sign for a negative formatted monetary quantity.
4784		
4785	int_p_cs_precedes	An integer set to 1 if the int_curr_symbol precedes the value for a monetary quantity with a non-negative value, and set to 0 if the symbol succeeds the value.
4786		
4787		
4788	int_n_cs_precedes	An integer set to 1 if the int_curr_symbol precedes the value for a monetary quantity with a negative value, and set to 0 if the symbol succeeds the value.
4789		
4790		
4791	int_p_sep_by_space	Set to a value indicating the separation of the int_curr_symbol , the sign string, and the value for a non-negative internationally formatted monetary quantity.
4792		
4793		
4794	int_n_sep_by_space	Set to a value indicating the separation of the int_curr_symbol , the sign string, and the value for a negative internationally formatted monetary quantity.
4795		
4796		
4797	int_p_sign_posn	An integer set to a value indicating the positioning of the positive_sign for a positive monetary quantity formatted with the international format.
4798		
4799	int_n_sign_posn	An integer set to a value indicating the positioning of the negative_sign for a negative monetary quantity formatted with the international format.
4800		

4801 7.3.3.1 LC_MONETARY Category in the POSIX Locale

4802 The monetary formatting definitions for the POSIX locale follow; the code listing depicting the
 4803 *localedef* input, the table representing the same information with the addition of *localeconv()* and
 4804 *nl_langinfo()* formats. All values are unspecified in the POSIX locale.

```

4805 LC_MONETARY
4806 # This is the POSIX locale definition for
4807 # the LC_MONETARY category.
4808 #
4809 int_curr_symbol      ""
4810 currency_symbol      ""
4811 mon_decimal_point    ""
4812 mon_thousands_sep   ""
4813 mon_grouping         -1
4814 positive_sign        ""
4815 negative_sign        ""
4816 int_frac_digits      -1
4817 frac_digits          -1
4818 p_cs_precedes        -1
4819 p_sep_by_space       -1
4820 n_cs_precedes        -1

```

```

4821     n_sep_by_space      -1
4822     p_sign_posn         -1
4823     n_sign_posn         -1
4824     int_p_cs_precedes   -1
4825     int_p_sep_by_space  -1
4826     int_n_cs_precedes   -1
4827     int_n_sep_by_space  -1
4828     int_p_sign_posn     -1
4829     int_n_sign_posn     -1
4830     #
4831     END LC_MONETARY

```

	Item	langinfo Constant	POSIX Locale Value	localeconv() Value	localedef Value
4834	int_curr_symbol	—	N/A	" "	" "
4835	currency_symbol	CRNCYSTR	N/A	" "	" "
4836	mon_decimal_point	—	N/A	" "	" "
4837	mon_thousands_sep	—	N/A	" "	" "
4838	mon_grouping	—	N/A	" "	-1
4839	positive_sign	—	N/A	" "	" "
4840	negative_sign	—	N/A	" "	" "
4841	int_frac_digits	—	N/A	{CHAR_MAX}	-1
4842	frac_digits	—	N/A	{CHAR_MAX}	-1
4843	p_cs_precedes	CRNCYSTR	N/A	{CHAR_MAX}	-1
4844	p_sep_by_space	—	N/A	{CHAR_MAX}	-1
4845	n_cs_precedes	CRNCYSTR	N/A	{CHAR_MAX}	-1
4846	n_sep_by_space	—	N/A	{CHAR_MAX}	-1
4847	p_sign_posn	—	N/A	{CHAR_MAX}	-1
4848	n_sign_posn	—	N/A	{CHAR_MAX}	-1
4849	int_p_cs_precedes	—	N/A	{CHAR_MAX}	-1
4850	int_p_sep_by_space	—	N/A	{CHAR_MAX}	-1
4851	int_n_cs_precedes	—	N/A	{CHAR_MAX}	-1
4852	int_n_sep_by_space	—	N/A	{CHAR_MAX}	-1
4853	int_p_sign_posn	—	N/A	{CHAR_MAX}	-1
4854	int_n_sign_posn	—	N/A	{CHAR_MAX}	-1

4855 The entry N/A indicates that the value is not available in the POSIX locale.

4856 7.3.4 LC_NUMERIC

4857 The *LC_NUMERIC* category shall define the rules and symbols that are used to format non-
4858 monetary numeric information. This information is available through the *localeconv()* function.

4859 Some of the information is also available in an alternative form via the *nl_langinfo()* function.

4860 The following items are defined in this category of the locale. The item names are the keywords
4861 recognized by the *localedef* utility when defining a locale. They are also similar to the member
4862 names of the **lconv** structure defined in [<locale.h>](#); see [<locale.h>](#) for the exact symbols in the
4863 header. The *localeconv()* function returns {CHAR_MAX} for unspecified integer items and the
4864 empty string (" ") for unspecified or size zero string items.

4865 In a locale definition file, the operands are strings, formatted as indicated by the grammar in
4866 [Section 7.4](#) (on page 165). For some keywords, the strings can only contain integers. Keywords

that are not provided, string values set to the empty string (" "), or integer keywords set to -1, shall be used to indicate that the value is not available in the locale. The following keywords shall be recognized:

copy Specify the name of an existing locale which shall be used as the definition of this category. If this keyword is specified, no other keyword shall be specified.

Note: This is a *localedef* utility keyword, unavailable through *localeconv()*.

decimal_point The operand is a string containing the symbol that shall be used as the decimal delimiter (radix character) in numeric, non-monetary formatted quantities. This keyword cannot be omitted and cannot be set to the empty string. In contexts where standards limit the **decimal_point** to a single byte, the result of specifying a multi-byte operand shall be unspecified.

thousands_sep The operand is a string containing the symbol that shall be used as a separator for groups of digits to the left of the decimal delimiter in numeric, non-monetary formatted monetary quantities. In contexts where standards limit the **thousands_sep** to a single byte, the result of specifying a multi-byte operand shall be unspecified.

grouping Define the size of each group of digits in formatted non-monetary quantities. The operand is a sequence of integers separated by <semicolon> characters. Each integer specifies the number of digits in each group, with the initial integer defining the size of the group immediately preceding the decimal delimiter, and the following integers defining the preceding groups. If the last integer is not -1, then the size of the previous group (if any) shall be repeatedly used for the remainder of the digits. If the last integer is -1, then no further grouping shall be performed.

7.3.4.1 LC_NUMERIC Category in the POSIX Locale

The non-monetary numeric formatting definitions for the POSIX locale follow; the code listing depicting the *localedef* input, the table representing the same information with the addition of *localeconv()* values, and *nl_langinfo()* constants.

```
LC_NUMERIC
# This is the POSIX locale definition for
# the LC_NUMERIC category.
#
decimal_point      "<period>"
thousands_sep      ""
grouping           -1
#
END LC_NUMERIC
```

Item	langinfo Constant	POSIX Locale Value	localeconv() Value	localedef Value
decimal_point	RADIXCHAR	"."	"."	"."
thousands_sep	THOUSEP	N/A	" "	" "
grouping	—	N/A	" "	-1

The entry N/A indicates that the value is not available in the POSIX locale.

7.3.5 LC_TIME

The *LC_TIME* category shall define the interpretation of the conversion specifications supported by the *date* utility and shall affect the behavior of the *strftime()*, *wcsftime()*, *strptime()*, and *nl_langinfo()* functions. Since the interfaces for C-language access and locale definition differ significantly, they are described separately.

7.3.5.1 LC_TIME Locale Definition

In a locale definition, the following mandatory keywords shall be recognized:

copy	Specify the name of an existing locale which shall be used as the definition of this category. If this keyword is specified, no other keyword shall be specified.
abday	Define the abbreviated weekday names, corresponding to the %a conversion specification (conversion specification in the <i>strftime()</i> , <i>wcsftime()</i> , and <i>strptime()</i> functions). The operand shall consist of seven &ltsemicolon>-separated strings, each surrounded by double-quotes. The first string shall be the abbreviated name of the day corresponding to Sunday, the second the abbreviated name of the day corresponding to Monday, and so on.
day	Define the full weekday names, corresponding to the %A conversion specification. The operand shall consist of seven &ltsemicolon>-separated strings, each surrounded by double-quotes. The first string is the full name of the day corresponding to Sunday, the second the full name of the day corresponding to Monday, and so on.
abmon	Define the abbreviated month names, corresponding to the %b conversion specification. The operand shall consist of twelve &ltsemicolon>-separated strings, each surrounded by double-quotes. The first string shall be the abbreviated name of the first month of the year (January), the second the abbreviated name of the second month, and so on.
mon	Define the full month names, corresponding to the %B conversion specification. The operand shall consist of twelve &ltsemicolon>-separated strings, each surrounded by double-quotes. The first string shall be the full name of the first month of the year (January), the second the full name of the second month, and so on.
d_t_fmt	Define the appropriate date and time representation, corresponding to the %c conversion specification. The operand shall consist of a string containing any combination of characters and conversion specifications. In addition, the string can contain escape sequences defined in the table in Table 5-1 (on page 121) ('\\', '\a', '\b', '\f', '\n', '\r', '\t', '\v').
d_fmt	Define the appropriate date representation, corresponding to the %x conversion specification. The operand shall consist of a string containing any combination of characters and conversion specifications. In addition, the string can contain escape sequences defined in Table 5-1 (on page 121).
t_fmt	Define the appropriate time representation, corresponding to the %X conversion specification. The operand shall consist of a string containing any combination of characters and conversion specifications. In addition, the string can contain escape sequences defined in Table 5-1 (on page 121).

4953	am_pm	Define the appropriate representation of the <i>ante-meridiem</i> and <i>post-meridiem</i> strings, corresponding to the %p conversion specification. The operand shall consist of two strings, separated by a <semicolon>, each surrounded by double-quotes. The first string shall represent the <i>ante-meridiem</i> designation, the last string the <i>post-meridiem</i> designation.
4954		
4955		
4956		
4957		
4958	t_fmt_ampm	Define the appropriate time representation in the 12-hour clock format with am_pm , corresponding to the %r conversion specification. The operand shall consist of a string and can contain any combination of characters and conversion specifications. If the string is empty, the 12-hour format is not supported in the locale.
4959		
4960		
4961		
4962		
4963	era	Define how years are counted and displayed for each era in a locale. The operand shall consist of <semicolon>-separated strings. Each string shall be an era description segment with the format:
4964		
4965		
4966		<i>direction:offset:start_date:end_date:era_name:era_format</i>
4967		according to the definitions below. There can be as many era description segments as are necessary to describe the different eras.
4968		
4969	Note:	The start of an era might not be the earliest point in the era—it may be the latest. For example, the Christian era BC starts on the day before January 1, AD 1, and increases with earlier time.
4970		
4971		
4972	<i>direction</i>	Either a '+' or a '-' character. The '+' character shall indicate that years closer to the <i>start_date</i> have lower numbers than those closer to the <i>end_date</i> . The '-' character shall indicate that years closer to the <i>start_date</i> have higher numbers than those closer to the <i>end_date</i> .
4973		
4974		
4975		
4976		
4977	<i>offset</i>	The number of the year closest to the <i>start_date</i> in the era, corresponding to the %EY conversion specification.
4978		
4979	<i>start_date</i>	A date in the form <i>yyyy/mm/dd</i> , where <i>yyyy</i> , <i>mm</i> , and <i>dd</i> are the year, month, and day numbers respectively of the start of the era. Years prior to AD 1 shall be represented as negative numbers.
4980		
4981		
4982	<i>end_date</i>	The ending date of the era, in the same format as the <i>start_date</i> , or one of the two special values "-*" or "+*". The value "-*" shall indicate that the ending date is the beginning of time. The value "+*" shall indicate that the ending date is the end of time.
4983		
4984		
4985		
4986	<i>era_name</i>	A string representing the name of the era, corresponding to the %EC conversion specification.
4987		
4988	<i>era_format</i>	A string for formatting the year in the era, corresponding to the %EY conversion specification.
4989		
4990	era_d_fmt	Define the format of the date in alternative era notation, corresponding to the %Ex conversion specification.
4991		
4992	era_t_fmt	Define the locale's appropriate alternative time format, corresponding to the %EX conversion specification.
4993		
4994	era_d_t_fmt	Define the locale's appropriate alternative date and time format, corresponding to the %Ec conversion specification.
4995		
4996	alt_digits	Define alternative symbols for digits, corresponding to the %O modified conversion specification. The operand shall consist of <semicolon>-separated strings, each surrounded by double-quotes. The first string shall be the
4997		
4998		

alternative symbol corresponding with zero, the second string the symbol corresponding with one, and so on. Up to 100 alternative symbol strings can be specified. The %O modifier shall indicate that the string corresponding to the value specified via the conversion specification shall be used instead of the value.

5004 7.3.5.2 LC_TIME C-Language Access

5005 The following constants used to identify items of *langinfo* data can be used as arguments to the
5006 *nl_langinfo()* function to access information in the *LC_TIME* category. These constants are
5007 defined in the **<langinfo.h>** header.

5008 **ABDAY_x** The abbreviated weekday names (for example, Sun), where *x* is a number
5009 from 1 to 7.

5010 **DAY_x** The full weekday names (for example, Sunday), where *x* is a number from 1 to
5011 7.

5012 **ABMON_x** The abbreviated month names (for example, Jan), where *x* is a number from 1
5013 to 12.

5014 **MON_x** The full month names (for example, January), where *x* is a number from 1 to
5015 12.

5016 **D_T_FMT** The appropriate date and time representation.

5017 **D_FMT** The appropriate date representation.

5018 **T_FMT** The appropriate time representation.

5019 **AM_STR** The appropriate ante-meridiem affix.

5020 **PM_STR** The appropriate post-meridiem affix.

5021 **T_FMT_AMPM** The appropriate time representation in the 12-hour clock format with
5022 **AM_STR** and **PM_STR**.

5023 **ERA** The era description segments, which describe how years are counted and
5024 displayed for each era in a locale. Each era description segment shall have the
5025 format:

5026 *direction:offset:start_date:end_date:era_name:era_format*

5027 according to the definitions below. There can be as many era description
5028 segments as are necessary to describe the different eras. Era description
5029 segments are separated by <semicolon> characters.

5030 *direction* Either a '+' or a '-' character. The '+' character shall indicate
5031 that years closer to the *start_date* have lower numbers than those
5032 closer to the *end_date*. The '-' character shall indicate that years
5033 closer to the *start_date* have higher numbers than those closer to
5034 the *end_date*.

5035 *offset* The number of the year closest to the *start_date* in the era.

5036 *start_date* A date in the form *yyyy/mm/dd*, where *yyyy*, *mm*, and *dd* are the
5037 year, month, and day numbers respectively of the start of the era.
5038 Years prior to AD 1 shall be represented as negative numbers.

5039	<i>end_date</i>	The ending date of the era, in the same format as the <i>start_date</i> ,
5040		or one of the two special values "-*" or "+*". The value "-*" shall indicate that the ending date is the beginning of time. The
5041		value "+*" shall indicate that the ending date is the end of time.
5042		
5043	<i>era_name</i>	The era, corresponding to the %EC conversion specification.
5044	<i>era_format</i>	The format of the year in the era, corresponding to the %EY
5045		conversion specification.
5046	ERA_D_FMT	The era date format.
5047	ERA_T_FMT	The locale's appropriate alternative time format, corresponding to the %EX
5048		conversion specification.
5049	ERA_D_T_FMT	The locale's appropriate alternative date and time format, corresponding to
5050		the %Ec conversion specification.
5051	ALT_DIGITS	The alternative symbols for digits, corresponding to the %O conversion
5052		specification modifier. The value consists of <semicolon>-separated symbols.
5053		The first is the alternative symbol corresponding to zero, the second is the
5054		symbol corresponding to one, and so on. Up to 100 alternative symbols may
5055		be specified.

5056 7.3.5.3 LC_TIME Category in the POSIX Locale

5057 The *LC_TIME* category definition of the POSIX locale follows; the code listing depicts the
 5058 *localedef* input; the table represents the same information with the addition of *localedef* keywords,
 5059 conversion specifiers used by the *date* utility and the *strftime()*, *wcsftime()*, and *strptime()*
 5060 functions, and *nl_langinfo()* constants.

```

5061 LC_TIME
5062 # This is the POSIX locale definition for
5063 # the LC_TIME category.
5064 #
5065 # Abbreviated weekday names (%a)
5066 abday      "<S><u><n>"; "<M><o><n>"; "<T><u><e>"; "<W><e><d>"; \
5067            "<T><h><u>"; "<F><r><i>"; "<S><a><t>"
5068 #
5069 # Full weekday names (%A)
5070 day        "<S><u><n><d><a><y>"; "<M><o><n><d><a><y>"; \
5071            "<T><u><e><s><d><a><y>"; "<W><e><d><n><e><s><d><a><y>"; \
5072            "<T><h><u><r><s><d><a><y>"; "<F><r><i><d><a><y>"; \
5073            "<S><a><t><u><r><d><a><y>"
5074 #
5075 # Abbreviated month names (%b)
5076 abmon      "<J><a><n>"; "<F><e><b>"; "<M><a><r>"; \
5077            "<A><p><r>"; "<M><a><y>"; "<J><u><n>"; \
5078            "<J><u><l>"; "<A><u><g>"; "<S><e><p>"; \
5079            "<O><c><t>"; "<N><o><v>"; "<D><e><c>"
5080 #
5081 # Full month names (%B)
5082 mon        "<J><a><n><u><a><r><y>"; "<F><e><b><r><u><a><r><y>"; \
5083            "<M><a><r><c><h>"; "<A><p><r><i><l>"; \
5084            "<M><a><y>"; "<J><u><n><e>"; \
5085            "<J><u><l><y>"; "<A><u><g><u><s><t>"; \

```



```

5086         "<S><e><p><t><e><m><b><e><r>"; "<O><c><t><o><b><e><r>"; \
5087         "<N><o><v><e><m><b><e><r>"; "<D><e><c><e><m><b><e><r>"
5088     #
5089     # Equivalent of AM/PM (%p)          "AM"; "PM"
5090     am_pm          "<A><M>"; "<P><M>"
5091     #
5092     # Appropriate date and time representation (%c)
5093     #      "%a %b %e %H:%M:%S %Y"
5094     d_t_fmt        "<percent-sign><a><space><percent-sign><b>\
5095     <space><percent-sign><e><space><percent-sign><H>\
5096     <colon><percent-sign><M><colon><percent-sign><S>\
5097     <space><percent-sign><Y>"
5098     #
5099     # Appropriate date representation (%x)    "%m/%d/%y"
5100     d_fmt          "<percent-sign><m><slash><percent-sign><d>\
5101     <slash><percent-sign><y>"
5102     #
5103     # Appropriate time representation (%X)    "%H:%M:%S"
5104     t_fmt          "<percent-sign><H><colon><percent-sign><M>\
5105     <colon><percent-sign><S>"
5106     #
5107     # Appropriate 12-hour time representation (%r) "%I:%M:%S %p"
5108     t_fmt_ampm     "<percent-sign><I><colon><percent-sign><M><colon>\
5109     <percent-sign><S><space><percent-sign><p>"
5110     #
5111     END LC_TIME

```

	localedef Keyword	langinfo Constant	Conversion Specification	POSIX Locale Value
5112	d_t_fmt	D_T_FMT	%c	"%a %b %e %H:%M:%S %Y"
5113	d_fmt	D_FMT	%x	"%m/%d/%y"
5114	t_fmt	T_FMT	%X	"%H:%M:%S"
5115	am_pm	AM_STR	%p	"AM"
5116	am_pm	PM_STR	%p	"PM"
5117	t_fmt_ampm	T_FMT_AMP	%r	"%I:%M:%S %p"
5118	day	DAY_1	%A	"Sunday"
5119	day	DAY_2	%A	"Monday"
5120	day	DAY_3	%A	"Tuesday"
5121	day	DAY_4	%A	"Wednesday"
5122	day	DAY_5	%A	"Thursday"
5123	day	DAY_6	%A	"Friday"
5124	day	DAY_7	%A	"Saturday"
5125	abday	ABDAY_1	%a	"Sun"
5126	abday	ABDAY_2	%a	"Mon"
5127	abday	ABDAY_3	%a	"Tue"
5128	abday	ABDAY_4	%a	"Wed"
5129	abday	ABDAY_5	%a	"Thu"
5130	abday	ABDAY_6	%a	"Fri"
5131	abday	ABDAY_7	%a	"Sat"
5132	mon	MON_1	%B	"January"
5133	mon	MON_2	%B	"February"
5134	mon	MON_3	%B	"March"

localedef Keyword	langinfo Constant	Conversion Specification	POSIX Locale Value
mon	MON_4	%B	"April"
mon	MON_5	%B	"May"
mon	MON_6	%B	"June"
mon	MON_7	%B	"July"
mon	MON_8	%B	"August"
mon	MON_9	%B	"September"
mon	MON_10	%B	"October"
mon	MON_11	%B	"November"
mon	MON_12	%B	"December"
abmon	ABMON_1	%b	"Jan"
abmon	ABMON_2	%b	"Feb"
abmon	ABMON_3	%b	"Mar"
abmon	ABMON_4	%b	"Apr"
abmon	ABMON_5	%b	"May"
abmon	ABMON_6	%b	"Jun"
abmon	ABMON_7	%b	"Jul"
abmon	ABMON_8	%b	"Aug"
abmon	ABMON_9	%b	"Sep"
abmon	ABMON_10	%b	"Oct"
abmon	ABMON_11	%b	"Nov"
abmon	ABMON_12	%b	"Dec"
era	ERA	%EC, %Ey, %EY	N/A
era_d_fmt	ERA_D_FMT	%Ex	N/A
era_t_fmt	ERA_T_FMT	%EX	N/A
era_d_t_fmt	ERA_D_T_FMT	%Ec	N/A
alt_digits	ALT_DIGITS	%O	N/A

The entry N/A indicates the value is not available in the POSIX locale.

7.3.6 LC_MESSAGES

The *LC_MESSAGES* category shall define the format and values used by various utilities for affirmative and negative responses. This information is available through the *nl_langinfo()* function.

The message catalog used by the standard utilities and selected by the *catopen()* function shall be determined by the setting of *NLSPATH*; see [Chapter 8](#) (on page 173). The *LC_MESSAGES* category can be specified as part of an *NLSPATH* substitution field.

The following keywords shall be recognized as part of the locale definition file.

copy Specify the name of an existing locale which shall be used as the definition of this category. If this keyword is specified, no other keyword shall be specified.

Note: This is a *localedef* keyword, unavailable through *nl_langinfo()*.

yesexpr The operand consists of an extended regular expression (see [Section 9.4](#), on page 188) that describes acceptable affirmative responses to a question expecting an affirmative or negative response.

noexpr The operand consists of an extended regular expression that describes acceptable negative responses to a question expecting an affirmative or negative response.

5182 7.3.6.1 LC_MESSAGES Category in the POSIX Locale

5183 The format and values for affirmative and negative responses of the POSIX locale follow; the
 5184 code listing depicting the *localedef* input, the table representing the same information with the
 5185 addition of *nl_langinfo()* constants.

```

5186 LC_MESSAGES
5187 # This is the POSIX locale definition for
5188 # the LC_MESSAGES category.
5189 #
5190 yesexpr "<circumflex><left-square-bracket><y><Y><right-square-bracket>"
5191 #
5192 noexpr  "<circumflex><left-square-bracket><n><N><right-square-bracket>"
5193 #
5194 END LC_MESSAGES

```

localedef Keyword	langinfo Constant	POSIX Locale Value
yesexpr	YESEXPR	" ^ [yY] "
noexpr	NOEXPR	" ^ [nN] "

5198 7.4 Locale Definition Grammar

5199 The grammar and lexical conventions in this section shall together describe the syntax for the
 5200 locale definition source. The general conventions for this style of grammar are described in XCU
 5201 [Section 1.3](#) (on page 2309). The grammar shall take precedence over the text in this chapter.

5202 7.4.1 Locale Lexical Conventions

5203 The lexical conventions for the locale definition grammar are described in this section.

5204 The following tokens shall be processed (in addition to those string constants shown in the
 5205 grammar):

5206	LOC_NAME	A string of characters representing the name of a locale.
5207	CHAR	Any single character.
5208	NUMBER	A decimal number, represented by one or more decimal digits.
5209	COLLSYMBOL	A symbolic name, enclosed between angle brackets. The string cannot duplicate any charmap symbol defined in the current charmap (if any), or a COLLELEMENT symbol.
5210		
5211		
5212	COLLELEMENT	A symbolic name, enclosed between angle brackets, which cannot duplicate either any charmap symbol or a COLLSYMBOL symbol.
5213		
5214	CHARCLASS	A string of alphanumeric characters from the portable character set, the first of which is not a digit, consisting of at least one and at most {CHARCLASS_NAME_MAX} bytes, and optionally surrounded by double-quotes.
5215		
5216		
5217		
5218	CHARSYMBOL	A symbolic name, enclosed between angle brackets, from the current charmap (if any).
5219		

5220	OCTAL_CHAR	One or more octal representations of the encoding of each byte in a
5221		single character. The octal representation consists of an escape
5222		character (normally a <backslash>) followed by two or more octal
5223		digits.
5224	HEX_CHAR	One or more hexadecimal representations of the encoding of each
5225		byte in a single character. The hexadecimal representation consists of
5226		an escape character followed by the constant <i>x</i> and two or more
5227		hexadecimal digits.
5228	DECIMAL_CHAR	One or more decimal representations of the encoding of each byte in
5229		a single character. The decimal representation consists of an escape
5230		character followed by a character 'd' and two or more decimal
5231		digits.
5232	ELLIPSIS	The string ". . .".
5233	EXTENDED_REG_EXP	An extended regular expression as defined in the grammar in Section
5234		9.5 (on page 191).
5235	EOL	The line termination character <newline>.

5236 7.4.2 Locale Grammar

5237 This section presents the grammar for the locale definition.

5238	%token	LOC_NAME
5239	%token	CHAR
5240	%token	NUMBER
5241	%token	COLLSYMBOL COLLELEMENT
5242	%token	CHARSYMBOL OCTAL_CHAR HEX_CHAR DECIMAL_CHAR
5243	%token	ELLIPSIS
5244	%token	EXTENDED_REG_EXP
5245	%token	EOL
5246	%start	locale_definition
5247	%%	
5248	locale_definition	: global_statements locale_categories
5249		locale_categories
5250		;
5251	global_statements	: global_statements symbol_redefine
5252		symbol_redefine
5253		;
5254	symbol_redefine	: 'escape_char' CHAR EOL
5255		'comment_char' CHAR EOL
5256		;
5257	locale_categories	: locale_categories locale_category
5258		locale_category
5259		;
5260	locale_category	: lc_ctype lc_collate lc_messages
5261		lc_monetary lc_numeric lc_time
5262		;

```

5263      /* The following grammar rules are common to all categories */
5264      char_list          : char_list char_symbol
5265                          | char_symbol
5266                          ;
5267      char_symbol        : CHAR | CHARSYMBOL
5268                          | OCTAL_CHAR | HEX_CHAR | DECIMAL_CHAR
5269                          ;
5270      elem_list          : elem_list char_symbol
5271                          | elem_list COLLSYMBOL
5272                          | elem_list COLLELEMENT
5273                          | char_symbol
5274                          | COLLSYMBOL
5275                          | COLLELEMENT
5276                          ;
5277      symb_list          : symb_list COLLSYMBOL
5278                          | COLLSYMBOL
5279                          ;
5280      locale_name        : LOC_NAME
5281                          | '""' LOC_NAME '""'
5282                          ;
5283      /* The following is the LC_CTYPE category grammar */
5284      lc_ctype           : ctype_hdr ctype_keywords          ctype_tlr
5285                          | ctype_hdr 'copy' locale_name EOL ctype_tlr
5286                          ;
5287      ctype_hdr          : 'LC_CTYPE' EOL
5288                          ;
5289      ctype_keywords     : ctype_keywords ctype_keyword
5290                          | ctype_keyword
5291                          ;
5292      ctype_keyword      : charclass_keyword charclass_list EOL
5293                          | charconv_keyword charconv_list EOL
5294                          | 'charclass' charclass_namelist EOL
5295                          ;
5296      charclass_namelist : charclass_namelist ';' CHARCLASS
5297                          | CHARCLASS
5298                          ;
5299      charclass_keyword  : 'upper' | 'lower' | 'alpha' | 'digit'
5300                          | 'punct' | 'xdigit' | 'space' | 'print'
5301                          | 'graph' | 'blank' | 'cntrl' | 'alnum'
5302                          | CHARCLASS
5303                          ;
5304      charclass_list     : charclass_list ';' char_symbol
5305                          | charclass_list ';' ELLIPSIS ';' char_symbol
5306                          | char_symbol
5307                          ;
5308      charconv_keyword   : 'toupper'

```

```

5309         | 'tolower'
5310         ;
5311     charconv_list      : charconv_list ';' charconv_entry
5312         | charconv_entry
5313         ;
5314     charconv_entry     : '(' char_symbol ',' char_symbol ')'
5315         ;
5316     ctype_tlr         : 'END' 'LC_CTYPE' EOL
5317         ;
5318     /* The following is the LC_COLLATE category grammar */
5319     lc_collate         : collate_hdr collate_keywords      collate_tlr
5320         | collate_hdr 'copy' locale_name EOL collate_tlr
5321         ;
5322     collate_hdr        : 'LC_COLLATE' EOL
5323         ;
5324     collate_keywords   :          order_statements
5325         | opt_statements order_statements
5326         ;
5327     opt_statements     : opt_statements collating_symbols
5328         | opt_statements collating_elements
5329         | collating_symbols
5330         | collating_elements
5331         ;
5332     collating_symbols  : 'collating-symbol' COLLSYMBOL EOL
5333         ;
5334     collating_elements : 'collating-element' COLLELEMENT
5335         | 'from' '"' elem_list '"' EOL
5336         ;
5337     order_statements   : order_start collation_order order_end
5338         ;
5339     order_start        : 'order_start' EOL
5340         | 'order_start' order_opts EOL
5341         ;
5342     order_opts         : order_opts ';' order_opt
5343         | order_opt
5344         ;
5345     order_opt          : order_opt ',' opt_word
5346         | opt_word
5347         ;
5348     opt_word           : 'forward' | 'backward' | 'position'
5349         ;
5350     collation_order    : collation_order collation_entry
5351         | collation_entry
5352         ;

```

```

5353      collation_entry      : COLLSYMBOL EOL
5354                          | collation_element weight_list EOL
5355                          | collation_element                EOL
5356                          ;
5357      collation_element     : char_symbol
5358                          | COLLELEMENT
5359                          | ELLIPSIS
5360                          | 'UNDEFINED'
5361                          ;
5362      weight_list           : weight_list ';' weight_symbol
5363                          | weight_list ';'
5364                          | weight_symbol
5365                          ;
5366      weight_symbol         : /* empty */
5367                          | char_symbol
5368                          | COLLSYMBOL
5369                          | '"' elem_list '"'
5370                          | '"' symb_list '"'
5371                          | ELLIPSIS
5372                          | 'IGNORE'
5373                          ;
5374      order_end             : 'order_end' EOL
5375                          ;
5376      collate_tlr           : 'END' 'LC_COLLATE' EOL
5377                          ;
5378      /* The following is the LC_MESSAGES category grammar */
5379      lc_messages            : messages_hdr messages_keywords      messages_tlr
5380                          | messages_hdr 'copy' locale_name EOL messages_tlr
5381                          ;
5382      messages_hdr           : 'LC_MESSAGES' EOL
5383                          ;
5384      messages_keywords      : messages_keywords messages_keyword
5385                          | messages_keyword
5386                          ;
5387      messages_keyword       : 'yesexpr' '"' EXTENDED_REG_EXP '"' EOL
5388                          | 'noexpr'  '"' EXTENDED_REG_EXP '"' EOL
5389                          ;
5390      messages_tlr           : 'END' 'LC_MESSAGES' EOL
5391                          ;
5392      /* The following is the LC_MONETARY category grammar */
5393      lc_monetary            : monetary_hdr monetary_keywords      monetary_tlr
5394                          | monetary_hdr 'copy' locale_name EOL monetary_tlr
5395                          ;
5396      monetary_hdr           : 'LC_MONETARY' EOL
5397                          ;

```

```

5398     monetary_keywords    : monetary_keywords monetary_keyword
5399                             | monetary_keyword
5400                             ;
5401     monetary_keyword       : mon_keyword_string mon_string EOL
5402                             | mon_keyword_char NUMBER EOL
5403                             | mon_keyword_char '-1' EOL
5404                             | mon_keyword_grouping mon_group_list EOL
5405                             ;
5406     mon_keyword_string      : 'int_curr_symbol' | 'currency_symbol'
5407                             | 'mon_decimal_point' | 'mon_thousands_sep'
5408                             | 'positive_sign' | 'negative_sign'
5409                             ;
5410     mon_string              : '"' char_list '"'
5411                             | '""'
5412                             ;
5413     mon_keyword_char        : 'int_frac_digits' | 'frac_digits'
5414                             | 'p_cs_precedes' | 'p_sep_by_space'
5415                             | 'n_cs_precedes' | 'n_sep_by_space'
5416                             | 'p_sign_posn' | 'n_sign_posn'
5417                             | 'int_p_cs_precedes' | 'int_p_sep_by_space'
5418                             | 'int_n_cs_precedes' | 'int_n_sep_by_space'
5419                             | 'int_p_sign_posn' | 'int_n_sign_posn'
5420                             ;
5421     mon_keyword_grouping    : 'mon_grouping'
5422                             ;
5423     mon_group_list          : NUMBER
5424                             | mon_group_list ';' NUMBER
5425                             ;
5426     monetary_tlr            : 'END' 'LC_MONETARY' EOL
5427                             ;
5428     /* The following is the LC_NUMERIC category grammar */
5429     lc_numeric               : numeric_hdr numeric_keywords      numeric_tlr
5430                             | numeric_hdr 'copy' locale_name EOL numeric_tlr
5431                             ;
5432     numeric_hdr              : 'LC_NUMERIC' EOL
5433                             ;
5434     numeric_keywords         : numeric_keywords numeric_keyword
5435                             | numeric_keyword
5436                             ;
5437     numeric_keyword          : num_keyword_string num_string EOL
5438                             | num_keyword_grouping num_group_list EOL
5439                             ;
5440     num_keyword_string       : 'decimal_point'
5441                             | 'thousands_sep'
5442                             ;
5443     num_string               : '"' char_list '"'

```



```

5444         | '""'
5445         ;
5446     num_keyword_grouping: 'grouping'
5447         ;
5448     num_group_list      : NUMBER
5449         | num_group_list ';' NUMBER
5450         ;
5451     numeric_tlr         : 'END' 'LC_NUMERIC' EOL
5452         ;
5453     /* The following is the LC_TIME category grammar */
5454     lc_time              : time_hdr time_keywords          time_tlr
5455         | time_hdr 'copy' locale_name EOL time_tlr
5456         ;
5457     time_hdr             : 'LC_TIME' EOL
5458         ;
5459     time_keywords        : time_keywords time_keyword
5460         | time_keyword
5461         ;
5462     time_keyword         : time_keyword_name time_list EOL
5463         | time_keyword_fmt time_string EOL
5464         | time_keyword_opt time_list EOL
5465         ;
5466     time_keyword_name    : 'abday' | 'day' | 'abmon' | 'mon'
5467         ;
5468     time_keyword_fmt     : 'd_t_fmt' | 'd_fmt' | 't_fmt'
5469         | 'am_pm' | 't_fmt_ampm'
5470         ;
5471     time_keyword_opt     : 'era' | 'era_d_fmt' | 'era_t_fmt'
5472         | 'era_d_t_fmt' | 'alt_digits'
5473         ;
5474     time_list            : time_list ';' time_string
5475         | time_string
5476         ;
5477     time_string          : '"' char_list '"'
5478         ;
5479     time_tlr             : 'END' 'LC_TIME' EOL
5480         ;

```


5481
5482

Chapter 8 Environment Variables

8.1 Environment Variable Definition

Environment variables defined in this chapter affect the operation of multiple utilities, functions, and applications. There are other environment variables that are of interest only to specific utilities. Environment variables that apply to a single utility only are defined as part of the utility description. See the ENVIRONMENT VARIABLES section of the utility descriptions in the Shell and Utilities volume of POSIX.1-2008 for information on environment variable usage.

The value of an environment variable is a string of characters. For a C-language program, an array of strings called the environment shall be made available when a process begins. The array is pointed to by the external variable *environ*, which is defined as:

```
extern char **environ;
```

These strings have the form *name=value*; *names* shall not contain the character '='. For values to be portable across systems conforming to POSIX.1-2008, the value shall be composed of characters from the portable character set (except NUL and as indicated below). There is no meaning associated with the order of strings in the environment. If more than one string in an environment of a process has the same *name*, the consequences are undefined.

Environment variable names used by the utilities in the Shell and Utilities volume of POSIX.1-2008 consist solely of uppercase letters, digits, and the <underscore> ('_') from the characters defined in Table 6-1 (on page 125) and do not begin with a digit. Other characters may be permitted by an implementation; applications shall tolerate the presence of such names. Uppercase and lowercase letters shall retain their unique identities and shall not be folded together. The name space of environment variable names containing lowercase letters is reserved for applications. Applications can define any environment variables with names from this name space without modifying the behavior of the standard utilities.

Note: Other applications may have difficulty dealing with environment variable names that start with a digit. For this reason, use of such names is not recommended anywhere.

The *values* that the environment variables may be assigned are not restricted except that they are considered to end with a null byte and the total space used to store the environment and the arguments to the process is limited to {ARG_MAX} bytes.

Other *name=value* pairs may be placed in the environment by, for example, calling any of the *setenv()*, *unsetenv()*, or *putenv()* functions, assigning a new value to the *environ* variable, or by using *envp* arguments when creating a process; see *exec* in the System Interfaces volume of POSIX.1-2008.

If the application modifies the pointers to which *environ* points, the behavior of all interfaces described in the System Interfaces volume of POSIX.1-2008 is undefined.

It is unwise to conflict with certain variables that are frequently exported by widely used command interpreters and applications:

5519	ARFLAGS	IFS	MAILPATH	PS1
5520	CC	LANG	MAILRC	PS2
5521	CDPATH	LC_ALL	MAKEFLAGS	PS3
5522	CFLAGS	LC_COLLATE	MAKESHELL	PS4
5523	CHARSET	LC_CTYPE	MANPATH	PWD
5524	COLUMNS	LC_MESSAGES	MBOX	RANDOM
5525	DATMSK	LC_MONETARY	MORE	SECONDS
5526	DEAD	LC_NUMERIC	MSGVERB	SHELL
5527	EDITOR	LC_TIME	NLSPATH	TERM
5528	ENV	LDFLAGS	NPROC	TERMCAP
5529	EXINIT	LEX	OLDPWD	TERMINFO
5530	FC	LFLAGS	OPTARG	TMPDIR
5531	FCEDIT	LINENO	OPTERR	TZ
5532	FFLAGS	LINES	OPTIND	USER
5533	GET	LISTER	PAGER	VISUAL
5534	GFLAGS	LOGNAME	PATH	YACC
5535	HISTFILE	LPDEST	PPID	YFLAGS
5536	HISTORY	MAIL	PRINTER	
5537	HISTSIZE	MAILCHECK	PROCLANG	
5538	HOME	MAILER	PROJECTDIR	

If the variables in the following two sections are present in the environment during the execution of an application or utility, they shall be given the meaning described below. Some are placed into the environment by the implementation at the time the user logs in; all can be added or changed by the user or any ancestor of the current process. The implementation adds or changes environment variables named in POSIX.1-2008 only as specified in POSIX.1-2008. If they are defined in the application's environment, the utilities in the Shell and Utilities volume of POSIX.1-2008 and the functions in the System Interfaces volume of POSIX.1-2008 assume they have the specified meaning. Conforming applications shall not set these environment variables to have meanings other than as described. See *getenv()* (on page 1017) and XCU [Section 2.12](#) (on page 2353) for methods of accessing these variables.

8.2 Internationalization Variables

This section describes environment variables that are relevant to the operation of internationalized interfaces described in POSIX.1-2008.

Users may use the following environment variables to announce specific localization requirements to applications. Applications can retrieve this information using the *setlocale()* function to initialize the correct behavior of the internationalized interfaces. The descriptions of the internationalization environment variables describe the resulting behavior only when the application locale is initialized in this way. The use of the internationalization variables by utilities described in the Shell and Utilities volume of POSIX.1-2008 is described in the ENVIRONMENT VARIABLES section for those utilities in addition to the global effects described in this section.

LANG This variable shall determine the locale category for native language, local customs, and coded character set in the absence of the *LC_ALL* and other *LC_** (*LC_COLLATE*, *LC_CTYPE*, *LC_MESSAGES*, *LC_MONETARY*, *LC_NUMERIC*, *LC_TIME*) environment variables. This can be used by applications to determine the language to use for error messages and instructions, collating sequences, date formats, and so on.

5566	<i>LC_ALL</i>	This variable shall determine the values for all locale categories. The value of the <i>LC_ALL</i> environment variable has precedence over any of the other environment variables starting with <i>LC_</i> (<i>LC_COLLATE</i> , <i>LC_CTYPE</i> , <i>LC_MESSAGES</i> , <i>LC_MONETARY</i> , <i>LC_NUMERIC</i> , <i>LC_TIME</i>) and the <i>LANG</i> environment variable.
5571	<i>LC_COLLATE</i>	This variable shall determine the locale category for character collation. It determines collation information for regular expressions and sorting, including equivalence classes and multi-character collating elements, in various utilities and the <i>strcoll()</i> and <i>strxfrm()</i> functions. Additional semantics of this variable, if any, are implementation-defined.
5576	<i>LC_CTYPE</i>	This variable shall determine the locale category for character handling functions, such as <i>tolower()</i> , <i>toupper()</i> , and <i>isalpha()</i> . This environment variable determines the interpretation of sequences of bytes of text data as characters (for example, single as opposed to multi-byte characters), the classification of characters (for example, alpha, digit, graph), and the behavior of character classes. Additional semantics of this variable, if any, are implementation-defined.
5583	<i>LC_MESSAGES</i>	This variable shall determine the locale category for processing affirmative and negative responses and the language and cultural conventions in which messages should be written. It also affects the behavior of the <i>catopen()</i> function in determining the message catalog. Additional semantics of this variable, if any, are implementation-defined. The language and cultural conventions of diagnostic and informative messages whose format is unspecified by POSIX.1-2008 should be affected by the setting of <i>LC_MESSAGES</i> .
5591	<i>LC_MONETARY</i>	This variable shall determine the locale category for monetary-related numeric formatting information. Additional semantics of this variable, if any, are implementation-defined.
5594	<i>LC_NUMERIC</i>	This variable shall determine the locale category for numeric formatting (for example, thousands separator and radix character) information in various utilities as well as the formatted I/O operations in <i>printf()</i> and <i>scanf()</i> and the string conversion functions in <i>strtod()</i> . Additional semantics of this variable, if any, are implementation-defined.
5599	<i>LC_TIME</i>	This variable shall determine the locale category for date and time formatting information. It affects the behavior of the time functions in <i>strptime()</i> . Additional semantics of this variable, if any, are implementation-defined.
5602	<i>NLSPATH</i>	This variable shall contain a sequence of templates that the <i>catopen()</i> function uses when attempting to locate message catalogs. Each template consists of an optional prefix, one or more conversion specifications, a pathname, and an optional suffix.
5606		For example:
5607		<code>NLSPATH="/system/nlslib/%N.cat"</code>
5608		defines that <i>catopen()</i> should look for all message catalogs in the directory /system/nlslib , where the catalog name should be constructed from the <i>name</i> parameter passed to <i>catopen()</i> (<i>%N</i>), with the suffix .cat .
5611		Conversion specifications consist of a ' <i>%</i> ' symbol, followed by a single-letter keyword. The following keywords are currently defined:
5612		

5613 %N The value of the *name* parameter passed to *catopen()*.

5614 %L The value of the *LC_MESSAGES* category.

5615 %l The *language* element from the *LC_MESSAGES* category.

5616 %t The *territory* element from the *LC_MESSAGES* category.

5617 %c The *codeset* element from the *LC_MESSAGES* category.

5618 %% A single '%' character.

5619 An empty string is substituted if the specified value is not currently defined.

5620 The separators <underscore> ('_') and <period> ('.') are not included in

5621 the %t and %c conversion specifications.

5622 Templates defined in *NLSPATH* are separated by <colon> characters (':'). A

5623 leading or two adjacent <colon> characters (": : ") is equivalent to specifying

5624 %N. For example:

5625 NLSPATH=":%N.cat:/nlslib/%L/%N.cat "

5626 indicates to *catopen()* that it should look for the requested message catalog in

5627 *name*, *name.cat*, and */nlslib/category/name.cat*, where *category* is the value of the

5628 *LC_MESSAGES* category of the current locale.

5629 Users should not set the *NLSPATH* variable unless they have a specific reason

5630 to override the default system path. Setting *NLSPATH* to override the default

5631 system path produces undefined results in the standard utilities and in

5632 applications with appropriate privileges.

5633 The environment variables *LANG*, *LC_ALL*, *LC_COLLATE*, *LC_CTYPE*, *LC_MESSAGES*,

5634 *LC_MONETARY*, *LC_NUMERIC*, *LC_TIME*, and *NLSPATH* provide for the support of

5635 internationalized applications. The standard utilities shall make use of these environment

5636 variables as described in this section and the individual ENVIRONMENT VARIABLES sections

5637 for the utilities. If these variables specify locale categories that are not based upon the same

5638 underlying codeset, the results are unspecified.

5639 The values of locale categories shall be determined by a precedence order; the first condition met

5640 below determines the value:

- 5641 1. If the *LC_ALL* environment variable is defined and is not null, the value of *LC_ALL* shall
- 5642 be used.
- 5643 2. If the *LC_** environment variable (*LC_COLLATE*, *LC_CTYPE*, *LC_MESSAGES*,
- 5644 *LC_MONETARY*, *LC_NUMERIC*, *LC_TIME*) is defined and is not null, the value of the
- 5645 environment variable shall be used to initialize the category that corresponds to the
- 5646 environment variable.
- 5647 3. If the *LANG* environment variable is defined and is not null, the value of the *LANG*
- 5648 environment variable shall be used.
- 5649 4. If the *LANG* environment variable is not set or is set to the empty string, the
- 5650 implementation-defined default locale shall be used.

5651 If the locale value is "C" or "POSIX", the POSIX locale shall be used and the standard utilities

5652 behave in accordance with the rules in [Section 7.2](#) (on page 136) for the associated category.

5653 If the locale value begins with a <slash>, it shall be interpreted as the pathname of a file that was

5654 created in the output format used by the *localedef* utility; see OUTPUT FILES under *localedef*.

5655 Referencing such a pathname shall result in that locale being used for the indicated category.

5656 XSI If the locale value has the form:

5657 `language[_territory][.codeset]`

5658 it refers to an implementation-provided locale, where settings of language, territory, and codeset
5659 are implementation-defined.

5660 `LC_COLLATE`, `LC_CTYPE`, `LC_MESSAGES`, `LC_MONETARY`, `LC_NUMERIC`, and `LC_TIME` are
5661 defined to accept an additional field `@modifier`, which allows the user to select a specific instance
5662 of localization data within a single category (for example, for selecting the dictionary as opposed
5663 to the character ordering of data). The syntax for these environment variables is thus defined as:

5664 `[language[_territory][.codeset][@modifier]]`

5665 For example, if a user wanted to interact with the system in French, but required to sort German
5666 text files, `LANG` and `LC_COLLATE` could be defined as:

5667 `LANG=Fr_FR`
5668 `LC_COLLATE=De_DE`

5669 This could be extended to select dictionary collation (say) by use of the `@modifier` field; for
5670 example:

5671 `LC_COLLATE=De_DE@dict`

5672 An implementation may support other formats.

5673 If the locale value is not recognized by the implementation, the behavior is unspecified.

5674 These environment variables are used by the `newlocale()` and `setlocale()` functions, and by the
5675 standard utilities.

5676 Additional criteria for determining a valid locale name are implementation-defined.

5677 8.3 Other Environment Variables

5678 **COLUMNS** This variable shall represent a decimal integer >0 used to indicate the user's
5679 preferred width in column positions for the terminal screen or window; see
5680 [Section 3.103](#) (on page 50). If this variable is unset or null, the implementation
5681 determines the number of columns, appropriate for the terminal or window,
5682 in an unspecified manner. When `COLUMNS` is set, any terminal-width
5683 information implied by `TERM` is overridden. Users and conforming
5684 applications should not set `COLUMNS` unless they wish to override the
5685 system selection and produce output unrelated to the terminal characteristics.

5686 Users should not need to set this variable in the environment unless there is a
5687 specific reason to override the implementation's default behavior, such as to
5688 display data in an area arbitrarily smaller than the terminal or window.

5689 XSI **DATEMSK** Indicates the pathname of the template file used by `getdate()`.

5690 **HOME** The system shall initialize this variable at the time of login to be a pathname of
5691 the user's home directory. See [<pwd.h>](#).

5692 **LINES** This variable shall represent a decimal integer >0 used to indicate the user's
5693 preferred number of lines on a page or the vertical screen or window size in
5694 lines. A line in this case is a vertical measure large enough to hold the tallest
5695 character in the character set being displayed. If this variable is unset or null,

5696		the implementation determines the number of lines, appropriate for the
5697		terminal or window (size, terminal baud rate, and so on), in an unspecified
5698		manner. When <i>LINES</i> is set, any terminal-height information implied by
5699		<i>TERM</i> is overridden. Users and conforming applications should not set <i>LINES</i>
5700		unless they wish to override the system selection and produce output
5701		unrelated to the terminal characteristics.
5702		Users should not need to set this variable in the environment unless there is a
5703		specific reason to override the implementation's default behavior, such as to
5704		display data in an area arbitrarily smaller than the terminal or window.
5705	<i>LOGNAME</i>	The system shall initialize this variable at the time of login to be the user's
5706		login name. See <pwd.h> . For a value of <i>LOGNAME</i> to be portable across
5707		implementations of POSIX.1-2008, the value should be composed of characters
5708		from the portable filename character set.
5709	XSI <i>MSGVERB</i>	Describes which message components shall be used in writing messages by
5710		<i>fntmsg()</i> .
5711	<i>PATH</i>	This variable shall represent the sequence of path prefixes that certain
5712		functions and utilities apply in searching for an executable file known only by
5713		a filename. The prefixes shall be separated by a <colon> (' : '). When a non-
5714		zero-length prefix is applied to this filename, a <slash> shall be inserted
5715		between the prefix and the filename if the prefix did not end in <slash>. A
5716		zero-length prefix is a legacy feature that indicates the current working
5717		directory. It appears as two adjacent <colon> characters (" : "), as an initial
5718		<colon> preceding the rest of the list, or as a trailing <colon> following the
5719		rest of the list. A strictly conforming application shall use an actual pathname
5720		(such as .) to represent the current working directory in <i>PATH</i> . The list shall
5721		be searched from beginning to end, applying the filename to each prefix, until
5722		an executable file with the specified name and appropriate execution
5723		permissions is found. If the pathname being sought contains a <slash>, the
5724		search through the path prefixes shall not be performed. If the pathname
5725		begins with a <slash>, the specified path is resolved (see Section 4.12 , on page
5726		111). If <i>PATH</i> is unset or is set to null, the path search is implementation-
5727		defined.
5728		Since <colon> is a separator in this context, directory names that might be +
5729		used in <i>PATH</i> should not include a <colon> character.
5730	<i>PWD</i>	This variable shall represent an absolute pathname of the current working
5731		directory. It shall not contain any components that are dot or dot-dot. The
5732		value is set by the <i>cd</i> utility, and by the <i>sh</i> utility during initialization.
5733	<i>SHELL</i>	This variable shall represent a pathname of the user's preferred command
5734		language interpreter. If this interpreter does not conform to the Shell
5735		Command Language in XCU Chapter 2 (on page 2319), utilities may behave
5736		differently from those described in POSIX.1-2008.
5737	<i>TMPDIR</i>	This variable shall represent a pathname of a directory made available for
5738		programs that need a place to create temporary files.
5739	<i>TERM</i>	This variable shall represent the terminal type for which output is to be
5740		prepared. This information is used by utilities and application programs
5741		wishing to exploit special capabilities specific to a terminal. The format and
5742		allowable values of this environment variable are unspecified.

5743	<i>TZ</i>	This variable shall represent timezone information. The contents of the
5744		environment variable named <i>TZ</i> shall be used by the <i>ctime()</i> , <i>ctime_r()</i> ,
5745		<i>localtime()</i> , <i>localtime_r()</i> , <i>strptime()</i> , <i>mktime()</i> , functions, and by various
5746		utilities, to override the default timezone. The value of <i>TZ</i> has one of the two
5747		forms (spaces inserted for clarity):
5748		<i>:characters</i>
5749		or:
5750		<i>std offset dst offset, rule</i>
5751		If <i>TZ</i> is of the first format (that is, if the first character is a <colon>), the
5752		characters following the <colon> are handled in an implementation-defined
5753		manner.
5754		The expanded format (for all <i>TZ</i> s whose value does not have a <colon> as the
5755		first character) is as follows:
5756		<i>stdoffset[dst[offset][,start[/time],end[/time]]]</i>
5757		Where:
5758	<i>std</i> and <i>dst</i>	Indicate no less than three, nor more than {TZNAME_MAX},
5759		bytes that are the designation for the standard (<i>std</i>) or the
5760		alternative (<i>dst</i> —such as Daylight Savings Time) timezone. Only
5761		<i>std</i> is required; if <i>dst</i> is missing, then the alternative time does
5762		not apply in this locale.
5763		Each of these fields may occur in either of two formats quoted or
5764		unquoted:
5765		— In the quoted form, the first character shall be the <less-
5766		than-sign> ('<') character and the last character shall be
5767		the <greater-than-sign> ('>') character. All characters
5768		between these quoting characters shall be alphanumeric
5769		characters from the portable character set in the current
5770		locale, the <plus-sign> ('+') character, or the minus-sign
5771		('−') character. The <i>std</i> and <i>dst</i> fields in this case shall not
5772		include the quoting characters.
5773		— In the unquoted form, all characters in these fields shall be
5774		alphabetic characters from the portable character set in the
5775		current locale.
5776		The interpretation of these fields is unspecified if either field is
5777		less than three bytes (except for the case when <i>dst</i> is missing),
5778		more than {TZNAME_MAX} bytes, or if they contain characters
5779		other than those specified.
5780	<i>offset</i>	Indicates the value added to the local time to arrive at
5781		Coordinated Universal Time. The <i>offset</i> has the form:
5782		<i>hh[:mm[:ss]]</i>
5783		The minutes (<i>mm</i>) and seconds (<i>ss</i>) are optional. The hour (<i>hh</i>)
5784		shall be required and may be a single digit. The <i>offset</i> following
5785		<i>std</i> shall be required. If no <i>offset</i> follows <i>dst</i> , the alternative time
5786		is assumed to be one hour ahead of standard time. One or more
5787		digits may be used; the value is always interpreted as a decimal

5788		number. The hour shall be between zero and 24, and the minutes
5789		(and seconds)—if present—between zero and 59. The result of
5790		using values outside of this range is unspecified. If preceded by
5791		a '−', the timezone shall be east of the Prime Meridian;
5792		otherwise, it shall be west (which may be indicated by an
5793		optional preceding '+').
5794	<i>rule</i>	Indicates when to change to and back from the alternative time.
5795		The <i>rule</i> has the form:
5796		<code>date[/time], date[/time]</code>
5797		where the first <i>date</i> describes when the change from standard to
5798		alternative time occurs and the second <i>date</i> describes when the
5799		change back happens. Each <i>time</i> field describes when, in current
5800		local time, the change to the other time is made.
5801		The format of <i>date</i> is one of the following:
5802	<i>Jn</i>	The Julian day <i>n</i> ($1 \leq n \leq 365$). Leap days shall not be
5803		counted. That is, in all years—including leap years—
5804		February 28 is day 59 and March 1 is day 60. It is
5805		impossible to refer explicitly to the occasional February
5806		29.
5807	<i>n</i>	The zero-based Julian day ($0 \leq n \leq 365$). Leap days shall
5808		be counted, and it is possible to refer to February 29.
5809	<i>Mm.n.d</i>	The <i>d</i> 'th day ($0 \leq d \leq 6$) of week <i>n</i> of month <i>m</i> of the
5810		year ($1 \leq n \leq 5, 1 \leq m \leq 12$, where week 5 means “the last
5811		<i>d</i> day in month <i>m</i> ” which may occur in either the fourth
5812		or the fifth week). Week 1 is the first week in which the
5813		<i>d</i> 'th day occurs. Day zero is Sunday.
5814		The <i>time</i> has the same format as <i>offset</i> except that no leading sign
5815		('−' or '+') is allowed. The default, if <i>time</i> is not given, shall be
5816		02:00:00.

5817

5818

Regular Expressions

5819 Regular Expressions (REs) provide a mechanism to select specific strings from a set of character
5820 strings.

5821 Regular expressions are a context-independent syntax that can represent a wide variety of
5822 character sets and character set orderings, where these character sets are interpreted according
5823 to the current locale. While many regular expressions can be interpreted differently depending
5824 on the current locale, many features, such as character class expressions, provide for contextual
5825 invariance across locales.

5826 The Basic Regular Expression (BRE) notation and construction rules in [Section 9.3](#) (on page 183)
5827 shall apply to most utilities supporting regular expressions. Some utilities, instead, support the
5828 Extended Regular Expressions (ERE) described in [Section 9.4](#) (on page 188); any exceptions for
5829 both cases are noted in the descriptions of the specific utilities using regular expressions. Both
5830 BREs and EREs are supported by the Regular Expression Matching interface in the System
5831 Interfaces volume of POSIX.1-2008 under *regcomp()*, *regex()*, and related functions.

5832 9.1 Regular Expression Definitions

5833 For the purposes of this section, the following definitions shall apply:

5834 entire regular expression

5835 The concatenated set of one or more BREs or EREs that make up the pattern specified for
5836 string selection.

5837 matched

5838 A sequence of zero or more characters shall be said to be matched by a BRE or ERE when
5839 the characters in the sequence correspond to a sequence of characters defined by the
5840 pattern.

5841 Matching shall be based on the bit pattern used for encoding the character, not on the
5842 graphic representation of the character. This means that if a character set contains two or
5843 more encodings for a graphic symbol, or if the strings searched contain text encoded in
5844 more than one codeset, no attempt is made to search for any other representation of the
5845 encoded symbol. If that is required, the user can specify equivalence classes containing all
5846 variations of the desired graphic symbol.

5847 The search for a matching sequence starts at the beginning of a string and stops when the
5848 first sequence matching the expression is found, where “first” is defined to mean “begins
5849 earliest in the string”. If the pattern permits a variable number of matching characters and
5850 thus there is more than one such sequence starting at that point, the longest such sequence
5851 is matched. For example, the BRE “bb*” matches the second to fourth characters of the
5852 string “abbbc”, and the ERE “(wee|week)(knights|night)” matches all ten
5853 characters of the string “weeknights”.

5854 Consistent with the whole match being the longest of the leftmost matches, each subpattern,
5855 from left to right, shall match the longest possible string. For this purpose, a null string shall
5856 be considered to be longer than no match at all. For example, matching the BRE

"\ (. * \) . *" against "abcdef", the subexpression " (\1) " is "abcdef", and matching the BRE "\ (a * \) *" against "bc", the subexpression " (\1) " is the null string.

When a multi-character collating element in a bracket expression (see [Section 9.3.5](#), on page 184) is involved, the longest sequence shall be measured in characters consumed from the string to be matched; that is, the collating element counts not as one element, but as the number of characters it matches.

BRE (ERE) matching a single character

A BRE or ERE that shall match either a single character or a single collating element.

Only a BRE or ERE of this type that includes a bracket expression (see [Section 9.3.5](#), on page 184) can match a collating element.

BRE (ERE) matching multiple characters

A BRE or ERE that shall match a concatenation of single characters or collating elements.

Such a BRE or ERE is made up from a BRE (ERE) matching a single character and BRE (ERE) special characters.

invalid

This section uses the term “invalid” for certain constructs or conditions. Invalid REs shall cause the utility or function using the RE to generate an error condition. When invalid is not used, violations of the specified syntax or semantics for REs produce undefined results: this may entail an error, enabling an extended syntax for that RE, or using the construct in error as literal characters to be matched. For example, the BRE construct "\ {1, 2, 3\}" does not comply with the grammar. A conforming application cannot rely on it producing an error nor matching the literal characters "\ {1, 2, 3\}".

9.2 Regular Expression General Requirements

The requirements in this section shall apply to both basic and extended regular expressions.

The use of regular expressions is generally associated with text processing. REs (BREs and EREs) operate on text strings; that is, zero or more characters followed by an end-of-string delimiter (typically NUL). Some utilities employing regular expressions limit the processing to lines; that is, zero or more characters followed by a <newline>. In the regular expression processing described in POSIX.1-2008, the <newline> is regarded as an ordinary character and both a <period> and a non-matching list can match one. The Shell and Utilities volume of POSIX.1-2008 specifies within the individual descriptions of those standard utilities employing regular expressions whether they permit matching of <newline> characters; if not stated otherwise, the use of literal <newline> characters or any escape sequence equivalent produces undefined results. Those utilities (like *grep*) that do not allow <newline> characters to match are responsible for eliminating any <newline> from strings before matching against the RE. The *regcomp()* function in the System Interfaces volume of POSIX.1-2008, however, can provide support for such processing without violating the rules of this section.

The interfaces specified in POSIX.1-2008 do not permit the inclusion of a NUL character in an RE or in the string to be matched. If during the operation of a standard utility a NUL is included in the text designated to be matched, that NUL may designate the end of the text string for the purposes of matching.

When a standard utility or function that uses regular expressions specifies that pattern matching shall be performed without regard to the case (uppercase or lowercase) of either data or patterns, then when each character in the string is matched against the pattern, not only the

character, but also its case counterpart (if any), shall be matched. This definition of case-insensitive processing is intended to allow matching of multi-character collating elements as well as characters, as each character in the string is matched using both its cases. For example, in a locale where "Ch" is a multi-character collating element and where a matching list expression matches such elements, the RE "[[.Ch.]]" when matched against the string "char" is in reality matched against "ch", "Ch", "cH", and "CH".

The implementation shall support any regular expression that does not exceed 256 bytes in length.

9.3 Basic Regular Expressions

9.3.1 BREs Matching a Single Character or Collating Element

A BRE ordinary character, a special character preceded by a <backslash>, or a <period> shall match a single character. A bracket expression shall match a single character or a single collating element.

9.3.2 BRE Ordinary Characters

An ordinary character is a BRE that matches itself: any character in the supported character set, except for the BRE special characters listed in [Section 9.3.3](#).

The interpretation of an ordinary character preceded by a <backslash> ('\\') is undefined, except for:

- The characters ') ' , ' (' , ' { ' , and ' } '
- The digits 1 to 9 inclusive (see [Section 9.3.6](#), on page 186)
- A character inside a bracket expression

9.3.3 BRE Special Characters

A BRE special character has special properties in certain contexts. Outside those contexts, or when preceded by a <backslash>, such a character is a BRE that matches the special character itself. The BRE special characters and the contexts in which they have their special meaning are as follows:

. [\ The <period>, <left-square-bracket>, and <backslash> shall be special except when used in a bracket expression (see [Section 9.3.5](#), on page 184). An expression containing a ' [' that is not preceded by a <backslash> and is not part of a bracket expression produces undefined results.

* The <asterisk> shall be special except when used:

- In a bracket expression
- As the first character of an entire BRE (after an initial ' ^ ' , if any)

- 5934 — As the first character of a subexpression (after an initial '[^]', if any); see [Section](#)
 5935 [9.3.6](#) (on page 186)
- 5936 ^ The <circumflex> shall be special when used as:
 5937 — An anchor (see [Section 9.3.8](#), on page 187)
 5938 — The first character of a bracket expression (see [Section 9.3.5](#))
- 5939 \$ The <dollar-sign> shall be special when used as an anchor.

5940 9.3.4 Periods in BREs

5941 A <period> ('.'), when used outside a bracket expression, is a BRE that shall match any
 5942 character in the supported character set except NUL.

5943 9.3.5 RE Bracket Expression

5944 A bracket expression (an expression enclosed in square brackets, "[]") is an RE that shall
 5945 match a single collating element contained in the non-empty set of collating elements
 5946 represented by the bracket expression.

5947 The following rules and definitions apply to bracket expressions:

- 5948 1. A bracket expression is either a matching list expression or a non-matching list
 5949 expression. It consists of one or more expressions: collating elements, collating symbols,
 5950 equivalence classes, character classes, or range expressions. The <right-square-bracket>
 5951 ('] ') shall lose its special meaning and represent itself in a bracket expression if it occurs
 5952 first in the list (after an initial <circumflex> ('[^]'), if any). Otherwise, it shall terminate
 5953 the bracket expression, unless it appears in a collating symbol (such as "[.] .] ") or is the
 5954 ending <right-square-bracket> for a collating symbol, equivalence class, or character
 5955 class. The special characters '.', '*', '[', and '\\ ' (<period>, <asterisk>, <left-square-
 5956 bracket>, and <backslash>, respectively) shall lose their special meaning within a bracket
 5957 expression.
 5958 The character sequences "[. ", "[= ", and "[: " (<left-square-bracket> followed by a
 5959 <period>, <equals-sign>, or <colon>) shall be special inside a bracket expression and are
 5960 used to delimit collating symbols, equivalence class expressions, and character class
 5961 expressions. These symbols shall be followed by a valid expression and the matching
 5962 terminating sequence ".] ", "=] ", or ":] ", as described in the following items.
- 5963 2. A matching list expression specifies a list that shall match any single-character collating
 5964 element in any of the expressions represented in the list. The first character in the list shall
 5965 not be the <circumflex>; for example, "[abc]" is an RE that matches any of the
 5966 characters 'a', 'b', or 'c'. It is unspecified whether a matching list expression matches
 5967 a multi-character collating element that is matched by one of the expressions.
- 5968 3. A non-matching list expression begins with a <circumflex> ('[^]'), and specifies a list that
 5969 shall match any single-character collating element except for the expressions represented
 5970 in the list after the leading <circumflex>. For example, "[^abc]" is an RE that matches
 5971 any character except the characters 'a', 'b', or 'c'. It is unspecified whether a non-
 5972 matching list expression matches a multi-character collating element that is not matched
 5973 by any of the expressions. The <circumflex> shall have this special meaning only when it
 5974 occurs first in the list, immediately following the <left-square-bracket>.

4. A collating symbol is a collating element enclosed within bracket-period ("[" and ".]") delimiters. Collating elements are defined as described in [Section 7.3.2.4](#) (on page 149). Conforming applications shall represent multi-character collating elements as collating symbols when it is necessary to distinguish them from a list of the individual characters that make up the multi-character collating element. For example, if the string "ch" is a collating element defined using the line:

```
collating-element <ch-digraph> from "<c><h>"
```

in the locale definition, the expression "[.ch.]" shall be treated as an RE containing the collating symbol 'ch', while "[ch]" shall be treated as an RE matching 'c' or 'h'. Collating symbols are recognized only inside bracket expressions. If the string is not a collating element in the current locale, the expression is invalid.

5. An equivalence class expression shall represent the set of collating elements belonging to an equivalence class, as described in [Section 7.3.2.4](#) (on page 149). Only primary equivalence classes shall be recognized. The class shall be expressed by enclosing any one of the collating elements in the equivalence class within bracket-equal ("[" and "=") delimiters. For example, if 'a', 'à', and '^' belong to the same equivalence class, then "[[=a=]b]", "[[=à=]b]", and "[[=^=]b]" are each equivalent to "[aà^b]". If the collating element does not belong to an equivalence class, the equivalence class expression shall be treated as a collating symbol.
6. A character class expression shall represent the union of two sets:

- a. The set of single-character collating elements whose characters belong to the character class, as defined in the *LC_CTYPE* category in the current locale.
- b. An unspecified set of multi-character collating elements.

All character classes specified in the current locale shall be recognized. A character class expression is expressed as a character class name enclosed within bracket-colon ("[: " and ":]") delimiters.

The following character class expressions shall be supported in all locales:

```
[ :alnum:]      [ :cntrl:]      [ :lower:]      [ :space:]
[ :alpha:]      [ :digit:]      [ :print:]      [ :upper:]
[ :blank:]      [ :graph:]      [ :punct:]      [ :xdigit:]
```

In addition, character class expressions of the form:

```
[ :name:]
```

are recognized in those locales where the *name* keyword has been given a **charclass** definition in the *LC_CTYPE* category.

7. In the POSIX locale, a range expression represents the set of collating elements that fall between two elements in the collation sequence, inclusive. In other locales, a range expression has unspecified behavior: strictly conforming applications shall not rely on whether the range expression is valid, or on the set of collating elements matched. A range expression shall be expressed as the starting point and the ending point separated by a <hyphen> ('-').

In the following, all examples assume the POSIX locale.

The starting range point and the ending range point shall be a collating element or collating symbol. An equivalence class expression used as a starting or ending point of a range expression produces unspecified results. An equivalence class can be used portably within a bracket expression, but only outside the range. If the represented set of collating

elements is empty, it is unspecified whether the expression matches nothing, or is treated as invalid.

The interpretation of range expressions where the ending range point is also the starting range point of a subsequent range expression (for example, "[a-m-o]") is undefined.

The <hyphen> character shall be treated as itself if it occurs first (after an initial '^', if any) or last in the list, or as an ending range point in a range expression. As examples, the expressions "[a-c]" and "[ac-]" are equivalent and match any of the characters 'a', 'c', or '-'; "[^a-c]" and "[^ac-]" are equivalent and match any characters except 'a', 'c', or '-'; the expression "[%--]" matches any of the characters between '%' and '-' inclusive; the expression "[--@]" matches any of the characters between '-' and '@' inclusive; and the expression "[a--@]" is either invalid or equivalent to '@', because the letter 'a' follows the symbol '-' in the POSIX locale. To use a <hyphen> as the starting range point, it shall either come first in the bracket expression or be specified as a collating symbol; for example, "[[][-.-]-0]", which matches either a <right-square-bracket> or any character or collating element that collates between <hyphen> and 0, inclusive.

If a bracket expression specifies both '-' and ']', the ']' shall be placed first (after the '^', if any) and the '-' last within the bracket expression.

8. If a bracket expression contains at least three list elements, where the first and last list elements are the same single-character element of <period>, <equals-sign>, or <colon>, then it is unspecified whether the bracket expression will be treated as a collating symbol, equivalence class, or character class, respectively; treated as a matching list expression; or rejected as an error.

9.3.6 BREs Matching Multiple Characters

The following rules can be used to construct BREs matching multiple characters from BREs matching a single character:

1. The concatenation of BREs shall match the concatenation of the strings matched by each component of the BRE.
2. A subexpression can be defined within a BRE by enclosing it between the character pairs "\(" and "\)". Such a subexpression shall match whatever it would have matched without the "\(" and "\)", except that anchoring within subexpressions is optional behavior; see [Section 9.3.8](#) (on page 187). Subexpressions can be arbitrarily nested.
3. The back-reference expression '\n' shall match the same (possibly empty) string of characters as was matched by a subexpression enclosed between "\(" and "\)" preceding the '\n'. The character 'n' shall be a digit from 1 through 9, specifying the *n*th subexpression (the one that begins with the *n*th "\(" from the beginning of the pattern and ends with the corresponding paired "\)"). The expression is invalid if less than *n* subexpressions precede the '\n'. The string matched by a contained subexpression shall be within the string matched by the containing subexpression. If the containing subexpression does not match, or if there is no match for the contained subexpression within the string matched by the containing subexpression, then back-reference expressions corresponding to the contained subexpression shall not match. When a subexpression matches more than one string, a back-reference expression corresponding to the subexpression shall refer to the last matched string. For example, the expression "\^(.*)\1\$" matches lines consisting of two adjacent appearances of the same string, and the expression "\(a\)*\1" fails to match 'a', the expression "\(a\)(b\)*\2" fails to match 'abab', and the expression "\^(ab\)*\1\$" matches lines consisting of one or more appearances of 'ab' followed by 'ab'.

matches 'ababbabb', but fails to match 'ababbab'.

4. When a BRE matching a single character, a subexpression, or a back-reference is followed by the special character <asterisk> ('*'), together with that <asterisk> it shall match what zero or more consecutive occurrences of the BRE would match. For example, "[ab]*" and "[ab][ab]" are equivalent when matching the string "ab".
5. When a BRE matching a single character, a subexpression, or a back-reference is followed by an interval expression of the format "{m}", "{m,}", or "{m,n}", together with that interval expression it shall match what repeated consecutive occurrences of the BRE would match. The values of *m* and *n* are decimal integers in the range $0 \leq m \leq n \leq \{RE\_DUP\_MAX\}$, where *m* specifies the exact or minimum number of occurrences and *n* specifies the maximum number of occurrences. The expression "{m}" shall match exactly *m* occurrences of the preceding BRE, "{m,}" shall match at least *m* occurrences, and "{m,n}" shall match any number of occurrences between *m* and *n*, inclusive.

For example, in the string "abababcccccd" the BRE "c{3}" is matched by characters seven to nine, the BRE "\ (ab\)\{4,}" is not matched at all, and the BRE "c{1,3}d" is matched by characters ten to thirteen.

The behavior of multiple adjacent duplication symbols ('*' and intervals) produces undefined results.

A subexpression repeated by an <asterisk> ('*') or an interval expression shall not match a null expression unless this is the only match for the repetition or it is necessary to satisfy the exact or minimum number of occurrences for the interval expression.

9.3.7 BRE Precedence

The order of precedence shall be as shown in the following table:

BRE Precedence (from high to low)	
Collation-related bracket symbols	[==] [::] [..]
Escaped characters	\<special character>
Bracket expression	[]
Subexpressions/back-references	\ (\) \n
Single-character-BRE duplication	* \{m,n\}
Concatenation	
Anchoring	^ \$

9.3.8 BRE Expression Anchoring

A BRE can be limited to matching strings that begin or end a line; this is called "anchoring". The <circumflex> and <dollar-sign> special characters shall be considered BRE anchors in the following contexts:

1. A <circumflex> ('^') shall be an anchor when used as the first character of an entire BRE. The implementation may treat the <circumflex> as an anchor when used as the first character of a subexpression. The <circumflex> shall anchor the expression (or optionally subexpression) to the beginning of a string; only sequences starting at the first character of a string shall be matched by the BRE. For example, the BRE "^ab" matches "ab" in the string "abcdef", but fails to match in the string "cdefab". The BRE "\ (^ab\)" may match the former string. A portable BRE shall escape a leading <circumflex> in a

subexpression to match a literal circumflex.

2. A <dollar-sign> ('\$') shall be an anchor when used as the last character of an entire BRE. The implementation may treat a <dollar-sign> as an anchor when used as the last character of a subexpression. The <dollar-sign> shall anchor the expression (or optionally subexpression) to the end of the string being matched; the <dollar-sign> can be said to match the end-of-string following the last character.
3. A BRE anchored by both '^' and '\$' shall match only an entire string. For example, the BRE "^abcdef\$" matches strings consisting only of "abcdef".

9.4 Extended Regular Expressions

The extended regular expression (ERE) notation and construction rules shall apply to utilities defined as using extended regular expressions; any exceptions to the following rules are noted in the descriptions of the specific utilities using EREs.

9.4.1 EREs Matching a Single Character or Collating Element

An ERE ordinary character, a special character preceded by a <backslash> or a <period> shall match a single character. A bracket expression shall match a single character or a single collating element. An ERE matching a single character enclosed in parentheses shall match the same as the ERE without parentheses would have matched.

9.4.2 ERE Ordinary Characters

An ordinary character is an ERE that matches itself. An ordinary character is any character in the supported character set, except for the ERE special characters listed in [Section 9.4.3](#). The interpretation of an ordinary character preceded by a <backslash> ('\\ ') is undefined.

9.4.3 ERE Special Characters

An ERE special character has special properties in certain contexts. Outside those contexts, or when preceded by a <backslash>, such a character shall be an ERE that matches the special character itself. The extended regular expression special characters and the contexts in which they shall have their special meaning are as follows:

- . [\ (The <period>, <left-square-bracket>, <backslash>, and <left-parenthesis> shall be special except when used in a bracket expression (see [Section 9.3.5](#), on page 184). Outside a bracket expression, a <left-parenthesis> immediately followed by a <right-parenthesis> produces undefined results.
-) The <right-parenthesis> shall be special when matched with a preceding <left-parenthesis>, both outside a bracket expression.
- * + ? { The <asterisk>, <plus-sign>, <question-mark>, and <left-brace> shall be special except when used in a bracket expression (see [Section 9.3.5](#), on page 184). Any of the following uses produce undefined results:
 - If these characters appear first in an ERE, or immediately following a <vertical-line>, <circumflex>, or <left-parenthesis>

- 6147 — If a <left-brace> is not part of a valid interval expression (see [Section 9.4.6](#))
- 6148 | The <vertical-line> is special except when used in a bracket expression (see [Section](#)
- 6149 9.3.5, on page 184). A <vertical-line> appearing first or last in an ERE, or immediately
- 6150 following a <vertical-line> or a <left-parenthesis>, or immediately preceding a <right-
- 6151 parenthesis>, produces undefined results.
- 6152 ^ The <circumflex> shall be special when used as:
- 6153 — An anchor (see [Section 9.4.9](#), on page 191)
- 6154 — The first character of a bracket expression (see [Section 9.3.5](#), on page 184)
- 6155 \$ The <dollar-sign> shall be special when used as an anchor.

6156 9.4.4 Periods in EREs

6157 A <period> (' . '), when used outside a bracket expression, is an ERE that shall match any
6158 character in the supported character set except NUL.

6159 9.4.5 ERE Bracket Expression

6160 The rules for ERE Bracket Expressions are the same as for Basic Regular Expressions; see [Section](#)
6161 9.3.5 (on page 184).

6162 9.4.6 EREs Matching Multiple Characters

6163 The following rules shall be used to construct EREs matching multiple characters from EREs
6164 matching a single character:

- 6165 1. A concatenation of EREs shall match the concatenation of the character sequences
6166 matched by each component of the ERE. A concatenation of EREs enclosed in parentheses
6167 shall match whatever the concatenation without the parentheses matches. For example,
6168 both the ERE "cd" and the ERE "(cd)" are matched by the third and fourth character of
6169 the string "abcdefabcdef".
- 6170 2. When an ERE matching a single character or an ERE enclosed in parentheses is followed
6171 by the special character <plus-sign> (' + '), together with that <plus-sign> it shall match
6172 what one or more consecutive occurrences of the ERE would match. For example, the
6173 ERE "b+(bc)" matches the fourth to seventh characters in the string "acabbbbcde".
6174 And, "[ab]+" and "[ab][ab]*" are equivalent.
- 6175 3. When an ERE matching a single character or an ERE enclosed in parentheses is followed
6176 by the special character <asterisk> (' * '), together with that <asterisk> it shall match
6177 what zero or more consecutive occurrences of the ERE would match. For example, the
6178 ERE "b*c" matches the first character in the string "cabbbbcde", and the ERE "b*cd"
6179 matches the third to seventh characters in the string "cabbbbcdebbbbbbbcdbbc". And,
6180 "[ab]*" and "[ab][ab]" are equivalent when matching the string "ab".
- 6181 4. When an ERE matching a single character or an ERE enclosed in parentheses is followed
6182 by the special character <question-mark> (' ? '), together with that <question-mark> it
6183 shall match what zero or one consecutive occurrences of the ERE would match. For
6184 example, the ERE "b?c" matches the second character in the string "acabbbbcde".

5. When an ERE matching a single character or an ERE enclosed in parentheses is followed by an interval expression of the format " $\{m\}$ ", " $\{m, \}$ ", or " $\{m, n\}$ ", together with that interval expression it shall match what repeated consecutive occurrences of the ERE would match. The values of m and n are decimal integers in the range $0 \leq m \leq n \leq \{RE\_DUP\_MAX\}$, where m specifies the exact or minimum number of occurrences and n specifies the maximum number of occurrences. The expression " $\{m\}$ " matches exactly m occurrences of the preceding ERE, " $\{m, \}$ " matches at least m occurrences, and " $\{m, n\}$ " matches any number of occurrences between m and n , inclusive.

For example, in the string "abababcccccccd" the ERE " $c\{3\}$ " is matched by characters seven to nine and the ERE " $(ab)\{2, \}$ " is matched by characters one to six.

The behavior of multiple adjacent duplication symbols ('+', '*', '?', and intervals) produces undefined results.

An ERE matching a single character repeated by an '*', '?', or an interval expression shall not match a null expression unless this is the only match for the repetition or it is necessary to satisfy the exact or minimum number of occurrences for the interval expression.

9.4.7 ERE Alternation

Two EREs separated by the special character <vertical-line> ('|') shall match a string that is matched by either. For example, the ERE " $a((bc)|d)$ " matches the string "abc" and the string "ad". Single characters, or expressions matching single characters, separated by the <vertical-line> and enclosed in parentheses, shall be treated as an ERE matching a single character.

9.4.8 ERE Precedence

The order of precedence shall be as shown in the following table:

ERE Precedence (from high to low)	
Collation-related bracket symbols	[==] [::] [..]
Escaped characters	\<special character>
Bracket expression	[]
Grouping	()
Single-character-ERE duplication	* + ? {m,n}
Concatenation	
Anchoring	^ \$
Alternation	

For example, the ERE " $abba|cde$ " matches either the string "abba" or the string "cde" (rather than the string "abbade" or "abbcde", because concatenation has a higher order of precedence than alternation).

9.4.9 ERE Expression Anchoring

An ERE can be limited to matching strings that begin or end a line; this is called “anchoring”. The <circumflex> and <dollar-sign> special characters shall be considered ERE anchors when used anywhere outside a bracket expression. This shall have the following effects:

1. A <circumflex> ('^') outside a bracket expression shall anchor the expression or subexpression it begins to the beginning of a string; such an expression or subexpression can match only a sequence starting at the first character of a string. For example, the EREs "^ab" and "(^ab)" match "ab" in the string "abcdef", but fail to match in the string "cdefab", and the ERE "a^b" is valid, but can never match because the 'a' prevents the expression "a^b" from matching starting at the first character.
2. A <dollar-sign> ('\$') outside a bracket expression shall anchor the expression or subexpression it ends to the end of a string; such an expression or subexpression can match only a sequence ending at the last character of a string. For example, the EREs "ef\$" and "(ef\$)" match "ef" in the string "abcdef", but fail to match in the string "cdefab", and the ERE "e\$f" is valid, but can never match because the 'f' prevents the expression "e\$" from matching ending at the last character.

9.5 Regular Expression Grammar

Grammars describing the syntax of both basic and extended regular expressions are presented in this section. The grammar takes precedence over the text. See XCU [Section 1.3](#) (on page 2309).

9.5.1 BRE/ERE Grammar Lexical Conventions

The lexical conventions for regular expressions are as described in this section.

Except as noted, the longest possible token or delimiter beginning at a given point is recognized.

The following tokens are processed (in addition to those string constants shown in the grammar):

COLL_ELEM_SINGLE	Any single-character collating element, unless it is a META_CHAR .
COLL_ELEM_MULTI	Any multi-character collating element.
BACKREF	Applicable only to basic regular expressions. The character string consisting of a <backslash> character followed by a single-digit numeral, '1' to '9'.
DUP_COUNT	Represents a numeric constant. It shall be an integer in the range 0 ≤ DUP_COUNT ≤ { RE_DUP_MAX }. This token is only recognized when the context of the grammar requires it. At all other times, digits not preceded by a <backslash> character are treated as ORD_CHAR .
META_CHAR	One of the characters: ^ When found first in a bracket expression – When found anywhere but first (after an initial '^', if any) or last in a bracket expression, or as the ending range point in a range expression

6258		]	When found anywhere but first (after an initial ' ^ ', if any) in a bracket expression
6260	L_ANCHOR		Applicable only to basic regular expressions. The character ' ^ ' when it appears as the first character of a basic regular expression and when not QUOTED_CHAR . The ' ^ ' may be recognized as an anchor elsewhere; see Section 9.3.8 (on page 187).
6264	ORD_CHAR		A character, other than one of the special characters in SPEC_CHAR .
6265	QUOTED_CHAR		In a BRE, one of the character sequences:
6266		\ ^ \ . \ * \ [\ \$ \ \	
6267			In an ERE, one of the character sequences:
6268		\ ^ \ . \ [\ \$ \ (\) \	
6269		\ * \ + \ ? \ { \ \	
6270	R_ANCHOR		(Applicable only to basic regular expressions.) The character ' \$ ' when it appears as the last character of a basic regular expression and when not QUOTED_CHAR . The ' \$ ' may be recognized as an anchor elsewhere; see Section 9.3.8 (on page 187).
6274	SPEC_CHAR		For basic regular expressions, one of the following special characters:
6275		.	Anywhere outside bracket expressions
6276		\	Anywhere outside bracket expressions
6277		[	Anywhere outside bracket expressions
6278		^	When used as an anchor (see Section 9.3.8 , on page 187) or when first in a bracket expression
6279		\$	When used as an anchor
6281		*	Anywhere except first in an entire RE, anywhere in a bracket expression, directly following " \(", directly following an anchoring ' ^ '
6284			For extended regular expressions, shall be one of the following special characters found anywhere outside bracket expressions:
6285		^ . [\$ ()	
6286		* + ? { \	
6288			(The close-parenthesis shall be considered special in this context only if matched with a preceding open-parenthesis.)
6289			

9.5.2 RE and Bracket Expression Grammar

This section presents the grammar for basic regular expressions, including the bracket expression grammar that is common to both BREs and EREs.

6293	%token	ORD_CHAR	QUOTED_CHAR	DUP_COUNT
6294	%token	BACKREF	L_ANCHOR	R_ANCHOR
6295	%token	Back_open_paren	Back_close_paren	
6296	/*	' \'	' \')	*/
6297	%token	Back_open_brace	Back_close_brace	

```

6298      /*      '\{'      '\}'      */
6299      /* The following tokens are for the Bracket Expression
6300         grammar common to both REs and EREs. */
6301      %token      COLL_ELEM_SINGLE COLL_ELEM_MULTI META_CHAR
6302      %token      Open_equal Equal_close Open_dot Dot_close Open_colon Colon_close
6303      /*      '['      '='      '['      '.'      ':'      ':'      */
6304      %token      class_name
6305      /* class_name is a keyword to the LC_CTYPE locale category */
6306      /* (representing a character class) in the current locale */
6307      /* and is only recognized between [: and :] */
6308      %start      basic_reg_exp
6309      %%
6310      /* -----
6311         Basic Regular Expression
6312         -----
6313      */
6314      basic_reg_exp :      RE_expression
6315                    | L_ANCHOR
6316                    |      R_ANCHOR
6317                    | L_ANCHOR      R_ANCHOR
6318                    | L_ANCHOR RE_expression
6319                    |      RE_expression R_ANCHOR
6320                    | L_ANCHOR RE_expression R_ANCHOR
6321                    ;
6322      RE_expression :      simple_RE
6323                    | RE_expression simple_RE
6324                    ;
6325      simple_RE : nondupl_RE
6326               | nondupl_RE RE_dupl_symbol
6327               ;
6328      nondupl_RE : one_char_or_coll_elem_RE
6329                | Back_open_paren RE_expression Back_close_paren
6330                | BACKREF
6331                ;
6332      one_char_or_coll_elem_RE : ORD_CHAR
6333                               | QUOTED_CHAR
6334                               | '.'
6335                               | bracket_expression
6336                               ;
6337      RE_dupl_symbol : '*'
6338                    | Back_open_brace DUP_COUNT      Back_close_brace
6339                    | Back_open_brace DUP_COUNT ','      Back_close_brace
6340                    | Back_open_brace DUP_COUNT ',' DUP_COUNT Back_close_brace
6341                    ;
6342      /* -----
6343         Bracket Expression
6344         -----
6345      */
6346      bracket_expression : '[' matching_list ']'

```

```

6347         | '[' nonmatching_list ']'
6348         ;
6349     matching_list : bracket_list
6350         ;
6351     nonmatching_list : '^' bracket_list
6352         ;
6353     bracket_list : follow_list
6354         | follow_list '-'
6355         ;
6356     follow_list : expression_term
6357         | follow_list expression_term
6358         ;
6359     expression_term : single_expression
6360         | range_expression
6361         ;
6362     single_expression : end_range
6363         | character_class
6364         | equivalence_class
6365         ;
6366     range_expression : start_range end_range
6367         | start_range '-'
6368         ;
6369     start_range : end_range '-'
6370         ;
6371     end_range : COLL_ELEM_SINGLE
6372         | collating_symbol
6373         ;
6374     collating_symbol : Open_dot COLL_ELEM_SINGLE Dot_close
6375         | Open_dot COLL_ELEM_MULTI Dot_close
6376         | Open_dot META_CHAR Dot_close
6377         ;
6378     equivalence_class : Open_equal COLL_ELEM_SINGLE Equal_close
6379         | Open_equal COLL_ELEM_MULTI Equal_close
6380         ;
6381     character_class : Open_colon class_name Colon_close
6382         ;

```

The BRE grammar does not permit **L_ANCHOR** or **R_ANCHOR** inside "\(" and "\)" (which implies that '^' and '\$' are ordinary characters). This reflects the semantic limits on the application, as noted in [Section 9.3.8](#) (on page 187). Implementations are permitted to extend the language to interpret '^' and '\$' as anchors in these locations, and as such, conforming applications cannot use unescaped '^' and '\$' in positions inside "\(" and "\)" that might be interpreted as anchors.

9.5.3 ERE Grammar

This section presents the grammar for extended regular expressions, excluding the bracket expression grammar.

Note: The bracket expression grammar and the associated **%token** lines are identical between BREs and EREs. It has been omitted from the ERE section to avoid unnecessary editorial duplication.

```
%token  ORD_CHAR QUOTED_CHAR DUP_COUNT
%start  extended_reg_exp
%%

/* -----
   Extended Regular Expression
   ----- */
extended_reg_exp      :      ERE_branch
                       | extended_reg_exp ' | ' ERE_branch
                       ;
ERE_branch             :      ERE_expression
                       | ERE_branch ERE_expression
                       ;
ERE_expression        : one_char_or_coll_elem_ERE
                       | '^'
                       | '$'
                       | '(' extended_reg_exp ')'
                       | ERE_expression ERE_dupl_symbol
                       ;
one_char_or_coll_elem_ERE : ORD_CHAR
                       | QUOTED_CHAR
                       | '.'
                       | bracket_expression
                       ;
ERE_dupl_symbol        : '*'
                       | '+'
                       | '?'
                       | '{' DUP_COUNT '}'
                       | '{' DUP_COUNT ',' '}'
                       | '{' DUP_COUNT ',' DUP_COUNT '}'
                       ;
```

The ERE grammar does not permit several constructs that previous sections specify as having undefined results:

- **ORD_CHAR** preceded by a <backslash> character
- One or more *ERE_dupl_symbols* appearing first in an ERE, or immediately following ' | ', '^', or ' ('
- '{' not part of a valid *ERE_dupl_symbol*
- '| ' appearing first or last in an ERE, or immediately following ' | ' or ' (', or immediately preceding ') '

Implementations are permitted to extend the language to allow these. Conforming applications cannot use such constructs.

6435
6436

6437 **10.1 Directory Structure and Files**

6438 The following directories shall exist on conforming systems and conforming applications shall
6439 make use of them only as described. Strictly conforming applications shall not assume the
6440 ability to create files in any of these directories, unless specified below.

6441 **/** The root directory.

6442 **/dev** Contains **/dev/console**, **/dev/null**, and **/dev/tty**, described below.

6443 The following directory shall exist on conforming systems and shall be used as described:

6444 **/tmp** A directory made available for applications that need a place to create temporary
6445 files. Applications shall be allowed to create files in this directory, but shall not
6446 assume that such files are preserved between invocations of the application.

6447 The following files shall exist on conforming systems and shall be both readable and writable:

6448 **/dev/null** An empty data source and infinite data sink. Data written to **/dev/null** shall be
6449 discarded. Reads from **/dev/null** shall always return end-of-file (EOF).

6450 **/dev/tty** In each process, a synonym for the controlling terminal associated with the process
6451 group of that process, if any. It is useful for programs or shell procedures that wish
6452 to be sure of writing messages to or reading data from the terminal no matter how
6453 output has been redirected. It can also be used for applications that demand the
6454 name of a file for output, when typed output is desired and it is tiresome to find
6455 out what terminal is currently in use.

6456 The following file shall exist on conforming systems and need not be readable or writable:

6457 **/dev/console** The **/dev/console** file is a generic name given to the system console (see [Section](#)
6458 [3.386](#), on page 95). It is usually linked to an implementation-defined special file. It
6459 shall provide an interface to the system console conforming to the requirements of
6460 [Chapter 11](#) (on page 199).

10.2 Output Devices and Terminal Types

The utilities in the Shell and Utilities volume of POSIX.1-2008 historically have been implemented on a wide range of terminal types, but a conforming implementation need not support all features of all utilities on every conceivable terminal. POSIX.1-2008 states which features are optional for certain classes of terminals in the individual utility description sections. The implementation shall document in the system documentation which terminal types it supports and which of these features and utilities are not supported by each terminal.

When a feature or utility is not supported on a specific terminal type, as allowed by POSIX.1-2008, and the implementation considers such a condition to be an error preventing use of the feature or utility, the implementation shall indicate such conditions through diagnostic messages or exit status values or both (as appropriate to the specific utility description) that inform the user that the terminal type lacks the appropriate capability.

POSIX.1-2008 uses a notational convention based on historical practice that identifies some of the control characters defined in [Section 7.3.1](#) (on page 139) in a manner easily remembered by users on many terminals. The correspondence between this “<control>-char” notation and the actual control characters is shown in the following table. When POSIX.1-2008 refers to a character by its <control>-name, it is referring to the actual control character shown in the Value column of the table, which is not necessarily the exact control key sequence on all terminals. Some terminals have keyboards that do not allow the direct transmission of all the non-alphanumeric characters shown. In such cases, the system documentation shall describe which data sequences transmitted by the terminal are interpreted by the system as representing the special characters.

Table 10-1 Control Character Names

Name	Value	Symbolic Name	Name	Value	Symbolic Name
<control>-A	<SOH>	<SOH>	<control>-Q	<DC1>	<DC1>
<control>-B	<STX>	<STX>	<control>-R	<DC2>	<DC2>
<control>-C	<ETX>	<ETX>	<control>-S	<DC3>	<DC3>
<control>-D	<EOT>	<EOT>	<control>-T	<DC4>	<DC4>
<control>-E	<ENQ>	<ENQ>	<control>-U	<NAK>	<NAK>
<control>-F	<ACK>	<ACK>	<control>-V	<SYN>	<SYN>
<control>-G	<BEL>	<alert>	<control>-W	<ETB>	<ETB>
<control>-H	<BS>	<backspace>	<control>-X	<CAN>	<CAN>
<control>-I	<HT>	<tab>	<control>-Y		
<control>-J	<LF>	<linefeed>	<control>-Z	<SUB>	<SUB>
<control>-K	<VT>	<vertical-tab>	<control>-[	<ESC>	<ESC>
<control>-L	<FF>	<form-feed>	<control>-\ <control>-]	<FS>	<FS>
<control>-M	<CR>	<carriage-return>	<control>-]	<GS>	<GS>
<control>-N	<SO>	<SO>	<control>-^	<RS>	<RS>
<control>-O	<SI>	<SI>	<control>-_ <control>-?	<US>	<US>
<control>-P	<DLE>	<DLE>	<control>-?	<DEL>	<DEL>

Note: The notation uses uppercase letters for arbitrary editorial reasons. There is no implication that the keystrokes represent control-shift-letter sequences.

6503

6504

General Terminal Interface

6505 This chapter describes a general terminal interface that shall be provided. It shall be supported
6506 on any asynchronous communications ports if the implementation provides them. It is
6507 implementation-defined whether it supports network connections or synchronous ports, or
6508 both.

6509 11.1 Interface Characteristics

6510 11.1.1 Opening a Terminal Device File

6511 When a terminal device file is opened, it normally causes the thread to wait until a connection is
6512 established. In practice, application programs seldom open these files; they are opened by
6513 special programs and become an application's standard input, output, and error files.

6514 Cases where applications do open a terminal device are as follows:

- 6515 1. Opening `/dev/tty`, or the pathname returned by `ctermid()`, in order to obtain a file
6516 descriptor for the controlling terminal; see [Section 11.1.3](#) (on page 200).
- 6517 2. Opening the slave side of a pseudo-terminal; see XSH [ptsname\(\)](#).
- 6518 3. Opening a modem or similar piece of equipment connected by a serial line. In this case,
6519 the terminal parameters (see [Section 11.2](#), on page 205) may be initialized to default
6520 settings by the implementation in between the last close of the device by any process and
6521 the next open of the device, or they may persist from one use to the next. The terminal
6522 parameters can be set to values that ensure the terminal behaves in a conforming manner
6523 by means of the `O_TTY_INIT` open flag when opening a terminal device that is not
6524 already open in any process, or by executing the `stty` utility with the operand **sane**.

6525 As described in `open()`, opening a terminal device file with the `O_NONBLOCK` flag clear shall
6526 cause the thread to block until the terminal device is ready and available. If `CLOCAL` mode is
6527 not set, this means blocking until a connection is established. If `CLOCAL` mode is set in the
6528 terminal, or the `O_NONBLOCK` flag is specified in the `open()`, the `open()` function shall return a
6529 file descriptor without waiting for a connection to be established.

11.1.2 Process Groups

A terminal may have a foreground process group associated with it. This foreground process group plays a special role in handling signal-generating input characters, as discussed in [Section 11.1.9](#) (on page 204).

A command interpreter process supporting job control can allocate the terminal to different jobs, or process groups, by placing related processes in a single process group and associating this process group with the terminal. A terminal's foreground process group may be set or examined by a process, assuming the permission requirements are met; see *tcgetpgrp()* and *tcsetpgrp()*. The terminal interface aids in this allocation by restricting access to the terminal by processes that are not in the current process group; see [Section 11.1.4](#) (on page 201).

When there is no longer any process whose process ID or process group ID matches the foreground process group ID, the terminal shall have no foreground process group. It is unspecified whether the terminal has a foreground process group when there is a process whose process ID matches the foreground process group ID, but whose process group ID does not. No actions defined in POSIX.1-2008, other than allocation of a controlling terminal or a successful call to *tcsetpgrp()*, shall cause a process group to become the foreground process group of the terminal.

11.1.3 The Controlling Terminal

A terminal may belong to a process as its controlling terminal. Each process of a session that has a controlling terminal has the same controlling terminal. A terminal may be the controlling terminal for at most one session. The controlling terminal for a session is allocated by the session leader in an implementation-defined manner. If a session leader has no controlling terminal, and opens a terminal device file that is not already associated with a session without using the *O_NOCTTY* option (see *open()*), it is implementation-defined whether the terminal becomes the controlling terminal of the session leader. If a process which is not a session leader opens a terminal file, or the *O_NOCTTY* option is used on *open()*, then that terminal shall not become the controlling terminal of the calling process. When a controlling terminal becomes associated with a session, its foreground process group shall be set to the process group of the session leader.

The controlling terminal is inherited by a child process during a *fork()* function call. A process relinquishes its controlling terminal when it creates a new session with the *setsid()* function; other processes remaining in the old session that had this terminal as their controlling terminal continue to have it. Upon the close of the last file descriptor in the system (whether or not it is in the current session) associated with the controlling terminal, it is unspecified whether all processes that had that terminal as their controlling terminal cease to have any controlling terminal. Whether and how a session leader can reacquire a controlling terminal after the controlling terminal has been relinquished in this fashion is unspecified. A process does not relinquish its controlling terminal simply by closing all of its file descriptors associated with the controlling terminal if other processes continue to have it open.

When a controlling process terminates, the controlling terminal is dissociated from the current session, allowing it to be acquired by a new session leader. Subsequent access to the terminal by other processes in the earlier session may be denied, with attempts to access the terminal treated as if a modem disconnect had been sensed.

11.1.4 Terminal Access Control

If a process is in the foreground process group of its controlling terminal, read operations shall be allowed, as described in [Section 11.1.5](#). Any attempts by a process in a background process group to read from its controlling terminal cause its process group to be sent a SIGTTIN signal unless one of the following special cases applies: if the reading process is ignoring the SIGTTIN signal or the reading thread is blocking the SIGTTIN signal, or if the process group of the reading process is orphaned, the *read()* shall return -1 , with *errno* set to [EIO] and no signal shall be sent. The default action of the SIGTTIN signal shall be to stop the process to which it is sent. See [<signal.h>](#).

If a process is in the foreground process group of its controlling terminal, write operations shall be allowed as described in [Section 11.1.8](#) (on page 203). Attempts by a process in a background process group to write to its controlling terminal shall cause the process group to be sent a SIGTTOU signal unless one of the following special cases applies: if TOSTOP is not set, or if TOSTOP is set and the process is ignoring the SIGTTOU signal or the writing thread is blocking the SIGTTOU signal, the process is allowed to write to the terminal and the SIGTTOU signal is not sent. If TOSTOP is set, the process group of the writing process is orphaned, the writing process is not ignoring the SIGTTOU signal, and the writing thread is not blocking the SIGTTOU signal, the *write()* shall return -1 , with *errno* set to [EIO] and no signal shall be sent.

Certain calls that set terminal parameters are treated in the same fashion as *write()*, except that TOSTOP is ignored; that is, the effect is identical to that of terminal writes when TOSTOP is set (see [Section 11.2.5](#) (on page 210), *tcdrain()*, *tcflow()*, *tcflush()*, *tcsendbreak()*, *tcsetattr()*, and *tcsetpgrp()*).

11.1.5 Input Processing and Reading Data

A terminal device associated with a terminal device file may operate in full-duplex mode, so that data may arrive even while output is occurring. Each terminal device file has an input queue associated with it, into which incoming data is stored by the system before being read by a process. The system may impose a limit, {MAX_INPUT}, on the number of bytes that may be stored in the input queue. The behavior of the system when this limit is exceeded is implementation-defined.

Two general kinds of input processing are available, determined by whether the terminal device file is in canonical mode or non-canonical mode. These modes are described in [Section 11.1.6](#) (on page 202) and [Section 11.1.7](#) (on page 202). Additionally, input characters are processed according to the *c_iflag* (see [Section 11.2.2](#), on page 206) and *c_lflag* (see [Section 11.2.5](#), on page 210) fields. Such processing can include “echoing”, which in general means transmitting input characters immediately back to the terminal when they are received from the terminal. This is useful for terminals that can operate in full-duplex mode.

The manner in which data is provided to a process reading from a terminal device file is dependent on whether the terminal file is in canonical or non-canonical mode, and on whether or not the O_NONBLOCK flag is set by *open()* or *fcntl()*.

If the O_NONBLOCK flag is clear, then the read request shall be blocked until data is available or a signal has been received. If the O_NONBLOCK flag is set, then the read request shall be completed, without blocking, in one of three ways:

1. If there is enough data available to satisfy the entire request, the *read()* shall complete successfully and shall return the number of bytes read.

2. If there is not enough data available to satisfy the entire request, the `read()` shall complete successfully, having read as much data as possible, and shall return the number of bytes it was able to read.
3. If there is no data available, the `read()` shall return `-1`, with `errno` set to `[EAGAIN]`.

When data is available depends on whether the input processing mode is canonical or non-canonical. [Section 11.1.6](#) and [Section 11.1.7](#) describe each of these input processing modes.

11.1.6 Canonical Mode Input Processing

In canonical mode input processing, terminal input is processed in units of lines. A line is delimited by a <newline> character (NL), an end-of-file character (EOF), or an end-of-line (EOL) character. See [Section 11.1.9](#) (on page 204) for more information on EOF and EOL. This means that a read request shall not return until an entire line has been typed or a signal has been received. Also, no matter how many bytes are requested in the `read()` call, at most one line shall be returned. It is not, however, necessary to read a whole line at once; any number of bytes, even one, may be requested in a `read()` without losing information.

If `{MAX_CANON}` is defined for this terminal device, it shall be a limit on the number of bytes in a line. The behavior of the system when this limit is exceeded is implementation-defined. If `{MAX_CANON}` is not defined, there shall be no such limit; see `pathconf()`.

Erase and kill processing occur when either of two special characters, the ERASE and KILL characters (see [Section 11.1.9](#), on page 204), is received. This processing shall affect data in the input queue that has not yet been delimited by an NL, EOF, or EOL character. This un-delimited data makes up the current line. The ERASE character shall delete the last character in the current line, if there is one. The KILL character shall delete all data in the current line, if there is any. The ERASE and KILL characters shall have no effect if there is no data in the current line. The ERASE and KILL characters themselves shall not be placed in the input queue.

11.1.7 Non-Canonical Mode Input Processing

In non-canonical mode input processing, input bytes are not assembled into lines, and erase and kill processing shall not occur. The values of the MIN and TIME members of the `c_cc` array are used to determine how to process the bytes received. POSIX.1-2008 does not specify whether the setting of `O_NONBLOCK` takes precedence over MIN or TIME settings. Therefore, if `O_NONBLOCK` is set, `read()` may return immediately, regardless of the setting of MIN or TIME. Also, if no data is available, `read()` may either return 0, or return `-1` with `errno` set to `[EAGAIN]`.

MIN represents the minimum number of bytes that should be received when the `read()` function returns successfully. TIME is a timer of 0.1 second granularity that is used to time out bursty and short-term data transmissions. If MIN is greater than `{MAX_INPUT}`, the response to the request is undefined. The four possible values for MIN and TIME and their interactions are described below.

Case A: MIN>0, TIME>0

In case A, TIME serves as an inter-byte timer which shall be activated after the first byte is received. Since it is an inter-byte timer, it shall be reset after a byte is received. The interaction between MIN and TIME is as follows. As soon as one byte is received, the inter-byte timer shall be started. If MIN bytes are received before the inter-byte timer expires (remember that the timer is reset upon receipt of each byte), the read shall be satisfied. If the timer expires before MIN bytes are received, the characters received to that point shall be returned to the user. Note that if TIME expires at least one byte shall be returned because the timer would not have been enabled unless a byte was received. In this case (MIN>0, TIME>0) the read shall block until the MIN and TIME mechanisms are activated by the receipt of the first byte, or a signal is received. If data is in the buffer at the time of the *read()*, the result shall be as if data has been received immediately after the *read()*.

Case B: MIN>0, TIME=0

In case B, since the value of TIME is zero, the timer plays no role and only MIN is significant. A pending read shall not be satisfied until MIN bytes are received (that is, the pending read shall block until MIN bytes are received), or a signal is received. A program that uses case B to read record-based terminal I/O may block indefinitely in the read operation.

Case C: MIN=0, TIME>0

In case C, since MIN=0, TIME no longer represents an inter-byte timer. It now serves as a read timer that shall be activated as soon as the *read()* function is processed. A read shall be satisfied as soon as a single byte is received or the read timer expires. Note that in case C if the timer expires, no bytes shall be returned. If the timer does not expire, the only way the read can be satisfied is if a byte is received. If bytes are not received, the read shall not block indefinitely waiting for a byte; if no byte is received within TIME*0.1 seconds after the read is initiated, the *read()* shall return a value of zero, having read no data. If data is in the buffer at the time of the *read()*, the timer shall be started as if data has been received immediately after the *read()*.

Case D: MIN=0, TIME=0

The minimum of either the number of bytes requested or the number of bytes currently available shall be returned without waiting for more bytes to be input. If no characters are available, *read()* shall return a value of zero, having read no data.

11.1.8 Writing Data and Output Processing

When a process writes one or more bytes to a terminal device file, they are processed according to the *c_oflag* field (see [Section 11.2.3](#), on page 207). The implementation may provide a buffering mechanism; as such, when a call to *write()* completes, all of the bytes written have been scheduled for transmission to the device, but the transmission has not necessarily completed. See *write()* for the effects of O_NONBLOCK on *write()*.

11.1.9 Special Characters

Certain characters have special functions on input or output or both. These functions are summarized as follows:

INTR Special character on input, which is recognized if the ISIG flag is set. Generates a SIGINT signal which is sent to all processes in the foreground process group for which the terminal is the controlling terminal. If ISIG is set, the INTR character shall be discarded when processed.

QUIT Special character on input, which is recognized if the ISIG flag is set. Generates a SIGQUIT signal which is sent to all processes in the foreground process group for which the terminal is the controlling terminal. If ISIG is set, the QUIT character shall be discarded when processed.

ERASE Special character on input, which is recognized if the ICANON flag is set. Erases the last character in the current line; see [Section 11.1.6](#) (on page 202). It shall not erase beyond the start of a line, as delimited by an NL, EOF, or EOL character. If ICANON is set, the ERASE character shall be discarded when processed.

KILL Special character on input, which is recognized if the ICANON flag is set. Deletes the entire line, as delimited by an NL, EOF, or EOL character. If ICANON is set, the KILL character shall be discarded when processed.

EOF Special character on input, which is recognized if the ICANON flag is set. When received, all the bytes waiting to be read are immediately passed to the process without waiting for a <newline>, and the EOF is discarded. Thus, if there are no bytes waiting (that is, the EOF occurred at the beginning of a line), a byte count of zero shall be returned from the *read()*, representing an end-of-file indication. If ICANON is set, the EOF character shall be discarded when processed.

NL Special character on input, which is recognized if the ICANON flag is set. It is the line delimiter <newline>. It cannot be changed.

EOL Special character on input, which is recognized if the ICANON flag is set. It is an additional line delimiter, like NL.

SUSP If the ISIG flag is set, receipt of the SUSP character shall cause a SIGTSTP signal to be sent to all processes in the foreground process group for which the terminal is the controlling terminal, and the SUSP character shall be discarded when processed.

STOP Special character on both input and output, which is recognized if the IXON (output control) or IXOFF (input control) flag is set. Can be used to suspend output temporarily. It is useful with CRT terminals to prevent output from disappearing before it can be read. If IXON is set, the STOP character shall be discarded when processed.

START Special character on both input and output, which is recognized if the IXON (output control) or IXOFF (input control) flag is set. Can be used to resume output that has been suspended by a STOP character. If IXON is set, the START character shall be discarded when processed.

CR Special character on input, which is recognized if the ICANON flag is set; it is the carriage-return character. When ICANON and ICRNL are set and IGNCR is not set, this character shall be translated into an NL, and shall have the same effect as an NL character.

The NL and CR characters cannot be changed. It is implementation-defined whether the START and STOP characters can be changed. The values for INTR, QUIT, ERASE, KILL, EOF, EOL, and SUSP shall be changeable to suit individual tastes. Special character functions associated with

changeable special control characters can be disabled individually.

If two or more special characters have the same value, the function performed when that character is received is undefined.

A special character is recognized not only by its value, but also by its context; for example, an implementation may support multi-byte sequences that have a meaning different from the meaning of the bytes when considered individually. Implementations may also support additional single-byte functions. These implementation-defined multi-byte or single-byte functions shall be recognized only if the IEXTEN flag is set; otherwise, data is received without interpretation, except as required to recognize the special characters defined in this section.

XSI If IEXTEN is set, the ERASE, KILL, and EOF characters can be escaped by a preceding `<backslash>` character, in which case no special function shall occur.

11.1.10 Modem Disconnect

If a modem disconnect is detected by the terminal interface for a controlling terminal, and if CLOCAL is not set in the `c_cflag` field for the terminal (see [Section 11.2.4](#), on page 209), the SIGHUP signal shall be sent to the controlling process for which the terminal is the controlling terminal. Unless other arrangements have been made, this shall cause the controlling process to terminate (see `exit()`). Any subsequent read from the terminal device shall return the value of zero, indicating end-of-file; see `read()`. Thus, processes that read a terminal file and test for end-of-file can terminate appropriately after a disconnect. If the EIO condition as specified in `read()` also exists, it is unspecified whether on EOF condition or [EIO] is returned. Any subsequent `write()` to the terminal device shall return `-1`, with `errno` set to [EIO], until the device is closed.

11.1.11 Closing a Terminal Device File

The last process to close a terminal device file shall cause any output to be sent to the device and shall cause any input to be discarded. If HUPCL is set in the control modes and the communications port supports a disconnect function, the terminal device shall perform a disconnect.

11.2 Parameters that Can be Set

11.2.1 The `termios` Structure

Routines that need to control certain terminal I/O characteristics shall do so by using the `termios` structure as defined in the `<termios.h>` header.

Since the `termios` structure may include additional members, and the standard members may include both standard and non-standard modes, the structure should never be initialized directly by the application as this may cause the terminal to behave in a non-conforming manner. When opening a terminal device (other than a pseudo-terminal) that is not already open in any process, it should be opened with the `O_TTY_INIT` flag before initializing the structure using `tcgetattr()` to ensure that any non-standard elements of the `termios` structure are set to values that result in conforming behavior of the terminal interface.

The members of the **termios** structure include (but are not limited to):

Member Type	Array Size	Member Name	Description
tcflag_t	NCCS	<i>c_iflag</i>	Input modes.
tcflag_t		<i>c_oflag</i>	Output modes.
tcflag_t		<i>c_cflag</i>	Control modes.
tcflag_t		<i>c_lflag</i>	Local modes.
cc_t		<i>c_cc[]</i>	Control characters.

The **tcflag_t** and **cc_t** types are defined in the **<termios.h>** header. They shall be unsigned integer types.

11.2.2 Input Modes

Values of the *c_iflag* field describe the basic terminal input control, and are composed of the bitwise-inclusive OR of the masks shown, which shall be bitwise-distinct. The mask name symbols in this table are defined in **<termios.h>**:

Mask Name	Description
BRKINT	Signal interrupt on break.
ICRNL	Map CR to NL on input.
IGNBRK	Ignore break condition.
IGNCR	Ignore CR.
IGNPAR	Ignore characters with parity errors.
INLCR	Map NL to CR on input.
INPCK	Enable input parity check.
ISTRIP	Strip character.
IXANY	Enable any character to restart output.
IXOFF	Enable start/stop input control.
IXON	Enable start/stop output control.
PARMRK	Mark parity errors.

In the context of asynchronous serial data transmission, a break condition shall be defined as a sequence of zero-valued bits that continues for more than the time to send one byte. The entire sequence of zero-valued bits is interpreted as a single break condition, even if it continues for a time equivalent to more than one byte. In contexts other than asynchronous serial data transmission, the definition of a break condition is implementation-defined.

If IGNBRK is set, a break condition detected on input shall be ignored; that is, not put on the input queue and therefore not read by any process. If IGNBRK is not set and BRKINT is set, the break condition shall flush the input and output queues, and if the terminal is the controlling terminal of a foreground process group, the break condition shall generate a single SIGINT signal to that foreground process group. If neither IGNBRK nor BRKINT is set, a break condition shall be read as a single 0x00, or if PARMRK is set, as 0xff 0x00 0x00.

If IGNPAR is set, a byte with a framing or parity error (other than break) shall be ignored.

If PARMRK is set, and IGNPAR is not set, a byte with a framing or parity error (other than break) shall be given to the application as the three-byte sequence 0xff 0x00 X, where 0xff 0x00 is a two-byte flag preceding each sequence and X is the data of the byte received in error. To avoid ambiguity in this case, if ISTRIP is not set, a valid byte of 0xff is given to the application as 0xff 0xff. If neither PARMRK nor IGNPAR is set, a framing or parity error (other than break) shall be given to the application as a single byte 0x00.

6817 If INPCK is set, input parity checking shall be enabled. If INPCK is not set, input parity checking
6818 shall be disabled, allowing output parity generation without input parity errors. Note that
6819 whether input parity checking is enabled or disabled is independent of whether parity detection
6820 is enabled or disabled (see [Section 11.2.4](#), on page 209). If parity detection is enabled but input
6821 parity checking is disabled, the hardware to which the terminal is connected shall recognize the
6822 parity bit, but the terminal special file shall not check whether or not this bit is correctly set.

6823 If ISTRIP is set, valid input bytes shall first be stripped to seven bits; otherwise, all eight bits
6824 shall be processed.

6825 If INLCR is set, a received NL character shall be translated into a CR character. If IGNCR is set, a
6826 received CR character shall be ignored (not read). If IGNCR is not set and ICRNL is set, a
6827 received CR character shall be translated into an NL character.

6828 If IXANY is set, any input character shall restart output that has been suspended.

6829 If IXON is set, start/stop output control shall be enabled. A received STOP character shall
6830 suspend output and a received START character shall restart output. When IXON is set, START
6831 and STOP characters are not read, but merely perform flow control functions. When IXON is not
6832 set, the START and STOP characters shall be read.

6833 If IXOFF is set, start/stop input control shall be enabled. The system shall transmit STOP
6834 characters, which are intended to cause the terminal device to stop transmitting data, as needed
6835 to prevent the input queue from overflowing and causing implementation-defined behavior,
6836 and shall transmit START characters, which are intended to cause the terminal device to resume
6837 transmitting data, as soon as the device can continue transmitting data without risk of
6838 overflowing the input queue. The precise conditions under which STOP and START characters
6839 are transmitted are implementation-defined.

6840 The initial input control value after *open()* is implementation-defined.

6841 11.2.3 Output Modes

6842 The *c_oflag* field specifies the terminal interface's treatment of output, and is composed of the
6843 bitwise-inclusive OR of the masks shown, which shall be bitwise-distinct. The mask name
6844 symbols in the following table are defined in [<termios.h>](#):

6845	Mask Name	Description
6846	OPOST	Perform output processing.
6847	XSI	ONLCR
6848	OCRNL	Map NL to CR-NL on output.
6849	ONOCR	Map CR to NL on output.
6850	ONLRET	No CR output at column 0.
6851	ONLRET	NL performs CR function.
6852	OFILL	Use fill characters for delay.
6853	OFDEL	Fill is DEL, else NUL.
6854	NLDLY	Select newline delays:
6855	NL0	Newline character type 0.
6856	NL1	Newline character type 1.
6857	CRDLY	Select carriage-return delays:
6858	CR0	Carriage-return delay type 0.
6859	CR1	Carriage-return delay type 1.
6860	CR2	Carriage-return delay type 2.
6861	CR3	Carriage-return delay type 3.
6862	TABDLY	Select horizontal-tab delays:
6863	TAB0	Horizontal-tab delay type 0.
6864	TAB1	Horizontal-tab delay type 1.
6865	TAB2	Horizontal-tab delay type 2.
6866	TAB3	Expand tabs to spaces.
6867	BSDLY	Select backspace delays:
6868	BS0	Backspace-delay type 0.
6869	BS1	Backspace-delay type 1.
6870	VTDLY	Select vertical-tab delays:
6871	VT0	Vertical-tab delay type 0.
6872	VT1	Vertical-tab delay type 1.
6873	FFDLY	Select form-feed delays:
6874	FF0	Form-feed delay type 0.
6875	FF1	Form-feed delay type 1.

If OPOST is set, output data shall be post-processed as described below, so that lines of text are modified to appear appropriately on the terminal device; otherwise, characters shall be transmitted without change.

If ONLCR is set, the NL character shall be transmitted as the CR-NL character pair. If OCRNL is set, the CR character shall be transmitted as the NL character. If ONOCR is set, no CR character shall be transmitted when at column 0 (first position). If ONLRET is set, the NL character is assumed to do the carriage-return function; the column pointer shall be set to 0 and the delays specified for CR shall be used. Otherwise, the NL character is assumed to do just the line-feed function; the column pointer remains unchanged. The column pointer shall also be set to 0 if the CR character is actually transmitted.

The delay bits specify how long transmission stops to allow for mechanical or other movement when certain characters are sent to the terminal. In all cases a value of 0 shall indicate no delay. If OFILL is set, fill characters shall be transmitted for delay instead of a timed delay. This is useful for high baud rate terminals which need only a minimal delay. If OFDEL is set, the fill character shall be DEL; otherwise, NUL.

If a <form-feed> or <vertical-tab> delay is specified, it shall last for about 2 seconds.

Newline delay shall last about 0.10 seconds. If ONLRET is set, the carriage-return delays shall be used instead of the newline delays. If OFILL is set, two fill characters shall be transmitted.

Carriage-return delay type 1 shall be dependent on the current column position, type 2 shall be

about 0.10 seconds, and type 3 shall be about 0.15 seconds. If OFILL is set, delay type 1 shall transmit two fill characters, and type 2 four fill characters.

Horizontal-tab delay type 1 shall be dependent on the current column position. Type 2 shall be about 0.10 seconds. Type 3 specifies that <tab> characters shall be expanded into <space> characters. If OFILL is set, two fill characters shall be transmitted for any delay.

Backspace delay shall last about 0.05 seconds. If OFILL is set, one fill character shall be transmitted.

The actual delays depend on line speed and system load.

The initial output control value after *open()* is implementation-defined.

11.2.4 Control Modes

The *c_cflag* field describes the hardware control of the terminal, and is composed of the bitwise-inclusive OR of the masks shown, which shall be bitwise-distinct. The mask name symbols in this table are defined in [<termios.h>](#); not all values specified are required to be supported by the underlying hardware (if any). If the terminal is a pseudo-terminal, it is unspecified whether non-default values are unsupported, or are supported and emulated in software, or are handled by *tcsetattr()*, *tcgetattr()*, and the *stty* utility as if they are supported but have no effect on the behavior of the terminal interface.

Mask Name	Description
CLOCAL	Ignore modem status lines.
CREAD	Enable receiver.
CSIZE	Number of bits transmitted or received per byte:
CS5	5 bits
CS6	6 bits
CS7	7 bits
CS8	8 bits.
CSTOPB	Send two stop bits, else one.
HUPCL	Hang up on last close.
PARENB	Parity enable.
PARODD	Odd parity, else even.

In addition, the input and output baud rates are stored in the **termios** structure. The symbols in the following table are defined in [<termios.h>](#). Not all values specified are required to be supported by the underlying hardware (if any). For pseudo-terminals, the input and output baud rates set in the **termios** structure need not affect the speed of data transmission through the terminal interface.

Note: The term “baud” is used historically here, but is not technically correct. This is properly “bits per second”, which may not be the same as baud. However, the term is used because of the historical usage and understanding.

Name	Description	Name	Description
B0	Hang up	B600	600 baud
B50	50 baud	B1200	1200 baud
B75	75 baud	B1800	1800 baud
B110	110 baud	B2400	2400 baud
B134	134.5 baud	B4800	4800 baud
B150	150 baud	B9600	9600 baud
B200	200 baud	B19200	19200 baud
B300	300 baud	B38400	38400 baud

The following functions are provided for getting and setting the values of the input and output baud rates in the **termios** structure: `cfgetispeed()`, `cfgetospeed()`, `cfsetispeed()`, and `cfsetospeed()`. The effects on the terminal device shall not become effective and not all errors need be detected until the `tcsetattr()` function is successfully called.

The CSIZE bits shall specify the number of transmitted or received bits per byte. If ISTRIP is not set, the value of all the other bits is unspecified. If ISTRIP is set, the value of all but the 7 low-order bits shall be zero, but the value of any other bits beyond CSIZE is unspecified when read. CSIZE shall not include the parity bit, if any. If CSTOPB is set, two stop bits shall be used; otherwise, one stop bit. For example, at 110 baud, two stop bits are normally used.

If CREAD is set, the receiver shall be enabled; otherwise, no characters shall be received.

If PARENB is set, parity generation and detection shall be enabled and a parity bit is added to each byte. If parity is enabled, PARODD shall specify odd parity if set; otherwise, even parity shall be used.

If HUPCL is set, the modem control lines for the port shall be lowered when the last process with the port open closes the port or the process terminates. The modem connection shall be broken.

If CLOCAL is set, a connection shall not depend on the state of the modem status lines. If CLOCAL is clear, the modem status lines shall be monitored.

Under normal circumstances, a call to the `open()` function shall wait for the modem connection to complete. However, if the `O_NONBLOCK` flag is set (see `open()`) or if CLOCAL has been set, the `open()` function shall return immediately without waiting for the connection.

If the object for which the control modes are set is not an asynchronous serial connection, some of the modes may be ignored; for example, if an attempt is made to set the baud rate on a network connection to a terminal on another host, the baud rate need not be set on the connection between that terminal and the machine to which it is directly connected.

The initial hardware control value after `open()` is implementation-defined.

11.2.5 Local Modes

The `c_lflag` field of the argument structure is used to control various functions. It is composed of the bitwise-inclusive OR of the masks shown, which shall be bitwise-distinct. The mask name symbols in this table are defined in [<termios.h>](#).

6970
6971
6972
6973
6974
6975
6976
6977
6978
6979

Mask Name	Description
ECHO	Enable echo.
ECHOE	Echo ERASE as an error correcting backspace.
ECHOK	Echo KILL.
ECHONL	Echo <newline>.
ICANON	Canonical input (erase and kill processing).
IEXTEN	Enable extended (implementation-defined) functions.
ISIG	Enable signals.
NOFLSH	Disable flush after interrupt, quit, or suspend.
TOSTOP	Send SIGTTOU for background output.

6980 If ECHO is set, input characters shall be echoed back to the terminal. If ECHO is clear, input
6981 characters shall not be echoed.

6982 If ECHOE and ICANON are set, the ERASE character shall cause the terminal to erase, if
6983 possible, the last character in the current line from the display. If there is no character to erase, an
6984 implementation may echo an indication that this was the case, or do nothing.

6985 If ECHOK and ICANON are set, the KILL character shall either cause the terminal to erase the
6986 line from the display or shall echo the <newline> character after the KILL character.

6987 If ECHONL and ICANON are set, the <newline> character shall be echoed even if ECHO is not
6988 set.

6989 If ICANON is set, canonical processing shall be enabled. This enables the erase and kill edit
6990 functions, and the assembly of input characters into lines delimited by NL, EOF, and EOL, as
6991 described in [Section 11.1.6](#) (on page 202).

6992 If ICANON is not set, read requests shall be satisfied directly from the input queue. A read shall
6993 not be satisfied until at least MIN bytes have been received or the timeout value TIME expired
6994 between bytes. The time value represents tenths of a second. See [Section 11.1.7](#) (on page 202) for
6995 more details.

6996 If IEXTEN is set, implementation-defined functions shall be recognized from the input data. It is
6997 implementation-defined how IEXTEN being set interacts with ICANON, ISIG, IXON, or IXOFF.
6998 If IEXTEN is not set, implementation-defined functions shall not be recognized and the
6999 corresponding input characters are processed as described for ICANON, ISIG, IXON, and
7000 IXOFF.

7001 If ISIG is set, each input character shall be checked against the special control characters INTR,
7002 QUIT, and SUSP. If an input character matches one of these control characters, the function
7003 associated with that character shall be performed. If ISIG is not set, no checking shall be done.
7004 Thus these special input functions are possible only if ISIG is set.

7005 If NOFLSH is set, the normal flush of the input and output queues associated with the INTR,
7006 QUIT, and SUSP characters shall not be done.

7007 If TOSTOP is set, the signal SIGTTOU shall be sent to the process group of a process that tries to |
7008 write to its controlling terminal if it is not in the foreground process group for that terminal. This |
7009 signal, by default, stops the members of the process group. Otherwise, the output generated by |
7010 that process shall be output to the current output stream. If the writing process is ignoring the |
7011 SIGTTOU signal or the writing thread is blocking the SIGTTOU signal, the process is allowed to |
7012 produce output, and the SIGTTOU signal shall not be sent.

7013 The initial local control value after *open()* is implementation-defined.

11.2.6 Special Control Characters

The special control character values shall be defined by the array `c_cc`. The subscript name and description for each element in both canonical and non-canonical modes are as follows:

Subscript Usage		Description
Canonical Mode	Non-Canonical Mode	
VEOF	VINTR	EOF character
VEOL		EOL character
VERASE		ERASE character
VINTR		INTR character
VKILL		KILL character
	VMIN	MIN value
VQUIT	VQUIT	QUIT character
VSUSP	VSUSP	SUSP character
	VTIME	TIME value
VSTART	VSTART	START character
VSTOP	VSTOP	STOP character

The subscript values are unique, except that the VMIN and VTIME subscripts may have the same values as the VEOF and VEOL subscripts, respectively.

Implementations that do not support changing the START and STOP characters may ignore the character values in the `c_cc` array indexed by the VSTART and VSTOP subscripts when `tcsetattr()` is called, but shall return the value in use when `tcgetattr()` is called.

The initial values of all control characters are implementation-defined.

If the value of one of the changeable special control characters (see [Section 11.1.9](#), on page 204) is `_POSIX_VDISABLE`, that function shall be disabled; that is, no input data is recognized as the disabled special character. If ICANON is not set, the value of `_POSIX_VDISABLE` has no special meaning for the VMIN and VTIME entries of the `c_cc` array.

Utility Conventions

12.1 Utility Argument Syntax

This section describes the argument syntax of the standard utilities and introduces terminology used throughout POSIX.1-2008 for describing the arguments processed by the utilities.

Within POSIX.1-2008, a special notation is used for describing the syntax of a utility's arguments. Unless otherwise noted, all utility descriptions use this notation, which is illustrated by this example (see XCU [Section 2.9.1](#), on page 2338):

```
utility_name[-a] [-b] [-c option_argument]
          [-d|-e] [-f[option_argument]] [operand...]
```

The notation used for the SYNOPSIS sections imposes requirements on the implementors of the standard utilities and provides a simple reference for the application developer or system user.

1. The utility in the example is named *utility_name*. It is followed by options, option-arguments, and operands. The arguments that consist of <hyphen> characters and single letters or digits, such as 'a', are known as "options" (or, historically, "flags"). Certain options are followed by an "option-argument", as shown with [-c *option_argument*]. The arguments following the last options and option-arguments are named "operands".

2. Option-arguments are shown separated from their options by <blank> characters, except when the option-argument is enclosed in the '[' and ']' notation to indicate that it is optional. This reflects the situation in which an optional option-argument (if present) is included within the same argument string as the option; for a mandatory option-argument, it is the next argument. The Utility Syntax Guidelines in [Section 12.2](#) (on page 215) require that the option be a separate argument from its option-argument and that option-arguments not be optional, but there are some exceptions in POSIX.1-2008 to ensure continued operation of historical applications:

- a. If the SYNOPSIS of a standard utility shows an option with a mandatory option-argument (as with [-c *option_argument*] in the example), a conforming application shall use separate arguments for that option and its option-argument. However, a conforming implementation shall also permit applications to specify the option and option-argument in the same argument string without intervening <blank> characters.

- b. If the SYNOPSIS shows an optional option-argument (as with [-f[*option_argument*]] in the example), a conforming application shall place any option-argument for that option directly adjacent to the option in the same argument string, without intervening <blank> characters. If the utility receives an argument containing only the option, it shall behave as specified in its description for an omitted option-argument; it shall not treat the next argument (if any) as the option-argument for that option.

3. Options are usually listed in alphabetical order unless this would make the utility description more confusing. There are no implied relationships between the options based upon the order in which they appear, unless otherwise stated in the OPTIONS section, or unless the exception in Guideline 11 of [Section 12.2](#) (on page 215) applies. If an option that does not have option-arguments is repeated, the results are undefined, unless otherwise stated.

4. Frequently, names of parameters that require substitution by actual values are shown with embedded <underscore> characters. Alternatively, parameters are shown as follows:

```
<parameter name>
```

The angle brackets are used for the symbolic grouping of a phrase representing a single parameter and conforming applications shall not include them in data submitted to the utility.

5. When a utility has only a few permissible options, they are sometimes shown individually, as in the example. Utilities with many flags generally show all of the individual flags (that do not take option-arguments) grouped, as in:

```
utility_name [-abcDxyz] [-p arg] [operand]
```

Utilities with very complex arguments may be shown as follows:

```
utility_name [options] [operands]
```

6. Unless otherwise specified, whenever an operand or option-argument is, or contains, a numeric value:

- The number is interpreted as a decimal integer.
- Numerals in the range 0 to 2 147 483 647 are syntactically recognized as numeric values.
- When the utility description states that it accepts negative numbers as operands or option-arguments, numerals in the range -2 147 483 647 to 2 147 483 647 are syntactically recognized as numeric values.
- Ranges greater than those listed here are allowed.

This does not mean that all numbers within the allowable range are necessarily semantically correct. A standard utility that accepts an option-argument or operand that is to be interpreted as a number, and for which a range of values smaller than that shown above is permitted by the POSIX.1-2008, describes that smaller range along with the description of the option-argument or operand. If an error is generated, the utility's diagnostic message shall indicate that the value is out of the supported range, not that it is syntactically incorrect.

7. Arguments or option-arguments enclosed in the ' [' and '] ' notation are optional and can be omitted. Conforming applications shall not include the ' [' and '] ' symbols in data submitted to the utility.

8. Arguments separated by the ' | ' (<vertical-line>) bar notation are mutually-exclusive. Conforming applications shall not include the ' | ' symbol in data submitted to the utility. Alternatively, mutually-exclusive options and operands may be listed with multiple synopsis lines.

For example:

```
utility_name -d[-a] [-c option_argument] [operand...]
utility_name [-a] [-b] [operand...]
```

When multiple synopsis lines are given for a utility, it is an indication that the utility has mutually-exclusive arguments. These mutually-exclusive arguments alter the functionality of the utility so that only certain other arguments are valid in combination with one of the mutually-exclusive arguments. Only one of the mutually-exclusive arguments is allowed for invocation of the utility. Unless otherwise stated in an accompanying OPTIONS section, the relationships between arguments depicted in the SYNOPSIS sections are mandatory requirements placed on conforming applications. The use of conflicting mutually-exclusive arguments produces undefined results, unless a utility description specifies otherwise. When an option is shown without the '[' and ']' brackets, it means that option is required for that version of the SYNOPSIS. However, it is not required to be the first argument, as shown in the example above, unless otherwise stated.

9. Ellipses ("...") are used to denote that one or more occurrences of an operand are allowed. When an option or an operand followed by ellipses is enclosed in brackets, zero or more options or operands can be specified. The form:

```
utility_name [-g option_argument]...[operand...]
```

indicates that multiple occurrences of the option and its option-argument preceding the ellipses are valid, with semantics as indicated in the OPTIONS section of the utility. (See also Guideline 11 in [Section 12.2](#).)

The form:

```
utility_name -f option_argument [-f option_argument]... [operand...]
```

indicates that the -f option is required to appear at least once and may appear multiple times.

10. When the synopsis line is too long to be printed on a single line in the Shell and Utilities volume of POSIX.1-2008, the indented lines following the initial line are continuation lines. An actual use of the command would appear on a single logical line.

12.2 Utility Syntax Guidelines

The following guidelines are established for the naming of utilities and for the specification of options, option-arguments, and operands. The *getopt()* function in the System Interfaces volume of POSIX.1-2008 assists utilities in handling options and operands that conform to these guidelines.

Operands and option-arguments can contain characters not specified in the portable character set.

The guidelines are intended to provide guidance to the authors of future utilities, such as those written specific to a local system or that are components of a larger application. Some of the standard utilities do not conform to all of these guidelines; in those cases, the OPTIONS sections describe the deviations.

7160	Guideline 1:	Utility names should be between two and nine characters, inclusive.	
7161	Guideline 2:	Utility names should include lowercase letters (the lower character classification) and digits only from the portable character set.	
7162			
7163	Guideline 3:	Each option name should be a single alphanumeric character (the alnum character classification) from the portable character set. The -W (capital-W) option shall be reserved for vendor options.	
7164			
7165			
7166		Multi-digit options should not be allowed.	
7167	Guideline 4:	All options should be preceded by the '-' delimiter character.	
7168	Guideline 5:	One or more options without option-arguments, followed by at most one option that takes an option-argument, should be accepted when grouped behind one '-' delimiter.	
7169			
7170			
7171	Guideline 6:	Each option and option-argument should be a separate argument, except as noted in Section 12.1 (on page 213), item (2).	
7172			
7173	Guideline 7:	Option-arguments should not be optional.	
7174	Guideline 8:	When multiple option-arguments are specified to follow a single option, they should be presented as a single argument, using <comma> characters within that argument or <blank> characters within that argument to separate them.	
7175			
7176			
7177	Guideline 9:	All options should precede operands on the command line.	
7178	Guideline 10:	The first -- argument that is not an option-argument should be accepted as a delimiter indicating the end of options. Any following arguments should be treated as operands, even if they begin with the '-' character.	
7179			
7180			
7181	Guideline 11:	The order of different options relative to one another should not matter, unless the options are documented as mutually-exclusive and such an option is documented to override any incompatible options preceding it. If an option that has option-arguments is repeated, the option and option-argument combinations should be interpreted in the order specified on the command line.	
7182			
7183			
7184			
7185			
7186			
7187	Guideline 12:	The order of operands may matter and position-related interpretations should be determined on a utility-specific basis.	
7188			
7189	Guideline 13:	For utilities that use operands to represent files to be opened for either reading or writing, the '-' operand should be used to mean only standard input (or standard output when it is clear from context that an output file is being specified) or a file named - .	
7190			
7191			
7192			
7193	Guideline 14:	If an argument can be identified according to Guidelines 3 through 10 as an option, or as a group of options without option-arguments behind one '-' delimiter, then it should be treated as such.	
7194			
7195			
7196	The utilities in the Shell and Utilities volume of POSIX.1-2008 that claim conformance to these guidelines shall conform completely to these guidelines as if these guidelines contained the term "shall" instead of "should". On some implementations, the utilities accept usage in violation of these guidelines for backwards-compatibility as well as accepting the required form.		
7197			
7198			
7199			
7200	Where a utility described in the Shell and Utilities volume of POSIX.1-2008 as conforming to these guidelines is required to accept, or not to accept, the operand '-' to mean standard input or output, this usage is explained in the OPERANDS section. Otherwise, if such a utility uses operands to represent files, it is implementation-defined whether the operand '-' stands for standard input (or standard output), or for a file named - .		
7201			
7202			
7203			
7204			

7205 It is recommended that all future utilities and applications use these guidelines to enhance user
7206 portability. The fact that some historical utilities could not be changed (to avoid breaking
7207 existing applications) should not deter this future goal.

7208

7209

Headers

7210 This chapter describes the contents of headers.

7211 Headers contain function prototypes, the definition of symbolic constants, common structures,
 7212 preprocessor macros, and defined types. Each function in the System Interfaces volume of
 7213 POSIX.1-2008 specifies the headers that an application shall include in order to use that function.
 7214 In most cases, only one header is required. These headers are present on an application
 7215 development system; they need not be present on the target execution system.

7216 **Format of Entries**

7217 The entries in this chapter are based on a common format as follows. The only sections relating
 7218 to conformance are the SYNOPSIS and DESCRIPTION.

7219 **NAME**

7220 This section gives the name or names of the entry and briefly states its purpose.

7221 **SYNOPSIS**

7222 This section summarizes the use of the entry being described.

7223 **DESCRIPTION**

7224 This section describes the functionality of the header.

7225 **APPLICATION USAGE**

7226 This section is informative. This section gives warnings and advice to application
 7227 developers about the entry. In the event of conflict between warnings and advice and a
 7228 normative part of this volume of POSIX.1-2008, the normative material is to be taken as
 7229 correct.

7230 **RATIONALE**

7231 This section is informative. This section contains historical information concerning the
 7232 contents of this volume of POSIX.1-2008 and why features were included or discarded
 7233 by the standard developers.

7234 **FUTURE DIRECTIONS**

7235 This section is informative. This section provides comments which should be used as a
 7236 guide to current thinking; there is not necessarily a commitment to adopt these future
 7237 directions.

7238 **SEE ALSO**

7239 This section is informative. This section gives references to related information.

7240 **CHANGE HISTORY**

7241 This section is informative. This section shows the derivation of the entry and any
 7242 significant changes that have been made to it.

7243 NAME

7244 aio.h — asynchronous input and output

7245 SYNOPSIS

7246 #include <aio.h>

7247 DESCRIPTION

7248 The <aio.h> header shall define the **aio_cb** structure, which shall include at least the following
7249 members:

7250	int	aio_fildes	File descriptor.
7251	off_t	aio_offset	File offset.
7252	volatile void *	aio_buf	Location of buffer.
7253	size_t	aio_nbytes	Length of transfer.
7254	int	aio_reqprio	Request priority offset.
7255	struct sigevent	aio_sigevent	Signal number and value.
7256	int	aio_lio_opcode	Operation to be performed.

7257 The <aio.h> header shall define the **off_t**, **pthread_attr_t**, **size_t**, and **ssize_t** types as described
7258 in <sys/types.h>.7259 The <aio.h> header shall define the **struct timespec** structure as described in <time.h>.7260 The tag **sigevent** shall be declared as naming an incomplete structure type, the contents of which
7261 are described in the <signal.h> header.

7262 The <aio.h> header shall define the following symbolic constants:

7263 AIO_ALLDONE A return value indicating that none of the requested operations could be
7264 canceled since they are already complete.7265 AIO_CANCELED A return value indicating that all requested operations have been
7266 canceled.

7267 AIO_NOTCANCELED

7268 A return value indicating that some of the requested operations could not
7269 be canceled since they are in progress.7270 LIO_NOP A *lio_listio()* element operation option indicating that no transfer is
7271 requested.7272 LIO_NOWAIT A *lio_listio()* synchronization operation indicating that the calling thread
7273 is to continue execution while the *lio_listio()* operation is being
7274 performed, and no notification is given when the operation is complete.7275 LIO_READ A *lio_listio()* element operation option requesting a read.7276 LIO_WAIT A *lio_listio()* synchronization operation indicating that the calling thread
7277 is to suspend until the *lio_listio()* operation is complete.7278 LIO_WRITE A *lio_listio()* element operation option requesting a write.7279 The following shall be declared as functions and may also be defined as macros. Function
7280 prototypes shall be provided.

7281	int	aio_cancel(int, struct aio_cb *);
7282	int	aio_error(const struct aio_cb *);
7283	FSC SIO	int aio_fsync(int, struct aio_cb *);
7284	int	aio_read(struct aio_cb *);
7285	ssize_t	aio_return(struct aio_cb *);
7286	int	aio_suspend(const struct aio_cb *const [], int,

```

7287         const struct timespec *);
7288     int      aio_write(struct aiocb *);
7289     int      lio_listio(int, struct aiocb *restrict const [restrict], int,
7290         struct sigevent *restrict);

```

7291 Inclusion of the <aio.h> header may make visible symbols defined in the headers <fcntl.h>,
 7292 <signal.h>, and <time.h>.

7293 APPLICATION USAGE

7294 None.

7295 RATIONALE

7296 None.

7297 FUTURE DIRECTIONS

7298 None.

7299 SEE ALSO

7300 <fcntl.h>, <signal.h>, <sys/types.h>, <time.h>

7301 XSH *aio_cancel()*, *aio_error()*, *aio_fsync()*, *aio_read()*, *aio_return()*, *aio_suspend()*, *aio_write()*,
 7302 *fsync()*, *lio_listio()*, *lseek()*, *read()*, *write()*

7303 CHANGE HISTORY

7304 First released in Issue 5. Included for alignment with the POSIX Realtime Extension.

7305 Issue 6

7306 The <aio.h> header is marked as part of the Asynchronous Input and Output option.

7307 The description of the constants is expanded.

7308 The **restrict** keyword is added to the prototype for *lio_listio()*.

7309 Issue 7

7310 The <aio.h> header is moved from the Asynchronous Input and Output option to the Base.

7311 This reference page is clarified with respect to macros and symbolic constants, and type and
 7312 structure declarations are added.

7313 POSIX.1-2008, Technical Corrigendum 1, XBD/TC1-2008/0038 [98] is applied.

+

NAME

arpa/inet.h — definitions for internet operations

SYNOPSIS

```
#include <arpa/inet.h>
```

DESCRIPTION

The <arpa/inet.h> header shall define the **in_port_t** and **in_addr_t** types as described in <netinet/in.h>.

The <arpa/inet.h> header shall define the **in_addr** structure as described in <netinet/in.h>.

IP6 The <arpa/inet.h> header shall define the **INET_ADDRSTRLEN** and **INET6_ADDRSTRLEN** macros as described in <netinet/in.h>.

The following shall be declared as functions, or defined as macros, or both. If functions are declared, function prototypes shall be provided.

```
uint32_t htonl(uint32_t);
uint16_t htons(uint16_t);
uint32_t ntohl(uint32_t);
uint16_t ntohs(uint16_t);
```

The <arpa/inet.h> header shall define the **uint32_t** and **uint16_t** types as described in <inttypes.h>.

The following shall be declared as functions and may also be defined as macros. Function prototypes shall be provided.

```
in_addr_t    inet_addr(const char *);
char         *inet_ntoa(struct in_addr);
const char   *inet_ntop(int, const void *restrict, char *restrict,
                        socklen_t);
int          inet_pton(int, const char *restrict, void *restrict);
```

Inclusion of the <arpa/inet.h> header may also make visible all symbols from <netinet/in.h> and <inttypes.h>.

APPLICATION USAGE

None.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

<inttypes.h>, <netinet/in.h>

XSH *htonl()*, *inet_addr()*, *inet_ntop()*

CHANGE HISTORY

First released in Issue 6. Derived from the XNS, Issue 5.2 specification.

The **restrict** keyword is added to the prototypes for *inet_ntop()* and *inet_pton()*.

Issue 7

SD5-XBD-ERN-6 is applied.

NAME

assert.h — verify program assertion

SYNOPSIS

```
#include <assert.h>
```

DESCRIPTION

CX The functionality described on this reference page is aligned with the ISO C standard. Any conflict between the requirements described here and the ISO C standard is unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

The <assert.h> header shall define the *assert()* macro. It refers to the macro NDEBUG which is not defined in the header. If NDEBUG is defined as a macro name before the inclusion of this header, the *assert()* macro shall be defined simply as:

```
#define assert(ignore) ((void) 0)
```

Otherwise, the macro behaves as described in *assert()*.

The *assert()* macro shall be redefined according to the current state of NDEBUG each time <assert.h> is included.

The *assert()* macro shall be implemented as a macro, not as a function. If the macro definition is suppressed in order to access an actual function, the behavior is undefined.

APPLICATION USAGE

None.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

XSH *assert()*

CHANGE HISTORY

First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6

The definition of the *assert()* macro is changed for alignment with the ISO/IEC 9899:1999 standard.

7385 NAME

7386 complex.h — complex arithmetic

7387 SYNOPSIS

7388 #include <complex.h>

7389 DESCRIPTION

7390 CX The functionality described on this reference page is aligned with the ISO C standard. Any
 7391 conflict between the requirements described here and the ISO C standard is unintentional. This
 7392 volume of POSIX.1-2008 defers to the ISO C standard.

7393 The <complex.h> header shall define the following macros:

7394 complex Expands to **_Complex**.

7395 **_Complex_I** Expands to a constant expression of type **const float _Complex**, with the value
 7396 of the imaginary unit (that is, a number i such that $i^2 = -1$).

7397 imaginary Expands to **_Imaginary**.

7398 **_Imaginary_I** Expands to a constant expression of type **const float _Imaginary** with the
 7399 value of the imaginary unit.

7400 **I** Expands to either **_Imaginary_I** or **_Complex_I**. If **_Imaginary_I** is not defined,
 7401 **I** expands to **_Complex_I**.

7402 The macros **imaginary** and **_Imaginary_I** shall be defined if and only if the implementation
 7403 supports imaginary types.

7404 An application may undefine and then, perhaps, redefine the **complex**, **imaginary**, and **I** macros.

7405 The following shall be declared as functions and may also be defined as macros. Function
 7406 prototypes shall be provided.

```

7407 double      cabs(double complex);
7408 float       cabsf(float complex);
7409 long double cabsl(long double complex);
7410 double complex cacos(double complex);
7411 float complex cacosf(float complex);
7412 double complex cacosh(double complex);
7413 float complex cacoshf(float complex);
7414 long double complex cacoshl(long double complex);
7415 long double complex cacosl(long double complex);
7416 double      carg(double complex);
7417 float       cargf(float complex);
7418 long double cargl(long double complex);
7419 double complex casin(double complex);
7420 float complex casinf(float complex);
7421 double complex casinh(double complex);
7422 float complex casinhf(float complex);
7423 long double complex casinhl(long double complex);
7424 long double complex casinl(long double complex);
7425 double complex catan(double complex);
7426 float complex catanf(float complex);
7427 double complex catanh(double complex);
7428 float complex catanhf(float complex);
7429 long double complex catanhl(long double complex);
7430 long double complex catanl(long double complex);

```

```

7431     double complex      ccos(double complex);
7432     float complex       ccosf(float complex);
7433     double complex      ccosh(double complex);
7434     float complex       ccoshf(float complex);
7435     long double complex ccoshl(long double complex);
7436     long double complex ccosl(long double complex);
7437     double complex      cexp(double complex);
7438     float complex       cexpf(float complex);
7439     long double complex cexpl(long double complex);
7440     double              cimag(double complex);
7441     float               cimagf(float complex);
7442     long double         cimagl(long double complex);
7443     double complex      clog(double complex);
7444     float complex       clogf(float complex);
7445     long double complex clogl(long double complex);
7446     double complex      conj(double complex);
7447     float complex       conjf(float complex);
7448     long double complex conjl(long double complex);
7449     double complex      cpow(double complex, double complex);
7450     float complex       cpowf(float complex, float complex);
7451     long double complex cpowl(long double complex, long double complex);
7452     double complex      cproj(double complex);
7453     float complex       cprojf(float complex);
7454     long double complex cprojl(long double complex);
7455     double              creal(double complex);
7456     float               crealf(float complex);
7457     long double         creall(long double complex);
7458     double complex      csin(double complex);
7459     float complex       csinf(float complex);
7460     double complex      csinh(double complex);
7461     float complex       csinhf(float complex);
7462     long double complex csinhl(long double complex);
7463     long double complex csinl(long double complex);
7464     double complex      csqrt(double complex);
7465     float complex       csqrtf(float complex);
7466     long double complex csqrtl(long double complex);
7467     double complex      ctan(double complex);
7468     float complex       ctanf(float complex);
7469     double complex      ctanh(double complex);
7470     float complex       ctanhf(float complex);
7471     long double complex ctanhl(long double complex);
7472     long double complex ctanl(long double complex);

```

APPLICATION USAGE

Values are interpreted as radians, not degrees.

RATIONALE

The choice of *I* instead of *i* for the imaginary unit concedes to the widespread use of the identifier *i* for other purposes. The application can use a different identifier, say *j*, for the imaginary unit by following the inclusion of the <complex.h> header with:

```
#undef I
#define j _Imaginary_I
```

An *I* suffix to designate imaginary constants is not required, as multiplication by *I* provides a sufficiently convenient and more generally useful notation for imaginary terms. The corresponding real type for the imaginary unit is **float**, so that use of *I* for algorithmic or notational convenience will not result in widening types.

On systems with imaginary types, the application has the ability to control whether use of the macro *I* introduces an imaginary type, by explicitly defining *I* to be `_Imaginary_I` or `_Complex_I`. Disallowing imaginary types is useful for some applications intended to run on implementations without support for such types.

The macro `_Imaginary_I` provides a test for whether imaginary types are supported.

The `cis()` function ($\cos(x) + I\sin(x)$) was considered but rejected because its implementation is easy and straightforward, even though some implementations could compute sine and cosine more efficiently in tandem.

FUTURE DIRECTIONS

The following function names and the same names suffixed with *f* or *l* are reserved for future use, and may be added to the declarations in the <complex.h> header.

```
cerf()      cexpm1()   clog2()
cerfc()     clog10()  clgamma()
cexp2()     clog1p()  ctgamma()
```

SEE ALSO

XSH `cabs()`, `cacos()`, `cacosh()`, `carg()`, `casin()`, `casinh()`, `catan()`, `catanh()`, `ccos()`, `ccosh()`, `cexp()`, `cimag()`, `clog()`, `conj()`, `cpow()`, `cproj()`, `creal()`, `csin()`, `csinh()`, `csqrt()`, `ctan()`, `ctanh()`

CHANGE HISTORY

First released in Issue 6. Included for alignment with the ISO/IEC 9899:1999 standard.

NAME

cpio.h — cpio archive values

SYNOPSIS

```
#include <cpio.h>
```

DESCRIPTION

The <cpio.h> header shall define the symbolic constants needed by the *c_mode* field of the *cpio* archive format, with the names and values given in the following table:

Name	Description	Value (Octal)
C_IRUSR	Read by owner.	0000400
C_IWUSR	Write by owner.	0000200
C_IXUSR	Execute by owner.	0000100
C_IRGRP	Read by group.	0000040
C_IWGRP	Write by group.	0000020
C_IXGRP	Execute by group.	0000010
C_IROTH	Read by others.	0000004
C_IWOTH	Write by others.	0000002
C_IXOTH	Execute by others.	0000001
C_ISUID	Set user ID.	0004000
C_ISGID	Set group ID.	0002000
C_ISVTX	On directories, restricted deletion flag.	0001000
C_ISDIR	Directory.	0040000
C_ISFIFO	FIFO.	0010000
C_ISREG	Regular file.	0100000
C_ISBLK	Block special.	0060000
C_ISCHR	Character special.	0020000
C_ISCTG	Reserved.	0110000
C_ISLNK	Symbolic link.	0120000
C_ISSOCK	Socket.	0140000

The <cpio.h> header shall define the following symbolic constant as a string:

```
MAGIC      "070707"
```

APPLICATION USAGE

None.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

XCU *pax*

CHANGE HISTORY

First released in the Headers Interface, Issue 3 specification. Derived from the POSIX.1-1988 standard.

Issue 6

The SEE ALSO is updated to refer to *pax*.

Issue 7

7547 The **<pio.h>** header is moved from the XSI option to the Base.

7548 This reference page is clarified with respect to macros and symbolic constants.

7550 **NAME**

7551 ctype.h — character types

7552 **SYNOPSIS**

7553 #include <ctype.h>

7554 **DESCRIPTION**

7555 CX Some of the functionality described on this reference page extends the ISO C standard.
7556 Applications shall define the appropriate feature test macro (see XSH [Section 2.2](#), on page 472) to
7557 enable the visibility of these symbols in this header.

7558 The <ctype.h> header shall define the **locale_t** type as described in <locale.h>, representing a
7559 locale object.

7560 The following shall be declared as functions and may also be defined as macros. Function
7561 prototypes shall be provided for use with ISO C standard compilers.

```
7562 int isalnum(int);
7563 CX int isalnum_l(int, locale_t);
7564 int isalpha(int);
7565 CX int isalpha_l(int, locale_t);
7566 OB XSI int isascii(int);
7567 int isblank(int);
7568 CX int isblank_l(int, locale_t);
7569 int iscntrl(int);
7570 CX int iscntrl_l(int, locale_t);
7571 int isdigit(int);
7572 CX int isdigit_l(int, locale_t);
7573 int isgraph(int);
7574 CX int isgraph_l(int, locale_t);
7575 int islower(int);
7576 CX int islower_l(int, locale_t);
7577 int isprint(int);
7578 CX int isprint_l(int, locale_t);
7579 int ispunct(int);
7580 CX int ispunct_l(int, locale_t);
7581 int isspace(int);
7582 CX int isspace_l(int, locale_t);
7583 int isupper(int);
7584 CX int isupper_l(int, locale_t);
7585 int isxdigit(int);
7586 CX int isxdigit_l(int, locale_t);
7587 OB XSI int toascii(int);
7588 int tolower(int);
7589 CX int tolower_l(int, locale_t);
7590 int toupper(int);
7591 CX int toupper_l(int, locale_t);
```

7592 The <ctype.h> header shall define the following as macros:

```
7593 OB XSI int _toupper(int);
7594 int _tolower(int);
```

7595 APPLICATION USAGE

7596 None.

7597 RATIONALE

7598 None.

7599 FUTURE DIRECTIONS

7600 None.

7601 SEE ALSO

7602 **<locale.h>**

7603 XSH Section 2.2 (on page 472), *isalnum()*, *isalpha()*, *isascii()*, *isblank()*, *isctrl()*, *isdigit()*,
7604 *isgraph()*, *islower()*, *isprint()*, *ispunct()*, *isspace()*, *isupper()*, *isxdigit()*, *mblen()*, *mbstowcs()*,
7605 *mbtowc()*, *setlocale()*, *toascii()*, *tolower()*, *_tolower()*, *toupper()*, *_toupper()*, *wcstombs()*, *wctomb()*

7606 CHANGE HISTORY

7607 First released in Issue 1. Derived from Issue 1 of the SVID.

7608 Issue 6

7609 Extensions beyond the ISO C standard are marked.

7610 Issue 7

7611 SD5-XBD-ERN-6 is applied, updating the wording regarding the function declarations for
7612 consistency.

7613 The *_l() functions are added from The Open Group Technical Standard, 2006, Extended API Set
7614 Part 4.

NAME

dirent.h — format of directory entries

SYNOPSIS

```
#include <dirent.h>
```

DESCRIPTION

The internal format of directories is unspecified.

The <dirent.h> header shall define the following type:

DIR A type representing a directory stream. The **DIR** type may be an incomplete type.

It shall also define the structure **dirent** which shall include the following members:

XSI	<code>ino_t</code>	<code>d_ino</code>	File serial number.
	<code>char</code>	<code>d_name[]</code>	Filename string of entry.

The <dirent.h> header shall define the `ino_t` type as described in <sys/types.h>.

The array `d_name` is of unspecified size, but shall contain a filename of at most {NAME_MAX} + bytes followed by a terminating null byte.

The following shall be declared as functions and may also be defined as macros. Function prototypes shall be provided.

```
int      alphasort(const struct dirent **, const struct dirent **);
int      closedir(DIR *);
int      dirfd(DIR *);
DIR      *fdopendir(int);
DIR      *opendir(const char *);
struct dirent *readdir(DIR *);
int      readdir_r(DIR *restrict, struct dirent *restrict,
                struct dirent **restrict);
void      rewinddir(DIR *);
int      scandir(const char *, struct dirent ***,
                int (*)(const struct dirent *),
                int (*)(const struct dirent **),
                const struct dirent **));
XSI      void      seekdir(DIR *, long);
long      telldir(DIR *);
```

APPLICATION USAGE

None.

RATIONALE

Information similar to that in the <dirent.h> header is contained in a file <sys/dir.h> in 4.2 BSD - and 4.3 BSD. The equivalent in these implementations of **struct dirent** from this volume of POSIX.1-2008 is **struct direct**. The filename was changed because the name <sys/dir.h> was also used in earlier implementations to refer to definitions related to the older access method; this produced name conflicts. The name of the structure was changed because this volume of POSIX.1-2008 does not completely define what is in the structure, so it could be different on some implementations from **struct direct**.

The name of an array of **char** of an unspecified size should not be used as an lvalue. Use of:

```
sizeof(d_name)
```

is incorrect; use:

7659 `strlen(d_name)`

7660 instead.

7661 The array of **char** *d_name* is not a fixed size. Implementations may need to declare **struct dirent**
 7662 with an array size for *d_name* of 1, but the actual number of bytes provided matches (or only
 7663 slightly exceeds) the length of the filename string.

7664 **FUTURE DIRECTIONS**

7665 None.

7666 **SEE ALSO**

7667 [**<sys/types.h>**](#)

7668 XSH [*alphasort\(\)*](#), [*closedir\(\)*](#), [*dirfd\(\)*](#), [*fdopendir\(\)*](#), [*readdir\(\)*](#), [*rewinddir\(\)*](#), [*seekdir\(\)*](#), [*telldir\(\)*](#)

7669 **CHANGE HISTORY**

7670 First released in Issue 2.

7671 **Issue 5**

7672 The DESCRIPTION is updated for alignment with the POSIX Threads Extension.

7673 **Issue 6**

7674 The Open Group Corrigendum U026/7 is applied, correcting the prototype for *readdir_r()*.

7675 The **restrict** keyword is added to the prototype for *readdir_r()*.

7676 **Issue 7**

7677 The *alphasort()*, *dirfd()*, and *scandir()* functions are added from The Open Group Technical
 7678 Standard, 2006, Extended API Set Part 1.

7679 The *fdopendir()* function is added from The Open Group Technical Standard, 2006, Extended API
 7680 Set Part 2.

7681 Austin Group Interpretation 1003.1-2001 #110 is applied, clarifying the definition of the **DIR**
 7682 type.

7683 POSIX.1-2008, Technical Corrigendum 1, XBD/TC1-2008/0039 [291], XBD/TC1-2008/0040 [291], +
 7684 XBD/TC1-2008/0041 [291], and XBD/TC1-2008/0042 [206] are applied.

7685 **NAME**

7686 dlfcn.h — dynamic linking

7687 **SYNOPSIS**

7688 #include <dlfcn.h>

7689 **DESCRIPTION**

7690 The <dlfcn.h> header shall define at least the following symbolic constants for use in the
7691 construction of a *dlopen()* *mode* argument:

7692	RTLD_LAZY	Relocations are performed at an implementation-defined time.
7693	RTLD_NOW	Relocations are performed when the object is loaded.
7694	RTLD_GLOBAL	All symbols are available for relocation processing of other modules.
7695	RTLD_LOCAL	All symbols are not made available for relocation processing by other
7696		modules.

7697 The following shall be declared as functions and may also be defined as macros. Function
7698 prototypes shall be provided.

```
7699         int     dlclose(void *);
7700         char    *dlerror(void);
7701         void    *dlopen(const char *, int);
7702         void    *dlsym(void *restrict, const char *restrict);
```

7703 **APPLICATION USAGE**

7704 None.

7705 **RATIONALE**

7706 None.

7707 **FUTURE DIRECTIONS**

7708 None.

7709 **SEE ALSO**

7710 XSH *dlclose()*, *dlerror()*, *dlopen()*, *dlsym()*

7711 **CHANGE HISTORY**

7712 First released in Issue 5.

7713 **Issue 6**

7714 The **restrict** keyword is added to the prototype for *dlsym()*.

7715 **Issue 7**

7716 The <dlfcn.h> header is moved from the XSI option to the Base.

7717 This reference page is clarified with respect to macros and symbolic constants.

7718 **NAME**

7719 errno.h — system error numbers

7720 **SYNOPSIS**

7721 #include <errno.h>

7722 **DESCRIPTION**

7723 CX Some of the functionality described on this reference page extends the ISO C standard. Any
 7724 conflict between the requirements described here and the ISO C standard is unintentional. This
 7725 volume of POSIX.1-2008 defers to the ISO C standard.

7726 The ISO C standard only requires the symbols [EDOM], [EILSEQ], and [ERANGE] to be defined.

7727 The <errno.h> header shall provide a declaration or definition for *errno*. The symbol *errno* shall
 7728 expand to a modifiable lvalue of type **int**. It is unspecified whether *errno* is a macro or an
 7729 identifier declared with external linkage. If a macro definition is suppressed in order to access an
 7730 actual object, or a program defines an identifier with the name *errno*, the behavior is undefined.

7731 The <errno.h> header shall define the following macros which shall expand to integer constant
 7732 expressions with type **int**, distinct positive values (except as noted below), and which shall be
 7733 suitable for use in **#if** preprocessing directives:

7734	[E2BIG]	Argument list too long.
7735	[EACCES]	Permission denied.
7736	[EADDRINUSE]	Address in use.
7737	[EADDRNOTAVAIL]	Address not available.
7738	[EAFNOSUPPORT]	Address family not supported.
7739	[EAGAIN]	Resource unavailable, try again (may be the same value as
7740		[EWOULDBLOCK]).
7741	[EALREADY]	Connection already in progress.
7742	[EBADF]	Bad file descriptor.
7743	[EBADMSG]	Bad message.
7744	[EBUSY]	Device or resource busy.
7745	[ECANCELED]	Operation canceled.
7746	[ECHILD]	No child processes.
7747	[ECONNABORTED]	Connection aborted.
7748	[ECONNREFUSED]	Connection refused.
7749	[ECONNRESET]	Connection reset.
7750	[EDEADLK]	Resource deadlock would occur.
7751	[EDESTADDRREQ]	Destination address required.
7752	[EDOM]	Mathematics argument out of domain of function.
7753	[EDQUOT]	Reserved.
7754	[EEXIST]	File exists.

7755		[EFAULT]	Bad address.
7756		[EFBIG]	File too large.
7757		[EHOSTUNREACH]	Host is unreachable.
7758		[EIDRM]	Identifier removed.
7759		[EILSEQ]	Illegal byte sequence.
7760		[EINPROGRESS]	Operation in progress.
7761		[EINTR]	Interrupted function.
7762		[EINVAL]	Invalid argument.
7763		[EIO]	I/O error.
7764		[EISCONN]	Socket is connected.
7765		[EISDIR]	Is a directory.
7766		[ELOOP]	Too many levels of symbolic links.
7767		[EMFILE]	File descriptor value too large.
7768		[EMLINK]	Too many links.
7769		[EMSGSIZE]	Message too large.
7770		[EMULTIHOP]	Reserved.
7771		[ENAMETOOLONG]	Filename too long.
7772		[ENETDOWN]	Network is down.
7773		[ENETRESET]	Connection aborted by network.
7774		[ENETUNREACH]	Network unreachable.
7775		[ENFILE]	Too many files open in system.
7776		[ENOBUFS]	No buffer space available.
7777	OB XSR	[ENODATA]	No message is available on the STREAM head read queue.
7778		[ENODEV]	No such device.
7779		[ENOENT]	No such file or directory.
7780		[ENOEXEC]	Executable file format error.
7781		[ENOLCK]	No locks available.
7782		[ENOLINK]	Reserved.
7783		[ENOMEM]	Not enough space.
7784		[ENOMSG]	No message of the desired type.
7785		[ENOPROTOOPT]	Protocol not available.
7786		[ENOSPC]	No space left on device.
7787	OB XSR	[ENOSR]	No STREAM resources.

7788	OB XSR	[ENOSTR]	Not a STREAM.	
7789		[ENOSYS]	Function not supported.	
7790		[ENOTCONN]	The socket is not connected.	
7791		[ENOTDIR]	Not a directory or a symbolic link to a directory.	
7792		[ENOTEMPTY]	Directory not empty.	
7793		[ENOTRECOVERABLE]		
7794			State not recoverable.	
7795		[ENOTSOCK]	Not a socket.	
7796		[ENOTSUP]	Not supported (may be the same value as [EOPNOTSUPP]).	
7797		[ENOTTY]	Inappropriate I/O control operation.	
7798		[ENXIO]	No such device or address.	
7799		[EOPNOTSUPP]	Operation not supported on socket (may be the same value as	
7800			[ENOTSUP]).	
7801		[EOVERFLOW]	Value too large to be stored in data type.	
7802		[EOWNERDEAD]	Previous owner died.	
7803		[EPERM]	Operation not permitted.	
7804		[EPIPE]	Broken pipe.	
7805		[EPROTO]	Protocol error.	
7806		[EPROTONOSUPPORT]		
7807			Protocol not supported.	
7808		[EPROTOTYPE]	Protocol wrong type for socket.	
7809		[ERANGE]	Result too large.	
7810		[EROFS]	Read-only file system.	
7811		[ESPIPE]	Invalid seek.	
7812		[ESRCH]	No such process.	
7813		[ESTALE]	Reserved.	
7814	OB XSR	[ETIME]	Stream <i>ioctl()</i> timeout.	
7815		[ETIMEDOUT]	Connection timed out.	
7816		[ETXTBSY]	Text file busy.	
7817		[EWOULDBLOCK]	Operation would block (may be the same value as [EAGAIN]).	
7818		[EXDEV]	Cross-device link.	

APPLICATION USAGE

Additional error numbers may be defined on conforming systems; see the System Interfaces volume of POSIX.1-2008.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

XSH [Section 2.3](#) (on page 481)

CHANGE HISTORY

First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 5

Updated for alignment with the POSIX Realtime Extension.

Issue 6

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The majority of the error conditions previously marked as extensions are now mandatory, except for the STREAMS-related error conditions.

Values for *errno* are now required to be distinct positive values rather than non-zero values. This change is for alignment with the ISO/IEC 9899:1999 standard.

Issue 7

Austin Group Interpretation 1003.1-2001 #050 is applied, allowing [ENOTSUP] and [EOPNOTSUPP] to be the same values.

The [ENOTRECOVERABLE] and [EOWNERDEAD] errors are added from The Open Group Technical Standard, 2006, Extended API Set Part 2.

Functionality relating to the XSI STREAMS option is marked obsolescent.

Functionality relating to the Threads option is moved to the Base.

This reference page is clarified with respect to macros and symbolic constants.

POSIX.1-2008, Technical Corrigendum 1, XBD/TC1-2008/0043 [324] is applied.

+

NAME

fcntl.h — file control options

SYNOPSIS

#include <fcntl.h>

DESCRIPTION

The <fcntl.h> header shall define the following symbolic constants for the *cmd* argument used by *fcntl()*. The values shall be unique and shall be suitable for use in **#if** preprocessing directives.

F_DUPFD Duplicate file descriptor.

F_DUPFD_CLOEXEC

Duplicate file descriptor with the close-on-exec flag **FD_CLOEXEC** set.

F_GETFD Get file descriptor flags.

F_SETFD Set file descriptor flags.

F_GETFL Get file status flags and file access modes.

F_SETFL Set file status flags.

F_GETLK Get record locking information.

F_SETLK Set record locking information.

F_SETLKW Set record locking information; wait if blocked.

F_GETOWN Get process or process group ID to receive SIGURG signals.

F_SETOWN Set process or process group ID to receive SIGURG signals.

The <fcntl.h> header shall define the following symbolic constant used for the *fcntl()* file descriptor flags, which shall be suitable for use in **#if** preprocessing directives.

FD_CLOEXEC Close the file descriptor upon execution of an *exec* family function.

The <fcntl.h> header shall also define the following symbolic constants for the *l_type* argument used for record locking with *fcntl()*. The values shall be unique and shall be suitable for use in **#if** preprocessing directives.

F_RDLCK Shared or read lock.

F_UNLCK Unlock.

F_WRLCK Exclusive or write lock.

The <fcntl.h> header shall define the values used for *l_whence*, **SEEK_SET**, **SEEK_CUR**, and **SEEK_END** as described in <stdio.h>.

The <fcntl.h> header shall define the following symbolic constants as file creation flags for use in the *oflag* value to *open()* and *openat()*. The values shall be bitwise-distinct and shall be suitable for use in **#if** preprocessing directives.

O_CLOEXEC The **FD_CLOEXEC** flag associated with the new descriptor shall be set to close +
the file descriptor upon execution of an *exec* family function.

O_CREAT Create file if it does not exist.

O_DIRECTORY Fail if not a directory. +

7886		O_EXCL	Exclusive use flag.	
7887		O_NOCTTY	Do not assign controlling terminal.	
7888		O_NOFOLLOW	Do not follow symbolic links.	+
7889		O_TRUNC	Truncate flag.	
7890		O_TTY_INIT	Set the termios structure terminal parameters to a state that provides conforming behavior; see Section 11.2 (on page 205).	
7892		The O_TTY_INIT flag can have the value zero and in this case it need not be bitwise-distinct from the other flags.		
7894		The <fcntl.h> header shall define the following symbolic constants for use as file status flags for <i>open()</i> , <i>openat()</i> , and <i>fcntl()</i> . The values shall be suitable for use in #if preprocessing directives.		
7896		O_APPEND	Set append mode.	
7897	SIO	O_DSYNC	Write according to synchronized I/O data integrity completion.	
7898		O_NONBLOCK	Non-blocking mode.	
7899	SIO	O_RSYNC	Synchronized read I/O operations.	
7900		O_SYNC	Write according to synchronized I/O file integrity completion.	
7901		The <fcntl.h> header shall define the following symbolic constant for use as the mask for file access modes. The value shall be suitable for use in #if preprocessing directives.		
7903		O_ACCMODE	Mask for file access modes.	
7904		The <fcntl.h> header shall define the following symbolic constants for use as the file access modes for <i>open()</i> , <i>openat()</i> , and <i>fcntl()</i> . The values shall be unique, except that O_EXEC and O_SEARCH may have equal values. The values shall be suitable for use in #if preprocessing directives.		
7908		O_EXEC	Open for execute only (non-directory files). The result is unspecified if this flag is applied to a directory.	
7910		O_RDONLY	Open for reading only.	
7911		O_RDWR	Open for reading and writing.	
7912		O_SEARCH	Open directory for search only. The result is unspecified if this flag is applied to a non-directory file.	
7914		O_WRONLY	Open for writing only.	
7915		The <fcntl.h> header shall define the symbolic constants for file modes for use as values of mode_t as described in <sys/stat.h>.		
7917		The <fcntl.h> header shall define the following symbolic constant as a special value used in place of a file descriptor for the <i>*at()</i> functions which take a directory file descriptor as a parameter:		
7920		AT_FDCWD	Use the current working directory to determine the target of relative file paths.	
7921		The <fcntl.h> header shall define the following symbolic constant as a value for the <i>flag</i> used by <i>faccessat()</i> :		
7923		AT_EACCESS	Check access using effective user and group ID.	
7924		The <fcntl.h> header shall define the following symbolic constant as a value for the <i>flag</i> used by		

7925 *fstatat()*, *fchmodat()*, *fchownat()*, and *utimensat()*:

7926 AT_SYMLINK_NOFOLLOW

7927 Do not follow symbolic links.

7928 The <fcntl.h> header shall define the following symbolic constant as a value for the flag used by *linkat()*:

7930 AT_SYMLINK_FOLLOW

7931 Follow symbolic link.

7932 The <fcntl.h> header shall define the following symbolic constant as a value for the flag used by *unlinkat()*:

7934 AT_REMOVEDIR

7935 Remove directory instead of file.

7936 ADV The <fcntl.h> header shall define the following symbolic constants for the *advice* argument used by *posix_fadvise()*:

7938 POSIX_FADV_DONTNEED

7939 The application expects that it will not access the specified data in the near future.

7940 POSIX_FADV_NOREUSE

7941 The application expects to access the specified data once and then not reuse it thereafter.

7942 POSIX_FADV_NORMAL

7943 The application has no advice to give on its behavior with respect to the specified data. It is the default characteristic if no advice is given for an open file.

7945 POSIX_FADV_RANDOM

7946 The application expects to access the specified data in a random order.

7947 POSIX_FADV_SEQUENTIAL

7948 The application expects to access the specified data sequentially from lower offsets to higher offsets.

7950 POSIX_FADV_WILLNEED

7951 The application expects to access the specified data in the near future.

7952 The <fcntl.h> header shall define the **flock** structure describing a file lock. It shall include the following members:

7954 short l_type Type of lock; F_RDLCK, F_WRLCK, F_UNLCK.

7955 short l_whence Flag for starting offset.

7956 off_t l_start Relative offset in bytes.

7957 off_t l_len Size; if 0 then until EOF.

7958 pid_t l_pid Process ID of the process holding the lock; returned with F_GETLK.

7959 The <fcntl.h> header shall define the **mode_t**, **off_t**, and **pid_t** types as described in <sys/types.h>.

7961 The following shall be declared as functions and may also be defined as macros. Function prototypes shall be provided.

7963 int creat(const char *, mode_t);

7964 int fcntl(int, int, ...);

7965 int open(const char *, int, ...);

7966 int openat(int, const char *, int, ...);

```
7967 ADV int posix_fadvise(int, off_t, off_t, int);
7968 int posix_fallocate(int, off_t, off_t);
```

7969 Inclusion of the <fcntl.h> header may also make visible all symbols from <sys/stat.h> and
7970 <unistd.h>.

7971 APPLICATION USAGE

7972 Although no existing implementation defines AT_SYMLINK_FOLLOW and
7973 AT_SYMLINK_NOFOLLOW as the same numeric value, POSIX.1-2008 does not prohibit that as
7974 the two constants are not used with the same interfaces.

7975 RATIONALE

7976 While many of the symbolic constants introduced in the <fcntl.h> header do not strictly need to
7977 be used in #if preprocessor directives, widespread historic practice has defined them as macros
7978 that are usable in such constructs, and examination of existing applications has shown that they
7979 are occasionally used in such a way. Therefore it was decided to retain this requirement on an
7980 implementation in POSIX.1-2008.

7981 FUTURE DIRECTIONS

7982 None.

7983 SEE ALSO

7984 [<stdio.h>](#), [<sys/stat.h>](#), [<sys/types.h>](#), [<unistd.h>](#)

7985 XSH [creat\(\)](#), [exec](#), [fcntl\(\)](#), [futimens\(\)](#), [open\(\)](#), [posix_fadvise\(\)](#), [posix_fallocate\(\)](#), [posix_madvise\(\)](#)

7986 CHANGE HISTORY

7987 First released in Issue 1. Derived from Issue 1 of the SVID.

7988 Issue 5

7989 The DESCRIPTION is updated for alignment with the POSIX Realtime Extension.

7990 Issue 6

7991 The following changes are made for alignment with the ISO POSIX-1: 1996 standard:

- 7992 • O_DSYNC and O_RSYNC are marked as part of the Synchronized Input and Output
7993 option.

7994 The following new requirements on POSIX implementations derive from alignment with the
7995 Single UNIX Specification:

- 7996 • The definition of the **mode_t**, **off_t**, and **pid_t** types is mandated.

7997 The F_GETOWN and F_SETOWN values are added for sockets.

7998 The [posix_fadvise\(\)](#), [posix_fallocate\(\)](#), and [posix_madvise\(\)](#) functions are added for alignment with
7999 IEEE Std 1003.1d-1999.

8000 IEEE PASC Interpretation 1003.1 #102 is applied, moving the prototype for [posix_madvise\(\)](#) to
8001 [<sys/mman.h>](#).

8002 IEEE Std 1003.1-2001/Cor 2-2004, item XBD/TC2/D6/18 is applied, updating the prototypes for
8003 [posix_fadvise\(\)](#) and [posix_fallocate\(\)](#) to be large file-aware, using **off_t** instead of **size_t**.

8004 Issue 7

8005 Austin Group Interpretation 1003.1-2001 #144 is applied, adding the O_TTY_INIT flag.

8006 Austin Group Interpretation 1003.1-2001 #171 is applied, adding support to set the
8007 FD_CLOEXEC flag atomically at [open\(\)](#), and adding the F_DUPFD_CLOEXEC flag.

8008 The *openat()* function is added from The Open Group Technical Standard, 2006, Extended API
8009 Set Part 2.

8010 Additional flags are added to support *faccessat()*, *fchmodat()*, *fchownat()*, *fstatat()*, *linkat()*,
8011 *open()*, *openat()*, and *unlinkat()*.

8012 This reference page is clarified with respect to macros and symbolic constants.

8013 Changes are made related to support for finegrained timestamps.

8014 Changes are made to allow a directory to be opened for searching.

8015 POSIX.1-2008, Technical Corrigendum 1, XBD/TC1-2008/0044 [274] and XBD/TC1-2008/0045 +
8016 [78,432] are applied.

8017 **NAME**

8018 fenv.h — floating-point environment

8019 **SYNOPSIS**

8020 #include <fenv.h>

8021 **DESCRIPTION**

8022 CX The functionality described on this reference page is aligned with the ISO C standard. Any
8023 conflict between the requirements described here and the ISO C standard is unintentional. This
8024 volume of POSIX.1-2008 defers to the ISO C standard.

8025 The <fenv.h> header shall define the following data types through **typedef**:

8026 **fenv_t** Represents the entire floating-point environment. The floating-point environment
8027 refers collectively to any floating-point status flags and control modes supported
8028 by the implementation.

8029 **fexcept_t** Represents the floating-point status flags collectively, including any status the
8030 implementation associates with the flags. A floating-point status flag is a system
8031 variable whose value is set (but never cleared) when a floating-point exception is
8032 raised, which occurs as a side-effect of exceptional floating-point arithmetic to
8033 provide auxiliary information. A floating-point control mode is a system variable
8034 whose value may be set by the user to affect the subsequent behavior of floating-
8035 point arithmetic.

8036 The <fenv.h> header shall define each of the following macros if and only if the implementation
8037 supports the floating-point exception by means of the floating-point functions *feclearexcept()*,
8038 *fegetexceptflag()*, *feraiseexcept()*, *fesetexceptflag()*, and *fetestexcept()*. The defined macros shall
8039 expand to integer constant expressions with values that are bitwise-distinct.

8040 FE_DIVBYZERO
8041 FE_INEXACT
8042 FE_INVALID
8043 FE_OVERFLOW
8044 FE_UNDERFLOW

8045 MX If the implementation supports the IEC 60559 Floating-Point option, all five macros shall be
8046 defined. Additional implementation-defined floating-point exceptions with macros beginning
8047 with FE_ and an uppercase letter may also be specified by the implementation.

8048 The <fenv.h> header shall define the macro FE_ALL_EXCEPT as the bitwise-inclusive OR of all
8049 floating-point exception macros defined by the implementation, if any. If no such macros are
8050 defined, then the macro FE_ALL_EXCEPT shall be defined as zero.

8051 The <fenv.h> header shall define each of the following macros if and only if the implementation
8052 supports getting and setting the represented rounding direction by means of the *fegetround()*
8053 and *fesetround()* functions. The defined macros shall expand to integer constant expressions
8054 whose values are distinct non-negative values.

8055 FE_DOWNWARD
8056 FE_TONEAREST
8057 FE_TOWARDZERO
8058 FE_UPWARD

8059 MX If the implementation supports the IEC 60559 Floating-Point option, all four macros shall be
8060 defined. Additional implementation-defined rounding directions with macros beginning with
8061 FE_ and an uppercase letter may also be specified by the implementation.

The <fenv.h> header shall define the following macro, which represents the default floating-point environment (that is, the one installed at program startup) and has type pointer to const-qualified `fenv_t`. It can be used as an argument to the functions within the <fenv.h> header that manage the floating-point environment.

```
FE_DFL_ENV
```

The following shall be declared as functions and may also be defined as macros. Function prototypes shall be provided.

```
int feclearexcept(int);
int fegetenv(fenv_t *);
int fegetexceptflag(fexcept_t *, int);
int fegetround(void);
int feholdexcept(fenv_t *);
int feraiseexcept(int);
int fesetenv(const fenv_t *);
int fesetexceptflag(const fexcept_t *, int);
int fesetround(int);
int fetestexcept(int);
int feupdateenv(const fenv_t *);
```

The `FENV_ACCESS` pragma provides a means to inform the implementation when an application might access the floating-point environment to test floating-point status flags or run under non-default floating-point control modes. The pragma shall occur either outside external declarations or preceding all explicit declarations and statements inside a compound statement. When outside external declarations, the pragma takes effect from its occurrence until another `FENV_ACCESS` pragma is encountered, or until the end of the translation unit. When inside a compound statement, the pragma takes effect from its occurrence until another `FENV_ACCESS` pragma is encountered (including within a nested compound statement), or until the end of the compound statement; at the end of a compound statement the state for the pragma is restored to its condition just before the compound statement. If this pragma is used in any other context, the behavior is undefined. If part of an application tests floating-point status flags, sets floating-point control modes, or runs under non-default mode settings, but was translated with the state for the `FENV_ACCESS` pragma off, the behavior is undefined. The default state (on or off) for the pragma is implementation-defined. (When execution passes from a part of the application translated with `FENV_ACCESS` off to a part translated with `FENV_ACCESS` on, the state of the floating-point status flags is unspecified and the floating-point control modes have their default settings.)

APPLICATION USAGE

This header is designed to support the floating-point exception status flags and directed-rounding control modes required by the IEC 60559:1989 standard, and other similar floating-point state information. Also it is designed to facilitate code portability among all systems.

Certain application programming conventions support the intended model of use for the floating-point environment:

- A function call does not alter its caller's floating-point control modes, clear its caller's floating-point status flags, nor depend on the state of its caller's floating-point status flags unless the function is so documented.
- A function call is assumed to require default floating-point control modes, unless its documentation promises otherwise.

- A function call is assumed to have the potential for raising floating-point exceptions, unless its documentation promises otherwise.

With these conventions, an application can safely assume default floating-point control modes (or be unaware of them). The responsibilities associated with accessing the floating-point environment fall on the application that does so explicitly.

Even though the rounding direction macros may expand to constants corresponding to the values of FLT_ROUNDS, they are not required to do so.

For example:

```
#include <fenv.h>
void f(double x)
{
    #pragma STDC FENV_ACCESS ON
    void g(double);
    void h(double);
    /* ... */
    g(x + 1);
    h(x + 1);
    /* ... */
}
```

If the function `g()` might depend on status flags set as a side-effect of the first `x+1`, or if the second `x+1` might depend on control modes set as a side-effect of the call to function `g()`, then the application shall contain an appropriately placed invocation as follows:

```
#pragma STDC FENV_ACCESS ON
```

RATIONALE

The `fexcept_t` Type

`fexcept_t` does not have to be an integer type. Its values must be obtained by a call to `fegetexceptflag()`, and cannot be created by logical operations from the exception macros. An implementation might simply implement `fexcept_t` as an `int` and use the representations reflected by the exception macros, but is not required to; other representations might contain extra information about the exceptions. `fexcept_t` might be a `struct` with a member for each exception (that might hold the address of the first or last floating-point instruction that caused that exception). The ISO/IEC 9899:1999 standard makes no claims about the internals of an `fexcept_t`, and so the user cannot inspect it.

Exception and Rounding Macros

Macros corresponding to unsupported modes and rounding directions are not defined by the implementation and must not be defined by the application. An application might use `#ifdef` to test for this.

FUTURE DIRECTIONS

None.

SEE ALSO

XSH [`feclearexcept\(\)`](#), [`fegetenv\(\)`](#), [`fegetexceptflag\(\)`](#), [`fegetround\(\)`](#), [`feholdexcept\(\)`](#), [`feraiseexcept\(\)`](#), [`fetestexcept\(\)`](#), [`feupdateenv\(\)`](#)

CHANGE HISTORY

First released in Issue 6. Included for alignment with the ISO/IEC 9899:1999 standard.

The return types for *feclearexcept()*, *fegetexceptflag()*, *feraiseexcept()*, *fesetexceptflag()*, *fegetenv()*, *fesetenv()*, and *feupdateenv()* are changed from **void** to **int** for alignment with the ISO/IEC 9899:1999 standard, Defect Report 202.

Issue 7

ISO/IEC 9899:1999 standard, Technical Corrigendum 2 #37 (SD5-XBD-ERN-49) is applied.

ISO/IEC 9899:1999 standard, Technical Corrigendum 3 #36 is applied.

SD5-XBD-ERN-48 and SD5-XBD-ERN-69 are applied.

This reference page is clarified with respect to macros and symbolic constants.

NAME

float.h — floating types

SYNOPSIS

```
#include <float.h>
```

DESCRIPTION

CX The functionality described on this reference page is aligned with the ISO C standard. Any conflict between the requirements described here and the ISO C standard is unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

The characteristics of floating types are defined in terms of a model that describes a representation of floating-point numbers and values that provide information about an implementation's floating-point arithmetic.

The following parameters are used to define the model for each floating-point type:

s Sign (± 1).

b Base or radix of exponent representation (an integer > 1).

e Exponent (an integer between a minimum $e_{\min}$ and a maximum $e_{\max}$).

p Precision (the number of base-*b* digits in the significand).

f_k Non-negative integers less than *b* (the significand digits).

A floating-point number *x* is defined by the following model:

$$x = sb^e \sum_{k=1}^p f_k b^{-k}, e_{\min} \leq e \leq e_{\max}$$

In addition to normalized floating-point numbers ($f_1 > 0$ if $x \neq 0$), floating types may be able to contain other kinds of floating-point numbers, such as subnormal floating-point numbers ($x \neq 0$, $e = e_{\min}$, $f_1 = 0$) and unnormalized floating-point numbers ($x \neq 0$, $e > e_{\min}$, $f_1 = 0$), and values that are not floating-point numbers, such as infinities and NaNs. A NaN is an encoding signifying Not-a-Number. A *quiet NaN* propagates through almost every arithmetic operation without raising a floating-point exception; a *signaling NaN* generally raises a floating-point exception when occurring as an arithmetic operand.

An implementation may give zero and non-numeric values, such as infinities and NaNs, a sign, or may leave them unsigned. Wherever such values are unsigned, any requirement in POSIX.1-2008 to retrieve the sign shall produce an unspecified sign and any requirement to set the sign shall be ignored.

The accuracy of the floating-point operations ('+', '-', '*', '/') and of the functions in <math.h> and <complex.h> that return floating-point results is implementation-defined, as is the accuracy of the conversion between floating-point internal representations and string representations performed by the functions in <stdio.h>, <stdlib.h>, and <wchar.h>. The implementation may state that the accuracy is unknown.

All integer values in the <float.h> header, except FLT_ROUNDS, shall be constant expressions suitable for use in #if preprocessing directives; all floating values shall be constant expressions. All except DECIMAL_DIG, FLT_EVAL_METHOD, FLT_RADIX, and FLT_ROUNDS have separate names for all three floating-point types. The floating-point model representation is provided for all values except FLT_EVAL_METHOD and FLT_ROUNDS.

The rounding mode for floating-point addition is characterized by the implementation-defined

8200 value of FLT_ROUNDS:

8201 -1 Indeterminable.

8202 0 Toward zero.

8203 1 To nearest.

8204 2 Toward positive infinity.

8205 3 Toward negative infinity.

8206 All other values for FLT_ROUNDS characterize implementation-defined rounding behavior.

8207 The values of operations with floating operands and values subject to the usual arithmetic

8208 conversions and of floating constants are evaluated to a format whose range and precision may

8209 be greater than required by the type. The use of evaluation formats is characterized by the

8210 implementation-defined value of FLT_EVAL_METHOD:

8211 -1 Indeterminable.

8212 0 Evaluate all operations and constants just to the range and precision of the type.

8213 1 Evaluate operations and constants of type **float** and **double** to the range and precision of

8214 the **double** type; evaluate **long double** operations and constants to the range and precision

8215 of the **long double** type.

8216 2 Evaluate all operations and constants to the range and precision of the **long double** type.

8217 All other negative values for FLT_EVAL_METHOD characterize implementation-defined

8218 behavior.

8219 The <float.h> header shall define the following values as constant expressions with

8220 implementation-defined values that are greater or equal in magnitude (absolute value) to those

8221 shown, with the same sign.

8222 • Radix of exponent representation, b .

8223 FLT_RADIX 2

8224 • Number of base-FLT_RADIX digits in the floating-point significand, p .

8225 FLT_MANT_DIG

8226 DBL_MANT_DIG

8227 LDBL_MANT_DIG

8228 • Number of decimal digits, n , such that any floating-point number in the widest supported

8229 floating type with $p_{\max}$ radix b digits can be rounded to a floating-point number with n

8230 decimal digits and back again without change to the value.

$$\begin{cases} p_{\max} \log_{10} b & \text{if } b \text{ is a power of } 10 \\ \left\lceil 1 + p_{\max} \log_{10} b \right\rceil & \text{otherwise} \end{cases}$$

8231 DECIMAL_DIG 10

- 8232 • Number of decimal digits, q , such that any floating-point number with q decimal digits can
 8233 be rounded into a floating-point number with p radix b digits and back again without
 8234 change to the q decimal digits.

$$\begin{cases} p \log_{10} b & \text{if } b \text{ is a power of } 10 \\ \left\lceil (p-1) \log_{10} b \right\rceil & \text{otherwise} \end{cases}$$

8235 FLT_DIG 6

8236 DBL_DIG 10

8237 LDBL_DIG 10

- 8238 • Minimum negative integer such that FLT_RADIX raised to that power minus 1 is a
 8239 normalized floating-point number, $e_{\min}$.

8240 FLT_MIN_EXP

8241 DBL_MIN_EXP

8242 LDBL_MIN_EXP

- 8243 • Minimum negative integer such that 10 raised to that power is in the range of normalized
 8244 floating-point numbers.

$$\left\lceil \log_{10} b^{e_{\min} - 1} \right\rceil$$

8245 FLT_MIN_10_EXP -37

8246 DBL_MIN_10_EXP -37

8247 LDBL_MIN_10_EXP -37

- 8248 • Maximum integer such that FLT_RADIX raised to that power minus 1 is a representable
 8249 finite floating-point number, $e_{\max}$.

8250 FLT_MAX_EXP

8251 DBL_MAX_EXP

8252 LDBL_MAX_EXP

8253 CX Additionally, FLT_MAX_EXP shall be at least as large as FLT_MANT_DIG, +
 8254 DBL_MAX_EXP shall be at least as large as DBL_MANT_DIG, and LDBL_MAX_EXP shall +
 8255 be at least as large as LDBL_MANT_DIG; which has the effect that FLT_MAX, DBL_MAX, +
 8256 and LDBL_MAX are integral.

- 8257 • Maximum integer such that 10 raised to that power is in the range of representable finite
 8258 floating-point numbers.

$$\left\lceil \log_{10}((1 - b^{-p}) b^{e_{\max}}) \right\rceil$$

8259 FLT_MAX_10_EXP +37

8260 DBL_MAX_10_EXP +37

8261 LDBL_MAX_10_EXP +37

8262 The <float.h> header shall define the following values as constant expressions with
8263 implementation-defined values that are greater than or equal to those shown:

- 8264
- Maximum representable finite floating-point number.

$$(1 - b^{-p}) b^{\epsilon_{\max}}$$

8265 FLT_MAX 1E+37

8266 DBL_MAX 1E+37

8267 LDBL_MAX 1E+37

8268 The <float.h> header shall define the following values as constant expressions with
8269 implementation-defined (positive) values that are less than or equal to those shown:

- 8270
- The difference between 1 and the least value greater than 1 that is representable in the
8271 given floating-point type, b^{1-p} .

8272 FLT_EPSILON 1E-5

8273 DBL_EPSILON 1E-9

8274 LDBL_EPSILON 1E-9

- 8275
- Minimum normalized positive floating-point number, $b^{\epsilon_{\min}-1}$.

8276 FLT_MIN 1E-37

8277 DBL_MIN 1E-37

8278 LDBL_MIN 1E-37

8279 APPLICATION USAGE

8280 None.

8281 RATIONALE

8282 All known hardware floating-point formats satisfy the property that the exponent range is larger
8283 than the number of mantissa digits. The ISO C standard permits a floating-point format where
8284 this property is not true, such that the largest finite value would not be integral; however, it is
8285 unlikely that there will ever be hardware support for such a floating-point format, and it
8286 introduces boundary cases that portable programs should not have to be concerned with (for
8287 example, a non-integral DBL_MAX means that *ceil()* would have to worry about overflow).
8288 Therefore, this standard imposes an additional requirement that the largest representable finite
8289 value is integral.

8290 FUTURE DIRECTIONS

8291 None.

8292 SEE ALSO

8293 <complex.h>, <math.h>, <stdio.h>, <stdlib.h>, <wchar.h>

CHANGE HISTORY

First released in Issue 4. Derived from the ISO C standard.

Issue 6

The description of the operations with floating-point values is updated for alignment with the ISO/IEC 9899:1999 standard.

Issue 7

ISO/IEC 9899:1999 standard, Technical Corrigendum 2 #4 (SD5-XBD-ERN-50) and #5 (SD5-XBD-ERN-51) are applied.

POSIX.1-2008, Technical Corrigendum 1, XBD/TC1-2008/0046 [346] and XBD/TC1-2008/0047 + [346] are applied.

NAME

fmtmsg.h — message display structures

SYNOPSISXSI `#include <fmtmsg.h>`**DESCRIPTION**

The <fmtmsg.h> header shall define the following symbolic constants:

MM_HARD	Source of the condition is hardware.
MM_SOFT	Source of the condition is software.
MM_FIRM	Source of the condition is firmware.
MM_APPL	Condition detected by application.
MM_UTIL	Condition detected by utility.
MM_OPSYS	Condition detected by operating system.
MM_RECOVER	Recoverable error.
MM_NRECOV	Non-recoverable error.
MM_HALT	Error causing application to halt.
MM_ERROR	Application has encountered a non-fatal fault.
MM_WARNING	Application has detected unusual non-error condition.
MM_INFO	Informative message.
MM_NOSEV	No severity level provided for the message.
MM_PRINT	Display message on standard error.
MM_CONSOLE	Display message on system console.

The table below indicates the null values and identifiers for *fmtmsg()* arguments. The <fmtmsg.h> header shall define the symbolic constants in the **Identifier** column, which shall have the type indicated in the **Type** column:

Argument	Type	Null-Value	Identifier
<i>label</i>	char *	(char*)0	MM_NULLLBL
<i>severity</i>	int	0	MM_NULLSEV
<i>class</i>	long	0L	MM_NULLMC
<i>text</i>	char *	(char*)0	MM_NULLTXT
<i>action</i>	char *	(char*)0	MM_NULLACT
<i>tag</i>	char *	(char*)0	MM_NULLTAG

The <fmtmsg.h> header shall also define the following symbolic constants for use as return values for *fmtmsg()*:

MM_OK	The function succeeded.
MM_NOTOK	The function failed completely.
MM_NOMSG	The function was unable to generate a message on standard error, but otherwise succeeded.

8341 MM_NOCON The function was unable to generate a console message, but otherwise
8342 succeeded.

8343 The following shall be declared as a function and may also be defined as a macro. A function
8344 prototype shall be provided.

8345 int fmtmsg(long, const char *, int,
8346 const char *, const char *, const char *);

8347 APPLICATION USAGE

8348 None.

8349 RATIONALE

8350 None.

8351 FUTURE DIRECTIONS

8352 None.

8353 SEE ALSO

8354 XSH *fmtmsg()*

8355 CHANGE HISTORY

8356 First released in Issue 4, Version 2.

8357 Issue 7

8358 This reference page is clarified with respect to macros and symbolic constants.

8359 **NAME**

8360 fnmatch.h — filename-matching types

8361 **SYNOPSIS**

8362 #include <fnmatch.h>

8363 **DESCRIPTION**

8364 The <fnmatch.h> header shall define the following symbolic constants:

8365 FNM_NOMATCH The string does not match the specified pattern.

8366 FNM_PATHNAME <slash> in *string* only matches <slash> in *pattern*.8367 FNM_PERIOD Leading <period> in *string* must be exactly matched by <period> in
8368 *pattern*.

8369 FNM_NOESCAPE Disable backslash escaping.

8370 The following shall be declared as a function and may also be defined as a macro. A function
8371 prototype shall be provided.

8372 int fnmatch(const char *, const char *, int);

8373 **APPLICATION USAGE**

8374 None.

8375 **RATIONALE**

8376 None.

8377 **FUTURE DIRECTIONS**

8378 None.

8379 **SEE ALSO**8380 XSH *fnmatch()*8381 **CHANGE HISTORY**

8382 First released in Issue 4. Derived from the ISO POSIX-2 standard.

8383 **Issue 6**

8384 The FNM_NOSYS constant is marked obsolescent.

8385 **Issue 7**

8386 The obsolescent FNM_NOSYS constant is removed.

8387 This reference page is clarified with respect to macros and symbolic constants.

NAME

ftw.h — file tree traversal

SYNOPSIS

```
XSI      #include <ftw.h>
```

DESCRIPTION

The <ftw.h> header shall define the **FTW** structure, which shall include at least the following members:

```
int  base
int  level
```

The <ftw.h> header shall define the following symbolic constants for use as values of the third argument to the application-supplied function that is passed as the second argument to *ftw()* and *nftw()*:

FTW_F	Non-directory file.
FTW_D	Directory.
FTW_DNR	Directory without read permission.
FTW_DP	Directory with subdirectories visited.
FTW_NS	Unknown type; <i>stat()</i> failed.
FTW_SL	Symbolic link.
FTW_SLN	Symbolic link that names a nonexistent file.

The <ftw.h> header shall define the following symbolic constants for use as values of the fourth argument to *nftw()*:

FTW_PHYS	Physical walk, does not follow symbolic links. Otherwise, <i>nftw()</i> follows links but does not walk down any path that crosses itself.
FTW_MOUNT	The walk does not cross a mount point.
FTW_DEPTH	All subdirectories are visited before the directory itself.
FTW_CHDIR	The walk changes to each directory before reading it.

The following shall be declared as functions and may also be defined as macros. Function prototypes shall be provided.

```
OB      int ftw(const char *, int (*)(const char *, const struct stat *,
8417      int), int);
8418      int nftw(const char *, int (*)(const char *, const struct stat *,
8419      int, struct FTW *), int, int);
```

The <ftw.h> header shall define the **stat** structure and the symbolic names for *st_mode* and the file type test macros as described in <sys/stat.h>.

Inclusion of the <ftw.h> header may also make visible all symbols from <sys/stat.h>.

8423 APPLICATION USAGE

8424 None.

8425 RATIONALE

8426 None.

8427 FUTURE DIRECTIONS

8428 None.

8429 SEE ALSO

8430 [**<sys/stat.h>**](#)

8431 XSH [*ftw\(\)*](#), [*nftw\(\)*](#)

8432 CHANGE HISTORY

8433 First released in Issue 1. Derived from Issue 1 of the SVID.

8434 Issue 5

8435 A description of FTW_DP is added.

8436 Issue 7

8437 The [*ftw\(\)*](#) function is marked obsolescent.

8438 This reference page is clarified with respect to macros and symbolic constants.

8439 POSIX.1-2008, Technical Corrigendum 1, XBD/TC1-2008/0048 [403] is applied.

+

8440 **NAME**

8441 glob.h — pathname pattern-matching types

8442 **SYNOPSIS**

8443 #include <glob.h>

8444 **DESCRIPTION**

8445 The <glob.h> header shall define the structures and symbolic constants used by the *glob()*
8446 function.

8447 The <glob.h> header shall define the **glob_t** structure type, which shall include at least the
8448 following members:

8449 size_t gl_pathc Count of paths matched by *pattern*.
8450 char **gl_pathv Pointer to a list of matched pathnames.
8451 size_t gl_offs Slots to reserve at the beginning of *gl_pathv*.

8452 The <glob.h> header shall define the **size_t** type as described in <sys/types.h>.

8453 The <glob.h> header shall define the following symbolic constants as values for the *flags*
8454 argument:

8455 GLOB_APPEND Append generated pathnames to those previously obtained.
8456 GLOB_DOOFFS Specify how many null pointers to add to the beginning of *gl_pathv*.
8457 GLOB_ERR Cause *glob()* to return on error.
8458 GLOB_MARK Each pathname that is a directory that matches *pattern* has a <slash>
8459 appended.
8460 GLOB_NOCHECK If *pattern* does not match any pathname, then return a list consisting of
8461 only *pattern*.
8462 GLOB_NOESCAPE Disable backslash escaping.
8463 GLOB_NOSORT Do not sort the pathnames returned.
8464 The <glob.h> header shall define the following symbolic constants as error return values:
8465 GLOB_ABORTED The scan was stopped because GLOB_ERR was set or (*errfunc)() returned non-zero.
8466
8467 GLOB_NOMATCH The pattern does not match any existing pathname, and
8468 GLOB_NOCHECK was not set in *flags*.
8469 GLOB_NOSPACE An attempt to allocate memory failed.

8470 The following shall be declared as functions and may also be defined as macros. Function
8471 prototypes shall be provided.

8472 int glob(const char *restrict, int, int (*)(const char *, int),
8473 glob_t *restrict);
8474 void globfree(glob_t *);

8475 **APPLICATION USAGE**

8476 None.

8477 **RATIONALE**

8478 None.

8479 **FUTURE DIRECTIONS**

8480 None.

8481 **SEE ALSO**

8482 <sys/types.h>

8483 XSH *glob()*8484 **CHANGE HISTORY**

8485 First released in Issue 4. Derived from the ISO POSIX-2 standard.

8486 **Issue 6**8487 The **restrict** keyword is added to the prototype for *glob()*.

8488 The GLOB_NOSYS constant is marked obsolescent.

8489 IEEE Std 1003.1-2001/Cor 1-2002, item XBD/TC1/D6/8 is applied, correcting the *glob()*
8490 prototype definition by removing the **restrict** qualifier from the function pointer argument.8491 **Issue 7**8492 SD5-XBD-ERN-56 is applied, adding a reference to <sys/types.h> for the **size_t**

8493 The obsolescent GLOB_NOSYS constant is removed.

8494 This reference page is clarified with respect to macros and symbolic constants.

8495 **NAME**

8496 grp.h — group structure

8497 **SYNOPSIS**

8498 #include <grp.h>

8499 **DESCRIPTION**

8500 The <grp.h> header shall declare the **group** structure, which shall include the following
8501 members:

8502 char *gr_name The name of the group.
8503 gid_t gr_gid Numerical group ID.
8504 char **gr_mem Pointer to a null-terminated array of character
8505 pointers to member names.

8506 The <grp.h> header shall define the **gid_t** and **size_t** types as described in <sys/types.h>.

8507 The following shall be declared as functions and may also be defined as macros. Function
8508 prototypes shall be provided.

8509 XSI void endgrent(void);
8510 struct group *getgrent(void);
8511 struct group *getgrgid(gid_t);
8512 int getgrgid_r(gid_t, struct group *, char *,
8513 size_t, struct group **);
8514 struct group *getgrnam(const char *);
8515 int getgrnam_r(const char *, struct group *, char *,
8516 size_t, struct group **);
8517 XSI void setgrent(void);

8518 **APPLICATION USAGE**

8519 None.

8520 **RATIONALE**

8521 None.

8522 **FUTURE DIRECTIONS**

8523 None.

8524 **SEE ALSO**

8525 <sys/types.h>

8526 XSH *endgrent()*, *getgrgid()*, *getgrnam()*

8527 **CHANGE HISTORY**

8528 First released in Issue 1.

8529 **Issue 5**

8530 The DESCRIPTION is updated for alignment with the POSIX Threads Extension.

8531 **Issue 6**

8532 The following new requirements on POSIX implementations derive from alignment with the
8533 Single UNIX Specification:

- 8534 • The definition of **gid_t** is mandated.
- 8535 • The *getgrgid_r()* and *getgrnam_r()* functions are marked as part of the Thread-Safe
8536 Functions option.

8537 **Issue 7**

8538 SD5-XBD-ERN-56 is applied, adding a reference to <sys/types.h> for the **size_t** type.

8539 POSIX.1-2008, Technical Corrigendum 1, XBD/TC1-2008/0049 [24] is applied.

8540 **NAME**

8541 iconv.h — codeset conversion facility

8542 **SYNOPSIS**

8543 #include <iconv.h>

8544 **DESCRIPTION**

8545 The <iconv.h> header shall define the following types:

8546 **iconv_t** Identifies the conversion from one codeset to another.

8547 **size_t** As described in <sys/types.h>.

8548 The following shall be declared as functions and may also be defined as macros. Function
8549 prototypes shall be provided.

```
8550 size_t iconv(iconv_t, char **restrict, size_t *restrict,
8551             char **restrict, size_t *restrict);
8552 int iconv_close(iconv_t);
8553 iconv_t iconv_open(const char *, const char *);
```

8554 **APPLICATION USAGE**

8555 None.

8556 **RATIONALE**

8557 None.

8558 **FUTURE DIRECTIONS**

8559 None.

8560 **SEE ALSO**

8561 <sys/types.h>

8562 XSH *iconv()*, *iconv_close()*, *iconv_open()*

8563 **CHANGE HISTORY**

8564 First released in Issue 4.

8565 **Issue 6**

8566 The **restrict** keyword is added to the prototype for *iconv()*.

8567 **Issue 7**

8568 SD5-XBD-ERN-56 is applied, adding a reference to <sys/types.h> for the **size_t** type.

8569 The <iconv.h> header is moved from the XSI option to the Base.

8570 NAME

8571 inttypes.h — fixed size integer types

8572 SYNOPSIS

8573 #include <inttypes.h>

8574 DESCRIPTION

8575 CX Some of the functionality described on this reference page extends the ISO C standard.
 8576 Applications shall define the appropriate feature test macro (see XSH Section 2.2, on page 472) to
 8577 enable the visibility of these symbols in this header.

8578 The <inttypes.h> header shall include the <stdint.h> header.

8579 The <inttypes.h> header shall define at least the following types: |

8580 **imaxdiv_t** Structure type that is the type of the value returned by the *imaxdiv()* function.8581 CX **wchar_t** As described in <stddef.h>. +

8582 The <inttypes.h> header shall define the following macros. Each expands to a character string
 8583 literal containing a conversion specifier, possibly modified by a length modifier, suitable for use
 8584 within the *format* argument of a formatted input/output function when converting the
 8585 corresponding integer type. These macros have the general form of PRI (character string literals
 8586 for the *fprintf()* and *fwprintf()* family of functions) or SCN (character string literals for the
 8587 *fscanf()* and *fwscanf()* family of functions), followed by the conversion specifier, followed by a
 8588 name corresponding to a similar type name in <stdint.h>. In these names, *N* represents the
 8589 width of the type as described in <stdint.h>. For example, *PRIdFAST32* can be used in a format
 8590 string to print the value of an integer of type **int_fast32_t**.

8591 The *fprintf()* macros for signed integers are:

8592	PRIdN	PRIdLEASTN	PRIdFASTN	PRIdMAX	PRIdPTR
8593	PRiN	PRiLEASTN	PRiFASTN	PRiMAX	PRiPTR

8594 The *fprintf()* macros for unsigned integers are:

8595	PRIoN	PRIoLEASTN	PRIoFASTN	PRIoMAX	PRIoPTR
8596	PRiUN	PRiULEASTN	PRiUFASTN	PRiUMAX	PRiUPTR
8597	PRIxN	PRiXLEASTN	PRiXFASTN	PRiXMAX	PRiXPTR
8598	PRIXN	PRiXLEASTN	PRiXFASTN	PRiXMAX	PRiXPTR

8599 The *fscanf()* macros for signed integers are:

8600	SCNdN	SCNdLEASTN	SCNdFASTN	SCNdMAX	SCNdPTR
8601	SCNiN	SCNiLEASTN	SCNiFASTN	SCNiMAX	SCNiPTR

8602 The *fscanf()* macros for unsigned integers are:

8603	SCNoN	SCNoLEASTN	SCNoFASTN	SCNoMAX	SCNoPTR
8604	SCNuN	SCNuLEASTN	SCNuFASTN	SCNuMAX	SCNuPTR
8605	SCNxN	SCNxLEASTN	SCNxFASTN	SCNxMAX	SCNxPTR

8606 For each type that the implementation provides in <stdint.h>, the corresponding *fprintf()* and
 8607 *fwprintf()* macros shall be defined and the corresponding *fscanf()* and *fwscanf()* macros shall be
 8608 defined unless the implementation does not have a suitable modifier for the type.

8609 The following shall be declared as functions and may also be defined as macros. Function
 8610 prototypes shall be provided.

```

8611     intmax_t  imaxabs(intmax_t);
8612     imaxdiv_t imaxdiv(intmax_t, intmax_t);
8613     intmax_t  strtoumax(const char *restrict, char **restrict, int);
8614     uintmax_t strtoumax(const char *restrict, char **restrict, int);
8615     intmax_t  wcstoumax(const wchar_t *restrict, wchar_t **restrict, int);
8616     uintmax_t wcstoumax(const wchar_t *restrict, wchar_t **restrict, int);

```

EXAMPLES

```

8617     #include <inttypes.h>
8618     #include <wchar.h>
8619     int main(void)
8620     {
8621         uintmax_t i = UINTMAX_MAX; // This type always exists.
8622         wprintf(L"The largest integer value is %020"
8623             PRIxMAX "\n", i);
8624         return 0;
8625     }
8626

```

APPLICATION USAGE

The purpose of **<inttypes.h>** is to provide a set of integer types whose definitions are consistent across machines and independent of operating systems and other implementation idiosyncrasies. It defines, through **typedef**, integer types of various sizes. Implementations are free to **typedef** them as ISO C standard integer types or extensions that they support. Consistent use of this header will greatly increase the portability of applications across platforms.

RATIONALE

The ISO/IEC 9899:1990 standard specified that the language should support four signed and unsigned integer data types—**char**, **short**, **int**, and **long**—but placed very little requirement on their size other than that **int** and **short** be at least 16 bits and **long** be at least as long as **int** and not smaller than 32 bits. For 16-bit systems, most implementations assigned 8, 16, 16, and 32 bits to **char**, **short**, **int**, and **long**, respectively. For 32-bit systems, the common practice has been to assign 8, 16, 32, and 32 bits to these types. This difference in **int** size can create some problems for users who migrate from one system to another which assigns different sizes to integer types, because the ISO C standard integer promotion rule can produce silent changes unexpectedly. The need for defining an extended integer type increased with the introduction of 64-bit systems.

FUTURE DIRECTIONS

Macro names beginning with PRI or SCN followed by any lowercase letter or 'X' may be added to the macros defined in the **<inttypes.h>** header.

SEE ALSO

<stddef.h>

XSH Section 2.2 (on page 472), *imaxabs()*, *imaxdiv()*, *strtoumax()*, *wcstoumax()*

CHANGE HISTORY

First released in Issue 5.

Issue 6

The Open Group Base Resolution bwg97-006 is applied.

This reference page is updated to align with the ISO/IEC 9899:1999 standard.

8655 **Issue 7**
8656

POSIX.1-2008, Technical Corrigendum 1, XBD/TC1-2008/0050 [211] is applied.

+

8657 **NAME**

8658 iso646.h — alternative spellings

8659 **SYNOPSIS**

8660 #include <iso646.h>

8661 **DESCRIPTION**

8662 CX The functionality described on this reference page is aligned with the ISO C standard. Any
8663 conflict between the requirements described here and the ISO C standard is unintentional. This
8664 volume of POSIX.1-2008 defers to the ISO C standard.

8665 The <iso646.h> header shall define the following eleven macros (on the left) that expand to the
8666 corresponding tokens (on the right):

8667	and	&&
8668	and_eq	&=
8669	bitand	&
8670	bitor	
8671	compl	~
8672	not	!
8673	not_eq	!=
8674	or	
8675	or_eq	=
8676	xor	^
8677	xor_eq	^=

8678 **APPLICATION USAGE**

8679 None.

8680 **RATIONALE**

8681 None.

8682 **FUTURE DIRECTIONS**

8683 None.

8684 **SEE ALSO**

8685 None.

8686 **CHANGE HISTORY**

8687 First released in Issue 5. Derived from ISO/IEC 9899:1990/Amendment 1:1995 (E).

8688 **NAME**

8689 langinfo.h — language information constants

8690 **SYNOPSIS**

8691 #include <langinfo.h>

8692 **DESCRIPTION**8693 The <langinfo.h> header shall define the symbolic constants used to identify items of *langinfo*
8694 data (see *nl_langinfo()*).8695 The <langinfo.h> header shall define the **locale_t** type as described in <locale.h>.8696 The <langinfo.h> header shall define the **nl_item** type as described in <nl_types.h>.8697 The <langinfo.h> header shall define the following symbolic constants with type **nl_item**. The
8698 entries under **Category** indicate in which *setlocale()* category each item is defined.

Constant	Category	Meaning
CODESET	LC_CTYPE	Codeset name.
D_T_FMT	LC_TIME	String for formatting date and time.
D_FMT	LC_TIME	Date format string.
T_FMT	LC_TIME	Time format string.
T_FMT_AMPM	LC_TIME	a.m. or p.m. time format string.
AM_STR	LC_TIME	Ante-meridiem affix.
PM_STR	LC_TIME	Post-meridiem affix.
DAY_1	LC_TIME	Name of the first day of the week (for example, Sunday).
DAY_2	LC_TIME	Name of the second day of the week (for example, Monday).
DAY_3	LC_TIME	Name of the third day of the week (for example, Tuesday).
DAY_4	LC_TIME	Name of the fourth day of the week (for example, Wednesday).
DAY_5	LC_TIME	Name of the fifth day of the week (for example, Thursday).
DAY_6	LC_TIME	Name of the sixth day of the week (for example, Friday).
DAY_7	LC_TIME	Name of the seventh day of the week (for example, Saturday).
ABDAY_1	LC_TIME	Abbreviated name of the first day of the week.
ABDAY_2	LC_TIME	Abbreviated name of the second day of the week.
ABDAY_3	LC_TIME	Abbreviated name of the third day of the week.
ABDAY_4	LC_TIME	Abbreviated name of the fourth day of the week.
ABDAY_5	LC_TIME	Abbreviated name of the fifth day of the week.
ABDAY_6	LC_TIME	Abbreviated name of the sixth day of the week.
ABDAY_7	LC_TIME	Abbreviated name of the seventh day of the week.
MON_1	LC_TIME	Name of the first month of the year.
MON_2	LC_TIME	Name of the second month.
MON_3	LC_TIME	Name of the third month.
MON_4	LC_TIME	Name of the fourth month.
MON_5	LC_TIME	Name of the fifth month.
MON_6	LC_TIME	Name of the sixth month.
MON_7	LC_TIME	Name of the seventh month.
MON_8	LC_TIME	Name of the eighth month.
MON_9	LC_TIME	Name of the ninth month.
MON_10	LC_TIME	Name of the tenth month.
MON_11	LC_TIME	Name of the eleventh month.
MON_12	LC_TIME	Name of the twelfth month.
ABMON_1	LC_TIME	Abbreviated name of the first month.

Constant	Category	Meaning
ABMON_2	LC_TIME	Abbreviated name of the second month.
ABMON_3	LC_TIME	Abbreviated name of the third month.
ABMON_4	LC_TIME	Abbreviated name of the fourth month.
ABMON_5	LC_TIME	Abbreviated name of the fifth month.
ABMON_6	LC_TIME	Abbreviated name of the sixth month.
ABMON_7	LC_TIME	Abbreviated name of the seventh month.
ABMON_8	LC_TIME	Abbreviated name of the eighth month.
ABMON_9	LC_TIME	Abbreviated name of the ninth month.
ABMON_10	LC_TIME	Abbreviated name of the tenth month.
ABMON_11	LC_TIME	Abbreviated name of the eleventh month.
ABMON_12	LC_TIME	Abbreviated name of the twelfth month.
ERA	LC_TIME	Era description segments.
ERA_D_FMT	LC_TIME	Era date format string.
ERA_D_T_FMT	LC_TIME	Era date and time format string.
ERA_T_FMT	LC_TIME	Era time format string.
ALT_DIGITS	LC_TIME	Alternative symbols for digits.
RADIXCHAR	LC_NUMERIC	Radix character.
THOUSEP	LC_NUMERIC	Separator for thousands.
YESEXPR	LC_MESSAGES	Affirmative response expression.
NOEXPR	LC_MESSAGES	Negative response expression.
CRNCYSTR	LC_MONETARY	Local currency symbol, preceded by '-' if the symbol should appear before the value, '+' if the symbol should appear after the value, or '.' if the symbol should replace the radix character. If the local currency symbol is the empty string, implementations may return the empty string ("").

If the locale's values for **p_cs_precedes** and **n_cs_precedes** do not match, the value of `nl_langinfo(CRNCYSTR)` and `nl_langinfo_l(CRNCYSTR, loc)` is unspecified.

The following shall be declared as a function and may also be defined as a macro. A function prototype shall be provided.

```
char *nl_langinfo(nl_item);
char *nl_langinfo_l(nl_item, locale_t);
```

Inclusion of the <langinfo.h> header may also make visible all symbols from <nl_types.h>.

APPLICATION USAGE

Wherever possible, users are advised to use functions compatible with those in the ISO C standard to access items of *langinfo* data. In particular, the `strftime()` function should be used to access date and time information defined in category `LC_TIME`. The `localeconv()` function should be used to access information corresponding to `RADIXCHAR`, `THOUSEP`, and `CRNCYSTR`.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

Chapter 7 (on page 135), <locale.h>, <nl_types.h>

XSH `nl_langinfo()`, `localeconv()`, `strfmon()`, `strftime()`

8782 **CHANGE HISTORY**

8783 First released in Issue 2.

8784 **Issue 5**

8785 The constants YESSTR and NOSTR are marked LEGACY.

8786 **Issue 6**

8787 The constants YESSTR and NOSTR are removed.

8788 IEEE Std 1003.1-2001/Cor 1-2002, item XBD/TC1/D6/9 is applied, adding a sentence to
8789 the “Meaning” column entry for the CRNCYSTR constant. This change is to accommodate
8790 historic practice.

8791 **Issue 7**

8792 The <langinfo.h> header is moved from the XSI option to the Base.

8793 The *nl_langinfo_l()* function is added from The Open Group Technical Standard, 2006, Extended
8794 API Set Part 4.

8795 This reference page is clarified with respect to macros and symbolic constants, and a declaration
8796 for the **locale_t** type is added.

8797 POSIX.1-2008, Technical Corrigendum 1, XBD/TC1-2008/0051 [107] is applied.

+

8798 **NAME**
8799 libgen.h — definitions for pattern matching functions

8800 **SYNOPSIS**

8801 XSI `#include <libgen.h>`

8802 **DESCRIPTION**

8803 The following shall be declared as functions and may also be defined as macros. Function
8804 prototypes shall be provided.

8805 `char *basename(char *);`
8806 `char *dirname(char *);`

8807 **APPLICATION USAGE**

8808 None.

8809 **RATIONALE**

8810 None.

8811 **FUTURE DIRECTIONS**

8812 None.

8813 **SEE ALSO**

8814 XSH *basename()*, *dirname()*

8815 **CHANGE HISTORY**

8816 First released in Issue 4, Version 2.

8817 **Issue 5**

8818 The function prototypes for *basename()* and *dirname()* are changed to indicate that the first
8819 argument is of type **char *** rather than **const char ***.

8820 **Issue 6**

8821 The **__loc1** symbol and the *regcmp()* and *regex()* functions are removed.

NAME

limits.h — implementation-defined constants

SYNOPSIS

```
#include <limits.h>
```

DESCRIPTION

Some of the functionality described on this reference page extends the ISO C standard. Applications shall define the appropriate feature test macro (see XSH [Section 2.2](#), on page 472) to enable the visibility of these symbols in this header.

Many of the symbols listed here are not defined by the ISO/IEC 9899:1999 standard. Such symbols are not shown as CX shaded, except under the heading “Numerical Limits”.

The <limits.h> header shall define macros and symbolic constants for various limits. Different categories of limits are described below, representing various limits on resources that the implementation imposes on applications. All macros and symbolic constants defined in this header shall be suitable for use in #if preprocessing directives.

Implementations may choose any appropriate value for each limit, provided it is not more restrictive than the Minimum Acceptable Values listed below. Symbolic constant names beginning with _POSIX may be found in <unistd.h>.

Applications should not assume any particular value for a limit. To achieve maximum portability, an application should not require more resource than the Minimum Acceptable Value quantity. However, an application wishing to avail itself of the full amount of a resource available on an implementation may make use of the value given in <limits.h> on that particular implementation, by using the macros and symbolic constants listed below. It should be noted, however, that many of the listed limits are not invariant, and at runtime, the value of the limit may differ from those given in this header, for the following reasons:

- The limit is pathname-dependent.
- The limit differs between the compile and runtime machines.

For these reasons, an application may use the *fpathconf()*, *pathconf()*, and *sysconf()* functions to determine the actual value of a limit at runtime.

The items in the list ending in _MIN give the most negative values that the mathematical types are guaranteed to be capable of representing. Numbers of a more negative value may be supported on some implementations, as indicated by the <limits.h> header on the implementation, but applications requiring such numbers are not guaranteed to be portable to all implementations. For positive constants ending in _MIN, this indicates the minimum acceptable value.

Runtime Invariant Values (Possibly Indeterminate)

A definition of one of the symbolic constants in the following list shall be omitted from <limits.h> on specific implementations where the corresponding value is equal to or greater than the stated minimum, but is unspecified.

This indetermination might depend on the amount of available memory space on a specific instance of a specific implementation. The actual value supported by a specific instance shall be provided by the *sysconf()* function.

{AIO_LISTIO_MAX}

Maximum number of I/O operations in a single list I/O call supported by the implementation.

Minimum Acceptable Value: {_POSIX_AIO_LISTIO_MAX}

8867 {AIO_MAX}
8868 Maximum number of outstanding asynchronous I/O operations supported by the
8869 implementation.
8870 Minimum Acceptable Value: {_POSIX_AIO_MAX}

8871 {AIO_PRIO_DELTA_MAX}
8872 The maximum amount by which a process can decrease its asynchronous I/O priority level
8873 from its own scheduling priority.
8874 Minimum Acceptable Value: 0

8875 {ARG_MAX}
8876 Maximum length of argument to the *exec* functions including environment data.
8877 Minimum Acceptable Value: {_POSIX_ARG_MAX}

8878 {ATEXIT_MAX}
8879 Maximum number of functions that may be registered with *atexit()*.
8880 Minimum Acceptable Value: 32

8881 {CHILD_MAX}
8882 Maximum number of simultaneous processes per real user ID.
8883 Minimum Acceptable Value: {_POSIX_CHILD_MAX}

8884 {DELAYTIMER_MAX}
8885 Maximum number of timer expiration overruns.
8886 Minimum Acceptable Value: {_POSIX_DELAYTIMER_MAX}

8887 {HOST_NAME_MAX}
8888 Maximum length of a host name (not including the terminating null) as returned from the
8889 *gethostname()* function.
8890 Minimum Acceptable Value: {_POSIX_HOST_NAME_MAX}

8891 XSI {IOV_MAX}
8892 Maximum number of **iovec** structures that one process has available for use with *readv()* or
8893 *writev()*.
8894 Minimum Acceptable Value: {_XOPEN_IOV_MAX}

8895 {LOGIN_NAME_MAX}
8896 Maximum length of a login name.
8897 Minimum Acceptable Value: {_POSIX_LOGIN_NAME_MAX}

8898 MSG {MQ_OPEN_MAX}
8899 The maximum number of open message queue descriptors a process may hold.
8900 Minimum Acceptable Value: {_POSIX_MQ_OPEN_MAX}

8901 MSG {MQ_PRIO_MAX}
8902 The maximum number of message priorities supported by the implementation.
8903 Minimum Acceptable Value: {_POSIX_MQ_PRIO_MAX}

8904 {OPEN_MAX}
8905 A value one greater than the maximum value that the system may assign to a newly-created
8906 file descriptor.
8907 Minimum Acceptable Value: {_POSIX_OPEN_MAX}

8908 {PAGESIZE}
8909 Size in bytes of a page.
8910 Minimum Acceptable Value: 1

8911 XSI **{PAGE_SIZE}**
8912 Equivalent to {PAGESIZE}. If either {PAGESIZE} or {PAGE_SIZE} is defined, the other is
8913 defined with the same value.

8914 {PTHREAD_DESTRUCTOR_ITERATIONS}
8915 Maximum number of attempts made to destroy a thread's thread-specific data values on
8916 thread exit.
8917 Minimum Acceptable Value: {_POSIX_THREAD_DESTRUCTOR_ITERATIONS}

8918 {PTHREAD_KEYS_MAX}
8919 Maximum number of data keys that can be created by a process.
8920 Minimum Acceptable Value: {_POSIX_THREAD_KEYS_MAX}

8921 {PTHREAD_STACK_MIN}
8922 Minimum size in bytes of thread stack storage.
8923 Minimum Acceptable Value: 0

8924 {PTHREAD_THREADS_MAX}
8925 Maximum number of threads that can be created per process.
8926 Minimum Acceptable Value: {_POSIX_THREAD_THREADS_MAX}

8927 {RE_DUP_MAX}
8928 Maximum number of repeated occurrences of a BRE or ERE interval expression; see [Section](#)
8929 [9.3.6](#) (on page 186) and [Section 9.4.6](#) (on page 189).
8930 Minimum Acceptable Value: {_POSIX2_RE_DUP_MAX}

8931 {RTSIG_MAX}
8932 Maximum number of realtime signals reserved for application use in this implementation.
8933 Minimum Acceptable Value: {_POSIX_RTSIG_MAX}

8934 {SEM_NSEMS_MAX}
8935 Maximum number of semaphores that a process may have.
8936 Minimum Acceptable Value: {_POSIX_SEM_NSEMS_MAX}

8937 {SEM_VALUE_MAX}
8938 The maximum value a semaphore may have.
8939 Minimum Acceptable Value: {_POSIX_SEM_VALUE_MAX}

8940 {SIGQUEUE_MAX}
8941 Maximum number of queued signals that a process may send and have pending at the
8942 receiver(s) at any time.
8943 Minimum Acceptable Value: {_POSIX_SIGQUEUE_MAX}

8944 SS | TSP **{SS_REPL_MAX}**
8945 The maximum number of replenishment operations that may be simultaneously pending
8946 for a particular sporadic server scheduler.
8947 Minimum Acceptable Value: {_POSIX_SS_REPL_MAX}

8948 {STREAM_MAX}
8949 Maximum number of streams that one process can have open at one time. If defined, it has
8950 the same value as {FOPEN_MAX} (see <stdio.h>).
8951 Minimum Acceptable Value: {_POSIX_STREAM_MAX}

8952 {SYMLOOP_MAX}
8953 Maximum number of symbolic links that can be reliably traversed in the resolution of a
8954 pathname in the absence of a loop.
8955 Minimum Acceptable Value: {_POSIX_SYMLOOP_MAX}

8956 {TIMER_MAX}
 8957 Maximum number of timers per process supported by the implementation.
 8958 Minimum Acceptable Value: {_POSIX_TIMER_MAX}

8959 OB TRC {TRACE_EVENT_NAME_MAX}
 8960 Maximum length of the trace event name (not including the terminating null).
 8961 Minimum Acceptable Value: {_POSIX_TRACE_EVENT_NAME_MAX}

8962 OB TRC {TRACE_NAME_MAX}
 8963 Maximum length of the trace generation version string or of the trace stream name (not
 8964 including the terminating null).
 8965 Minimum Acceptable Value: {_POSIX_TRACE_NAME_MAX}

8966 OB TRC {TRACE_SYS_MAX}
 8967 Maximum number of trace streams that may simultaneously exist in the system.
 8968 Minimum Acceptable Value: {_POSIX_TRACE_SYS_MAX}

8969 OB TRC {TRACE_USER_EVENT_MAX}
 8970 Maximum number of user trace event type identifiers that may simultaneously exist in a
 8971 traced process, including the predefined user trace event
 8972 POSIX_TRACE_UNNAMED_USER_EVENT.
 8973 Minimum Acceptable Value: {_POSIX_TRACE_USER_EVENT_MAX}

8974 {TTY_NAME_MAX}
 8975 Maximum length of terminal device name.
 8976 Minimum Acceptable Value: {_POSIX_TTY_NAME_MAX}

8977 {TZNAME_MAX}
 8978 Maximum number of bytes supported for the name of a timezone (not of the TZ variable).
 8979 Minimum Acceptable Value: {_POSIX_TZNAME_MAX}

8980 **Note:** The length given by {TZNAME_MAX} does not include the quoting characters mentioned in
 8981 [Section 8.3](#) (on page 177).

Pathname Variable Values

The values in the following list may be constants within an implementation or may vary from one pathname to another. For example, file systems or directories may have different characteristics.

A definition of one of the symbolic constants in the following list shall be omitted from the <limits.h> header on specific implementations where the corresponding value is equal to or greater than the stated minimum, but where the value can vary depending on the file to which it is applied. The actual value supported for a specific pathname shall be provided by the *pathconf()* function.

8991 {FILESIZEBITS}
 8992 Minimum number of bits needed to represent, as a signed integer value, the maximum size
 8993 of a regular file allowed in the specified directory.
 8994 Minimum Acceptable Value: 32

8995 {LINK_MAX}
 8996 Maximum number of links to a single file.
 8997 Minimum Acceptable Value: {_POSIX_LINK_MAX}

8998 {MAX_CANON}
 8999 Maximum number of bytes in a terminal canonical input line.
 9000 Minimum Acceptable Value: {_POSIX_MAX_CANON}

9001		{MAX_INPUT}
9002		Minimum number of bytes for which space is available in a terminal input queue; therefore,
9003		the maximum number of bytes a conforming application may require to be typed as input
9004		before reading them.
9005		Minimum Acceptable Value: {_POSIX_MAX_INPUT}
9006		{NAME_MAX}
9007		Maximum number of bytes in a filename (not including the terminating null of a filename
9008		string).
9009		Minimum Acceptable Value: {_POSIX_NAME_MAX}
9010	XSI	Minimum Acceptable Value: {_XOPEN_NAME_MAX}
9011		{PATH_MAX}
9012		Maximum number of bytes the implementation will store as a pathname in a user-supplied
9013		buffer of unspecified size, including the terminating null character. Minimum number the
9014		implementation will accept as the maximum number of bytes in a pathname.
9015		Minimum Acceptable Value: {_POSIX_PATH_MAX}
9016	XSI	Minimum Acceptable Value: {_XOPEN_PATH_MAX}
9017		{PIPE_BUF}
9018		Maximum number of bytes that is guaranteed to be atomic when writing to a pipe.
9019		Minimum Acceptable Value: {_POSIX_PIPE_BUF}
9020	ADV	{POSIX_ALLOC_SIZE_MIN}
9021		Minimum number of bytes of storage actually allocated for any portion of a file.
9022		Minimum Acceptable Value: Not specified.
9023	ADV	{POSIX_REC_INCR_XFER_SIZE}
9024		Recommended increment for file transfer sizes between the
9025		{POSIX_REC_MIN_XFER_SIZE} and {POSIX_REC_MAX_XFER_SIZE} values.
9026		Minimum Acceptable Value: Not specified.
9027	ADV	{POSIX_REC_MAX_XFER_SIZE}
9028		Maximum recommended file transfer size.
9029		Minimum Acceptable Value: Not specified.
9030	ADV	{POSIX_REC_MIN_XFER_SIZE}
9031		Minimum recommended file transfer size.
9032		Minimum Acceptable Value: Not specified.
9033	ADV	{POSIX_REC_XFER_ALIGN}
9034		Recommended file transfer buffer alignment.
9035		Minimum Acceptable Value: Not specified.
9036		{SYMLINK_MAX}
9037		Maximum number of bytes in a symbolic link.
9038		Minimum Acceptable Value: {_POSIX_SYMLINK_MAX}

Runtime Increaseable Values

The magnitude limitations in the following list shall be fixed by specific implementations. An application should assume that the value of the symbolic constant defined by <limits.h> in a specific implementation is the minimum that pertains whenever the application is run under that implementation. A specific instance of a specific implementation may increase the value relative to that supplied by <limits.h> for that implementation. The actual value supported by a specific instance shall be provided by the *sysconf()* function.

{BC_BASE_MAX}

Maximum *obase* values allowed by the *bc* utility.

Minimum Acceptable Value: {_POSIX2_BC_BASE_MAX}

{BC_DIM_MAX}

Maximum number of elements permitted in an array by the *bc* utility.

Minimum Acceptable Value: {_POSIX2_BC_DIM_MAX}

{BC_SCALE_MAX}

Maximum *scale* value allowed by the *bc* utility.

Minimum Acceptable Value: {_POSIX2_BC_SCALE_MAX}

{BC_STRING_MAX}

Maximum length of a string constant accepted by the *bc* utility.

Minimum Acceptable Value: {_POSIX2_BC_STRING_MAX}

{CHARCLASS_NAME_MAX}

Maximum number of bytes in a character class name.

Minimum Acceptable Value: {_POSIX2_CHARCLASS_NAME_MAX}

{COLL_WEIGHTS_MAX}

Maximum number of weights that can be assigned to an entry of the *LC_COLLATE* **order** keyword in the locale definition file; see [Chapter 7](#) (on page 135).

Minimum Acceptable Value: {_POSIX2_COLL_WEIGHTS_MAX}

{EXPR_NEST_MAX}

Maximum number of expressions that can be nested within parentheses by the *expr* utility.

Minimum Acceptable Value: {_POSIX2_EXPR_NEST_MAX}

{LINE_MAX}

Unless otherwise noted, the maximum length, in bytes, of a utility's input line (either standard input or another file), when the utility is described as processing text files. The length includes room for the trailing <newline>.

Minimum Acceptable Value: {_POSIX2_LINE_MAX}

{NGROUPS_MAX}

Maximum number of simultaneous supplementary group IDs per process.

Minimum Acceptable Value: {_POSIX2_NGROUPS_MAX}

{RE_DUP_MAX}

Maximum number of repeated occurrences of a regular expression permitted when using the interval notation *\{m,n\}*; see [Chapter 9](#) (on page 181).

Minimum Acceptable Value: {_POSIX2_RE_DUP_MAX}

9080 **Maximum Values**

9081 The <limits.h> header shall define the following symbolic constants with the values shown.
 9082 These are the most restrictive values for certain features on an implementation. A conforming
 9083 implementation shall provide values no larger than these values. A conforming application must
 9084 not require a smaller value for correct operation.

9085 {_POSIX_CLOCKRES_MIN}

9086 The resolution of the CLOCK_REALTIME clock, in nanoseconds.

9087 Value: 20 000 000

9088 MON If the Monotonic Clock option is supported, the resolution of the CLOCK_MONOTONIC
 9089 clock, in nanoseconds, is represented by {_POSIX_CLOCKRES_MIN}.

9090 **Minimum Values**

9091 The <limits.h> header shall define the following symbolic constants with the values shown.
 9092 These are the most restrictive values for certain features on an implementation conforming to
 9093 this volume of POSIX.1-2008. Related symbolic constants are defined elsewhere in this volume
 9094 of POSIX.1-2008 which reflect the actual implementation and which need not be as restrictive. |
 9095 For each of these limits, a conforming implementation shall provide a value at least this large or |
 9096 shall have no limit. A strictly conforming application must not require a larger value for correct |
 9097 operation.

9098 {_POSIX_AIO_LISTIO_MAX}

9099 The number of I/O operations that can be specified in a list I/O call.

9100 Value: 2

9101 {_POSIX_AIO_MAX}

9102 The number of outstanding asynchronous I/O operations.

9103 Value: 1

9104 {_POSIX_ARG_MAX}

9105 Maximum length of argument to the *exec* functions including environment data.

9106 Value: 4 096

9107 {_POSIX_CHILD_MAX}

9108 Maximum number of simultaneous processes per real user ID.

9109 Value: 25

9110 {_POSIX_DELAYTIMER_MAX}

9111 The number of timer expiration overruns.

9112 Value: 32

9113 {_POSIX_HOST_NAME_MAX}

9114 Maximum length of a host name (not including the terminating null) as returned from the
 9115 *gethostname()* function.

9116 Value: 255

9117 {_POSIX_LINK_MAX}

9118 Maximum number of links to a single file.

9119 Value: 8

9120 {_POSIX_LOGIN_NAME_MAX}

9121 The size of the storage required for a login name, in bytes (including the terminating null).

9122 Value: 9

9123		{_POSIX_MAX_CANON}	
9124		Maximum number of bytes in a terminal canonical input queue.	
9125		Value: 255	
9126		{_POSIX_MAX_INPUT}	
9127		Maximum number of bytes allowed in a terminal input queue.	
9128		Value: 255	
9129	MSG	{_POSIX_MQ_OPEN_MAX}	
9130		The number of message queues that can be open for a single process.	
9131		Value: 8	
9132	MSG	{_POSIX_MQ_PRIO_MAX}	
9133		The maximum number of message priorities supported by the implementation.	
9134		Value: 32	
9135		{_POSIX_NAME_MAX}	
9136		Maximum number of bytes in a filename (not including the terminating null of a filename string).	
9137		Value: 14	
9138			
9139		{_POSIX_NGROUPS_MAX}	
9140		Maximum number of simultaneous supplementary group IDs per process.	
9141		Value: 8	
9142		{_POSIX_OPEN_MAX}	
9143		A value one greater than the maximum value that the system may assign to a newly-created file descriptor.	
9144		Value: 20	
9145			
9146		{_POSIX_PATH_MAX}	
9147		Minimum number the implementation will accept as the maximum number of bytes in a pathname.	
9148		Value: 256	
9149			
9150		{_POSIX_PIPE_BUF}	
9151		Maximum number of bytes that is guaranteed to be atomic when writing to a pipe.	
9152		Value: 512	
9153		{_POSIX_RE_DUP_MAX}	
9154		The number of repeated occurrences of a BRE permitted by the <i>regexec()</i> and <i>regcomp()</i> functions when using the interval notation $\{ \backslash(m,n\backslash \}$; see Section 9.3.6 (on page 186).	
9155		Value: 255	
9156			
9157		{_POSIX_RTSIG_MAX}	
9158		The number of realtime signal numbers reserved for application use.	
9159		Value: 8	
9160		{_POSIX_SEM_NSEMS_MAX}	
9161		The number of semaphores that a process may have.	
9162		Value: 256	
9163		{_POSIX_SEM_VALUE_MAX}	
9164		The maximum value a semaphore may have.	
9165		Value: 32 767	

9166		{_POSIX_SIGQUEUE_MAX}
9167		The number of queued signals that a process may send and have pending at the receiver(s)
9168		at any time.
9169		Value: 32
9170		{_POSIX_SSIZE_MAX}
9171		The value that can be stored in an object of type ssize_t .
9172		Value: 32 767
9173	SS TSP	{_POSIX_SS_REPL_MAX}
9174		The number of replenishment operations that may be simultaneously pending for a
9175		particular sporadic server scheduler.
9176		Value: 4
9177		{_POSIX_STREAM_MAX}
9178		The number of streams that one process can have open at one time.
9179		Value: 8
9180		{_POSIX_SYMLINK_MAX}
9181		The number of bytes in a symbolic link.
9182		Value: 255
9183		{_POSIX_SYMLINK_MAX}
9184		The number of symbolic links that can be traversed in the resolution of a pathname in the
9185		absence of a loop.
9186		Value: 8
9187		{_POSIX_THREAD_DESTRUCTOR_ITERATIONS}
9188		The number of attempts made to destroy a thread's thread-specific data values on thread
9189		exit.
9190		Value: 4
9191		{_POSIX_THREAD_KEYS_MAX}
9192		The number of data keys per process.
9193		Value: 128
9194		{_POSIX_THREAD_THREADS_MAX}
9195		The number of threads per process.
9196		Value: 64
9197		{_POSIX_TIMER_MAX}
9198		The per-process number of timers.
9199		Value: 32
9200	OB TRC	{_POSIX_TRACE_EVENT_NAME_MAX}
9201		The length in bytes of a trace event name (not including the terminating null).
9202		Value: 30
9203	OB TRC	{_POSIX_TRACE_NAME_MAX}
9204		The length in bytes of a trace generation version string or a trace stream name (not
9205		including the terminating null).
9206		Value: 8
9207	OB TRC	{_POSIX_TRACE_SYS_MAX}
9208		The number of trace streams that may simultaneously exist in the system.
9209		Value: 8

9210 OB TRC `{_POSIX_TRACE_USER_EVENT_MAX}`
 9211 The number of user trace event type identifiers that may simultaneously exist in a traced
 9212 process, including the predefined user trace event
 9213 `POSIX_TRACE_UNNAMED_USER_EVENT`.
 9214 Value: 32

9215 `{_POSIX_TTY_NAME_MAX}`
 9216 The size of the storage required for a terminal device name, in bytes (including the
 9217 terminating null).
 9218 Value: 9

9219 `{_POSIX_TZNAME_MAX}`
 9220 Maximum number of bytes supported for the name of a timezone (not of the *TZ* variable).
 9221 Value: 6

9222 **Note:** The length given by `{_POSIX_TZNAME_MAX}` does not include the quoting characters
 9223 mentioned in [Section 8.3](#) (on page 177).

9224 `{_POSIX2_BC_BASE_MAX}`
 9225 Maximum *obase* values allowed by the *bc* utility.
 9226 Value: 99

9227 `{_POSIX2_BC_DIM_MAX}`
 9228 Maximum number of elements permitted in an array by the *bc* utility.
 9229 Value: 2 048

9230 `{_POSIX2_BC_SCALE_MAX}`
 9231 Maximum *scale* value allowed by the *bc* utility.
 9232 Value: 99

9233 `{_POSIX2_BC_STRING_MAX}`
 9234 Maximum length of a string constant accepted by the *bc* utility.
 9235 Value: 1 000

9236 `{_POSIX2_CHARCLASS_NAME_MAX}`
 9237 Maximum number of bytes in a character class name.
 9238 Value: 14

9239 `{_POSIX2_COLL_WEIGHTS_MAX}`
 9240 Maximum number of weights that can be assigned to an entry of the *LC_COLLATE* **order**
 9241 keyword in the locale definition file; see [Chapter 7](#) (on page 135).
 9242 Value: 2

9243 `{_POSIX2_EXPR_NEST_MAX}`
 9244 Maximum number of expressions that can be nested within parentheses by the *expr* utility.
 9245 Value: 32

9246 `{_POSIX2_LINE_MAX}`
 9247 Unless otherwise noted, the maximum length, in bytes, of a utility's input line (either
 9248 standard input or another file), when the utility is described as processing text files. The
 9249 length includes room for the trailing <newline>.
 9250 Value: 2 048

9251 `{_POSIX2_RE_DUP_MAX}`
 9252 Maximum number of repeated occurrences of a regular expression permitted when using
 9253 the interval notation `\{m,n\}`; see [Chapter 9](#) (on page 181).
 9254 Value: 255

9255 XSI **{_XOPEN_IOV_MAX}**
 9256 Maximum number of **iovec** structures that one process has available for use with *readv()* or
 9257 *writev()*.
 9258 Value: 16

9259 XSI **{_XOPEN_NAME_MAX}**
 9260 Maximum number of bytes in a filename (not including the terminating null of a filename
 9261 string).
 9262 Value: 255

9263 XSI **{_XOPEN_PATH_MAX}**
 9264 Minimum number the implementation will accept as the maximum number of bytes in a
 9265 pathname.
 9266 Value: 1 024

9267 Numerical Limits

9268 The <limits.h> header shall define the following macros and, except for {CHAR_BIT},
 9269 {LONG_BIT}, {MB_LEN_MAX}, and {WORD_BIT}, they shall be replaced by expressions that
 9270 have the same type as would an expression that is an object of the corresponding type converted
 9271 according to the integer promotions.

9272 If the value of an object of type **char** is treated as a signed integer when used in an expression,
 9273 the value of {CHAR_MIN} is the same as that of {SCHAR_MIN} and the value of {CHAR_MAX}
 9274 is the same as that of {SCHAR_MAX}. Otherwise, the value of {CHAR_MIN} is 0 and the value
 9275 of {CHAR_MAX} is the same as that of {UCHAR_MAX}.

9276 {CHAR_BIT}
 9277 Number of bits in a type **char**.
 9278 CX Value: 8

9279 {CHAR_MAX}
 9280 Maximum value for an object of type **char**.
 9281 Value: {UCHAR_MAX} or {SCHAR_MAX}

9282 {CHAR_MIN}
 9283 Minimum value for an object of type **char**.
 9284 Value: {SCHAR_MIN} or 0

9285 {INT_MAX}
 9286 Maximum value for an object of type **int**.
 9287 CX Minimum Acceptable Value: 2 147 483 647

9288 {INT_MIN}
 9289 Minimum value for an object of type **int**.
 9290 CX Maximum Acceptable Value: -2 147 483 647

9291 {LLONG_MAX}
 9292 Maximum value for an object of type **long long**.
 9293 Minimum Acceptable Value: +9 223 372 036 854 775 807

9294 {LLONG_MIN}
 9295 Minimum value for an object of type **long long**.
 9296 Maximum Acceptable Value: -9 223 372 036 854 775 807

9297 CX **{LONG_BIT}**
 9298 Number of bits in an object of type **long**.
 9299 Minimum Acceptable Value: 32

9300		{LONG_MAX}
9301		Maximum value for an object of type long .
9302		Minimum Acceptable Value: +2 147 483 647
9303		{LONG_MIN}
9304		Minimum value for an object of type long .
9305		Maximum Acceptable Value: -2 147 483 647
9306		{MB_LEN_MAX}
9307		Maximum number of bytes in a character, for any supported locale.
9308		Minimum Acceptable Value: 1
9309		{SCHAR_MAX}
9310		Maximum value for an object of type signed char .
9311	CX	Value: +127
9312		{SCHAR_MIN}
9313		Minimum value for an object of type signed char .
9314	CX	Value: -128
9315		{SHRT_MAX}
9316		Maximum value for an object of type short .
9317		Minimum Acceptable Value: +32 767
9318		{SHRT_MIN}
9319		Minimum value for an object of type short .
9320		Maximum Acceptable Value: -32 767
9321	CX	{SSIZE_MAX}
9322		Maximum value for an object of type ssize_t .
9323		Minimum Acceptable Value: {_POSIX_SSIZE_MAX}
9324		{UCHAR_MAX}
9325		Maximum value for an object of type unsigned char .
9326	CX	Value: 255
9327		{UINT_MAX}
9328		Maximum value for an object of type unsigned .
9329	CX	Minimum Acceptable Value: 4 294 967 295
9330		{ULLONG_MAX}
9331		Maximum value for an object of type unsigned long long .
9332		Minimum Acceptable Value: 18 446 744 073 709 551 615
9333		{ULONG_MAX}
9334		Maximum value for an object of type unsigned long .
9335		Minimum Acceptable Value: 4 294 967 295
9336		{USHRT_MAX}
9337		Maximum value for an object of type unsigned short .
9338		Minimum Acceptable Value: 65 535
9339	CX	{WORD_BIT}
9340		Number of bits in an object of type int .
9341		Minimum Acceptable Value: 32

Other Invariant Values

The <limits.h> header shall define the following symbolic constants:

{NL_ARGMAX}

Maximum value of *n* in conversion specifications using the "%n\$" sequence in calls to the *printf()* and *scanf()* families of functions.

Minimum Acceptable Value: 9

{NL_LANGMAX}

Maximum number of bytes in a *LANG* name.

Minimum Acceptable Value: 14

{NL_MSGMAX}

Maximum message number.

Minimum Acceptable Value: 32 767

{NL_SETMAX}

Maximum set number.

Minimum Acceptable Value: 255

{NL_TEXTMAX}

Maximum number of bytes in a message string.

Minimum Acceptable Value: {_POSIX2_LINE_MAX}

{NZERO}

Default process priority.

Minimum Acceptable Value: 20

APPLICATION USAGE

None.

RATIONALE

A request was made to reduce the value of {_POSIX_LINK_MAX} from the value of 8 specified for it in the POSIX.1-1990 standard to 2. The standard developers decided to deny this request for several reasons:

- They wanted to avoid making any changes to the standard that could break conforming applications, and the requested change could have that effect.
- The use of multiple hard links to a file cannot always be replaced with use of symbolic links. Symbolic links are semantically different from hard links in that they associate a pathname with another pathname rather than a pathname with a file. This has implications for access control, file permanence, and transparency.
- The original standard developers had considered the issue of allowing for implementations that did not in general support hard links, and decided that this would reduce consensus on the standard.

Systems that support historical versions of the development option of the ISO POSIX-2 standard retain the name {_POSIX2_RE_DUP_MAX} as an alias for {_POSIX_RE_DUP_MAX}.

{PATH_MAX}

IEEE PASC Interpretation 1003.1 #15 addressed the inconsistency in the standard with the definition of pathname and the description of {PATH_MAX}, allowing application developers to allocate either {PATH_MAX} or {PATH_MAX}+1 bytes. The inconsistency has been removed by correction to the {PATH_MAX} definition to include the null character. With this change, applications that previously allocated {PATH_MAX} bytes will continue to succeed.

9387 {SYMLINK_MAX}
 9388 This symbol refers to space for data that is stored in the file system, as opposed to
 9389 {PATH_MAX} which is the length of a name that can be passed to a function. In some
 9390 existing implementations, the pathnames pointed to by symbolic links are stored in the
 9391 *inodes* of the links, so it is important that {SYMLINK_MAX} not be constrained to be as large
 9392 as {PATH_MAX}.

9393 FUTURE DIRECTIONS

9394 None.

9395 SEE ALSO

9396 Chapter 7 (on page 135), <stdio.h>, <unistd.h>
 9397 XSH Section 2.2 (on page 472), *fpathconf()*, *sysconf()*

9398 CHANGE HISTORY

9399 First released in Issue 1.

9400 Issue 5

9401 The DESCRIPTION is updated for alignment with the POSIX Realtime Extension and the POSIX
 9402 Threads Extension.

9403 {FILESIZEBITS} is added for the Large File Summit extensions.

9404 The minimum acceptable values for {INT_MAX}, {INT_MIN}, and {UINT_MAX} are changed to
 9405 make 32-bit values the minimum requirement.

9406 The entry is restructured to improve readability.

9407 Issue 6

9408 The Open Group Corrigendum U033/4 is applied. The wording is made clear for {CHAR_MIN},
 9409 {INT_MIN}, {LONG_MIN}, {SCHAR_MIN}, and {SHRT_MIN} that these are maximum
 9410 acceptable values.

9411 The following new requirements on POSIX implementations derive from alignment with the
 9412 Single UNIX Specification:

- 9413 • The minimum value for {CHILD_MAX} is 25. This is a FIPS requirement.
- 9414 • The minimum value for {OPEN_MAX} is 20. This is a FIPS requirement.
- 9415 • The minimum value for {NGROUPS_MAX} is 8. This is also a FIPS requirement.

9416 Symbolic constants are added for {_POSIX_SYMLINK_MAX}, {_POSIX_SYMLOOP_MAX},
 9417 {_POSIX_RE_DUP_MAX}, {RE_DUP_MAX}, {SYMLOOP_MAX}, and {SYMLINK_MAX}.

9418 The following values are added for alignment with IEEE Std 1003.1d-1999:

9419 {_POSIX_SS_REPL_MAX}
 9420 {SS_REPL_MAX}
 9421 {POSIX_ALLOC_SIZE_MIN}
 9422 {POSIX_REC_INCR_XFER_SIZE}
 9423 {POSIX_REC_MAX_XFER_SIZE}
 9424 {POSIX_REC_MIN_XFER_SIZE}
 9425 {POSIX_REC_XFER_ALIGN}

9426 Reference to CLOCK_MONOTONIC is added in the description of {_POSIX_CLOCKRES_MIN}
 9427 for alignment with IEEE Std 1003.1j-2000.

9428 The constants {LLONG_MIN}, {LLONG_MAX}, and {ULLONG_MAX} are added for alignment

with the ISO/IEC 9899: 1999 standard.

The following values are added for alignment with IEEE Std 1003.1q-2000:

```
{_POSIX_TRACE_EVENT_NAME_MAX}
{TRACE_EVENT_NAME_MAX}
{TRACE_NAME_MAX}
{TRACE_SYS_MAX}
{TRACE_USER_EVENT_MAX}
```

The new limits {_XOPEN_NAME_MAX} and {_XOPEN_PATH_MAX} are added as minimum values for {PATH_MAX} and {NAME_MAX} limits on XSI-conformant systems.

The LEGACY symbols {PASS_MAX} and {TMP_MAX} are removed.

The values for the limits {CHAR_BIT}, {SCHAR_MAX}, and {UCHAR_MAX} are now required to be 8, +127, and 255, respectively.

The value for the limit {CHAR_MAX} is now {UCHAR_MAX} or {SCHAR_MAX}.

The value for the limit {CHAR_MIN} is now {SCHAR_MIN} or zero.

IEEE Std 1003.1-2001/Cor 1-2002, item XBD/TC1/D6/10 is applied, correcting the value of {_POSIX_CHILD_MAX} from 6 to 25. This is for FIPS 151-2 alignment.

IEEE Std 1003.1-2001/Cor 2-2004, item XBD/TC2/D6/19 is applied, updating the values for {INT_MAX}, {UINT_MAX}, and {INT_MIN} to be CX extensions over the ISO C standard, and correcting {WORD_BIT} from 16 to 32.

IEEE Std 1003.1-2001/Cor 2-2004, item XBD/TC2/D6/20 is applied, removing {CHARCLASS_NAME_MAX} from the “Other Invariant Values” section (it also occurs under “Runtime Inceasable Values”).

Issue 7

Austin Group Interpretations 1003.1-2001 #143 and #160 are applied.

Austin Group Interpretation 1003.1-2001 #173 is applied, updating the descriptions of {TRACE_EVENT_NAME_MAX} and {TRACE_NAME_MAX} to not include the terminating null.

SD5-XBD-ERN-36 is applied, changing the description of {RE_DUP_MAX}.

SD5-XBD-ERN-90 is applied.

{NL_NMAX} is removed; it should have been removed in Issue 6.

The Trace option values are marked obsolescent.

The {ATEXIT_MAX}, {LONG_BIT}, {NL_MSGMAX}, {NL_SETMAX}, {NL_TEXTMAX}, and {WORD_BIT} values are moved from the XSI option to the Base.

The AIO_* and _POSIX_AIO_* values are moved from the Asynchronous Input and Output option to the Base.

The {_POSIX_RT_SIG_MAX}, {_POSIX_SIGQUEUE_MAX}, {RTSIG_MAX}, and {SIGQUEUE_MAX} values are moved from the Realtime Signals Extension option to the Base.

Functionality relating to the Threads and Timers options is moved to the Base.

9470 This reference page is clarified with respect to macros and symbolic constants.
 9471 POSIX.1-2008, Technical Corrigendum 1, XBD/TC1-2008/0052 [108], XBD/TC1-2008/0053 [291], +
 9472 XBD/TC1-2008/0054 [182,427], XBD/TC1-2008/0055 [291], XBD/TC1-2008/0056 [371], +
 9473 XBD/TC1-2008/0057 [291], XBD/TC1-2008/0058 [108], and XBD/TC1-2008/0059 [291] are +
 9474 applied.

9475 **NAME**

9476 locale.h — category macros

9477 **SYNOPSIS**

9478 #include <locale.h>

9479 **DESCRIPTION**

9480 CX Some of the functionality described on this reference page extends the ISO C standard.
 9481 Applications shall define the appropriate feature test macro (see XSH [Section 2.2](#), on page 472) to
 9482 enable the visibility of these symbols in this header.

9483 The <locale.h> header shall define the **lconv** structure, which shall include at least the following
 9484 members. (See the definitions of *LC_MONETARY* in [Section 7.3.3](#) (on page 154) and [Section 7.3.4](#)
 9485 (on page 157).)

```

9486 char    *currency_symbol
9487 char    *decimal_point
9488 char    frac_digits
9489 char    *grouping
9490 char    *int_curr_symbol
9491 char    int_frac_digits
9492 char    int_n_cs_precedes
9493 char    int_n_sep_by_space
9494 char    int_n_sign_posn
9495 char    int_p_cs_precedes
9496 char    int_p_sep_by_space
9497 char    int_p_sign_posn
9498 char    *mon_decimal_point
9499 char    *mon_grouping
9500 char    *mon_thousands_sep
9501 char    *negative_sign
9502 char    n_cs_precedes
9503 char    n_sep_by_space
9504 char    n_sign_posn
9505 char    *positive_sign
9506 char    p_cs_precedes
9507 char    p_sep_by_space
9508 char    p_sign_posn
9509 char    *thousands_sep

```

9510 The <locale.h> header shall define NULL (as described in <stddef.h>) and at least the following
 9511 as macros:

```

9512 LC_ALL
9513 LC_COLLATE
9514 LC_CTYPE
9515 CX LC_MESSAGES
9516 LC_MONETARY
9517 LC_NUMERIC
9518 LC_TIME

```

9519 which shall expand to integer constant expressions with distinct values for use as the first
 9520 argument to the *setlocale()* function.

9521 Implementations may add additional masks using the form *LC_** and an uppercase letter.

9522 CX The <locale.h> header shall contain at least the following macros representing bitmasks for use
 9523 with the *newlocale()* function for each supported locale category:

9524 LC_COLLATE_MASK
 9525 LC_CTYPE_MASK
 9526 LC_MESSAGES_MASK
 9527 LC_MONETARY_MASK
 9528 LC_NUMERIC_MASK
 9529 LC_TIME_MASK

9530 Implementations may add additional masks using the form LC_*_MASK.
 9531 In addition, a macro to set the bits for all categories set shall be defined:
 9532 LC_ALL_MASK

9533 The <locale.h> header shall define LC_GLOBAL_LOCALE, a special locale object descriptor
 9534 used by the *duplocale()* and *uselocale()* functions.

9535 The <locale.h> header shall define the **locale_t** type, representing a locale object.

9536 The following shall be declared as functions and may also be defined as macros. Function
 9537 prototypes shall be provided for use with ISO C standard compilers.

9538 CX `locale_t duplocale(locale_t);`
 9539 `void freelocale(locale_t);`
 9540 `struct lconv *localeconv(void);`
 9541 CX `locale_t newlocale(int, const char *, locale_t);`
 9542 `char *setlocale(int, const char *);`
 9543 CX `locale_t uselocale(locale_t);`

9544 APPLICATION USAGE

9545 None.

9546 RATIONALE

9547 None.

9548 FUTURE DIRECTIONS

9549 None.

9550 SEE ALSO

9551 [Chapter 8](#) (on page 173), [<stddef.h>](#)

9552 XSH [duplocale\(\)](#), [freelocale\(\)](#), [localeconv\(\)](#), [newlocale\(\)](#), [setlocale\(\)](#), [uselocale\(\)](#)

9553 CHANGE HISTORY

9554 First released in Issue 3.

9555 Included for alignment with the ISO C standard.

9556 Issue 6

9557 The **lconv** structure is expanded with new members (**int_n_cs_precedes**, **int_n_sep_by_space**,
 9558 **int_n_sign_posn**, **int_p_cs_precedes**, **int_p_sep_by_space**, and **int_p_sign_posn**) for alignment
 9559 with the ISO/IEC 9899:1999 standard.

9560 Extensions beyond the ISO C standard are marked.

Issue 7

The *duplocale()*, *freelocale()*, *newlocale()*, and *uselocale()* functions are added from The Open Group Technical Standard, 2006, Extended API Set Part 4.

This reference page is clarified with respect to macros and symbolic constants.

POSIX.1-2008, Technical Corrigendum 1, XBD/TC1-2008/0060 [301,427] is applied.

+

9566 **NAME**

9567 **math.h** — mathematical declarations

9568 **SYNOPSIS**

9569 `#include <math.h>`

9570 **DESCRIPTION**

9571 CX Some of the functionality described on this reference page extends the ISO C standard.
9572 Applications shall define the appropriate feature test macro (see XSH [Section 2.2](#), on page 472) to
9573 enable the visibility of these symbols in this header.

9574 The <math.h> header shall define at least the following types:

9575 **float_t** A real-floating type at least as wide as **float**.

9576 **double_t** A real-floating type at least as wide as **double**, and at least as wide as **float_t**.

9577 If FLT_EVAL_METHOD equals 0, **float_t** and **double_t** shall be **float** and **double**, respectively; if
9578 FLT_EVAL_METHOD equals 1, they shall both be **double**; if FLT_EVAL_METHOD equals 2,
9579 they shall both be **long double**; for other values of FLT_EVAL_METHOD, they are otherwise
9580 implementation-defined.

9581 The <math.h> header shall define the following macros, where real-floating indicates that the
9582 argument shall be an expression of real-floating type:

```
9583 int fpclassify(real-floating x);
9584 int isfinite(real-floating x);
9585 int isgreater(real-floating x, real-floating y);
9586 int isgreaterequal(real-floating x, real-floating y);
9587 int isinf(real-floating x);
9588 int isless(real-floating x, real-floating y);
9589 int islessequal(real-floating x, real-floating y);
9590 int islessgreater(real-floating x, real-floating y);
9591 int isnan(real-floating x);
9592 int isnormal(real-floating x);
9593 int isunordered(real-floating x, real-floating y);
9594 int signbit(real-floating x);
```

9595 The <math.h> header shall define the following symbolic constants. The values shall have type
9596 **double** and shall be accurate within the precision of the **double** type.

9597	XSI	M_E	Value of e
9598	XSI	M_LOG2E	Value of $\log_2 e$
9599	XSI	M_LOG10E	Value of $\log_{10} e$
9600	XSI	M_LN2	Value of $\log_e 2$
9601	XSI	M_LN10	Value of $\log_e 10$
9602	XSI	M_PI	Value of π
9603	XSI	M_PI_2	Value of $\pi/2$
9604	XSI	M_PI_4	Value of $\pi/4$
9605	XSI	M_1_PI	Value of $1/\pi$
9606	XSI	M_2_PI	Value of $2/\pi$

9607	XSI	M_2_SQRTPI	Value of $2/\sqrt{\pi}$
9608	XSI	M_SQRT2	Value of $\sqrt{2}$
9609	XSI	M_SQRT1_2	Value of $1/\sqrt{2}$
9610		The <math.h> header shall define the following symbolic constant:	
9611	OB XSI	MAXFLOAT	Same value as FLT_MAX in <float.h>.
9612		The <math.h> header shall define the following macros:	
9613		HUGE_VAL	A positive double constant expression, not necessarily representable as a
9614			float . Used as an error value returned by the mathematics library.
9615			HUGE_VAL evaluates to +infinity on systems supporting IEEE Std 754-1985.
9616		HUGE_VALF	A positive float constant expression. Used as an error value returned by the
9617			mathematics library. HUGE_VALF evaluates to +infinity on systems
9618			supporting IEEE Std 754-1985.
9619		HUGE_VALL	A positive long double constant expression. Used as an error value returned
9620			by the mathematics library. HUGE_VALL evaluates to +infinity on systems
9621			supporting IEEE Std 754-1985.
9622		INFINITY	A constant expression of type float representing positive or unsigned infinity,
9623			if available; else a positive constant of type float that overflows at translation
9624			time.
9625		NAN	A constant expression of type float representing a quiet NaN. This macro is
9626			only defined if the implementation supports quiet NaNs for the float type.
9627		The following macros shall be defined for number classification. They represent the mutually-	
9628		exclusive kinds of floating-point values. They expand to integer constant expressions with	
9629		distinct values. Additional implementation-defined floating-point classifications, with macro	
9630		definitions beginning with FP_ and an uppercase letter, may also be specified by the	
9631		implementation.	
9632		FP_INFINITE	
9633		FP_NAN	
9634		FP_NORMAL	
9635		FP_SUBNORMAL	
9636		FP_ZERO	
9637		The following optional macros indicate whether the <i>fma()</i> family of functions are fast compared	
9638		with direct code:	
9639		FP_FAST_FMA	
9640		FP_FAST_FMAF	
9641		FP_FAST_FMAL	
9642		If defined, the FP_FAST_FMA macro shall expand to the integer constant 1 and shall indicate	
9643		that the <i>fma()</i> function generally executes about as fast as, or faster than, a multiply and an add	
9644		of double operands. If undefined, the speed of execution is unspecified. The other macros have	
9645		the equivalent meaning for the float and long double versions.	
9646		The following macros shall expand to integer constant expressions whose values are returned by	
9647		<i>ilogb(x)</i> if <i>x</i> is zero or NaN, respectively. The value of FP_ILOGB0 shall be either {INT_MIN} or	
9648		-{INT_MAX}. The value of FP_ILOGBNAN shall be either {INT_MAX} or {INT_MIN}.	

9649 FP_ILOGB0
9650 FP_ILOGBNAN

9651 The following macros shall expand to the integer constants 1 and 2, respectively;

9652 MATH_ERRNO
9653 MATH_ERREXCEPT

9654 The following macro shall expand to an expression that has type **int** and the value
9655 MATH_ERRNO, MATH_ERREXCEPT, or the bitwise-inclusive OR of both:

9656 math_errhandling

9657 The value of math_errhandling is constant for the duration of the program. It is unspecified
9658 whether math_errhandling is a macro or an identifier with external linkage. If a macro definition
9659 is suppressed or a program defines an identifier with the name math_errhandling , the behavior
9660 is undefined. If the expression (math_errhandling & MATH_ERREXCEPT) can be non-zero, the
9661 implementation shall define the macros FE_DIVBYZERO, FE_INVALID, and FE_OVERFLOW in
9662 <fenv.h>.

9663 The following shall be declared as functions and may also be defined as macros. Function
9664 prototypes shall be provided.

9665 double acos(double);
9666 float acosf(float);
9667 double acosh(double);
9668 float acoshf(float);
9669 long double acoshl(long double);
9670 long double acosl(long double);
9671 double asin(double);
9672 float asinf(float);
9673 double asinh(double);
9674 float asinhf(float);
9675 long double asinhl(long double);
9676 long double asinl(long double);
9677 double atan(double);
9678 double atan2(double, double);
9679 float atan2f(float, float);
9680 long double atan2l(long double, long double);
9681 float atanf(float);
9682 double atanh(double);
9683 float atanhf(float);
9684 long double atanh1(long double);
9685 long double atan1(long double);
9686 double cbrt(double);
9687 float cbrtf(float);
9688 long double cbrtl(long double);
9689 double ceil(double);
9690 float ceilf(float);
9691 long double ceill(long double);
9692 double copysign(double, double);
9693 float copysignf(float, float);
9694 long double copysignl(long double, long double);
9695 double cos(double);

```

9696     float      cosf(float);
9697     double     cosh(double);
9698     float      coshf(float);
9699     long double coshl(long double);
9700     long double cosl(long double);
9701     double     erf(double);
9702     double     erfc(double);
9703     float      erfcf(float);
9704     long double erfcl(long double);
9705     float      erff(float);
9706     long double erfl(long double);
9707     double     exp(double);
9708     double     exp2(double);
9709     float      exp2f(float);
9710     long double exp2l(long double);
9711     float      expf(float);
9712     long double expl(long double);
9713     double     expm1(double);
9714     float      expm1f(float);
9715     long double expm1l(long double);
9716     double     fabs(double);
9717     float      fabsf(float);
9718     long double fabsl(long double);
9719     double     fdim(double, double);
9720     float      fdimf(float, float);
9721     long double fdiml(long double, long double);
9722     double     floor(double);
9723     float      floorf(float);
9724     long double floorl(long double);
9725     double     fma(double, double, double);
9726     float      fmaf(float, float, float);
9727     long double fmal(long double, long double, long double);
9728     double     fmax(double, double);
9729     float      fmaxf(float, float);
9730     long double fmaxl(long double, long double);
9731     double     fmin(double, double);
9732     float      fminf(float, float);
9733     long double fminl(long double, long double);
9734     double     fmod(double, double);
9735     float      fmodf(float, float);
9736     long double fmodl(long double, long double);
9737     double     frexp(double, int *);
9738     float      frexpf(float, int *);
9739     long double frexpl(long double, int *);
9740     double     hypot(double, double);
9741     float      hypotf(float, float);
9742     long double hypotl(long double, long double);
9743     int        ilogb(double);
9744     int        ilogbf(float);
9745     int        ilogbl(long double);
9746     XSI double   j0(double);
9747     double     j1(double);

```

```

9748 double      jn(int, double);
9749 double      ldexp(double, int);
9750 float       ldexpf(float, int);
9751 long double ldexpl(long double, int);
9752 double      lgamma(double);
9753 float       lgammaf(float);
9754 long double lgammal(long double);
9755 long long   llrint(double);
9756 long long   llrintf(float);
9757 long long   llrintl(long double);
9758 long long   llround(double);
9759 long long   llroundf(float);
9760 long long   llroundl(long double);
9761 double      log(double);
9762 double      log10(double);
9763 float       log10f(float);
9764 long double log10l(long double);
9765 double      log1p(double);
9766 float       log1pf(float);
9767 long double log1pl(long double);
9768 double      log2(double);
9769 float       log2f(float);
9770 long double log2l(long double);
9771 double      logb(double);
9772 float       logbf(float);
9773 long double logbl(long double);
9774 float       logf(float);
9775 long double logl(long double);
9776 long        lrint(double);
9777 long        lrintf(float);
9778 long        lrintl(long double);
9779 long        lround(double);
9780 long        lroundf(float);
9781 long        lroundl(long double);
9782 double      modf(double, double *);
9783 float       modff(float, float *);
9784 long double modfl(long double, long double *);
9785 double      nan(const char *);
9786 float       nanf(const char *);
9787 long double nanl(const char *);
9788 double      nearbyint(double);
9789 float       nearbyintf(float);
9790 long double nearbyintl(long double);
9791 double      nextafter(double, double);
9792 float       nextafterf(float, float);
9793 long double nextafterl(long double, long double);
9794 double      nexttoward(double, long double);
9795 float       nexttowardf(float, long double);
9796 long double nexttowardl(long double, long double);
9797 double      pow(double, double);
9798 float       powf(float, float);
9799 long double powl(long double, long double);

```

```

9800     double      remainder(double, double);
9801     float        remainderf(float, float);
9802     long double  remainderl(long double, long double);
9803     double      remquo(double, double, int *);
9804     float        remquof(float, float, int *);
9805     long double  remquol(long double, long double, int *);
9806     double      rint(double);
9807     float        rintf(float);
9808     long double  rintl(long double);
9809     double      round(double);
9810     float        roundf(float);
9811     long double  roundl(long double);
9812     double      scalbln(double, long);
9813     float        scalblnf(float, long);
9814     long double  scalblnl(long double, long);
9815     double      scalbn(double, int);
9816     float        scalbnf(float, int);
9817     long double  scalbnl(long double, int);
9818     double      sin(double);
9819     float        sinf(float);
9820     double      sinh(double);
9821     float        sinhlf(float);
9822     long double sinhl(long double);
9823     long double sinl(long double);
9824     double      sqrt(double);
9825     float        sqrtf(float);
9826     long double sqrtl(long double);
9827     double      tan(double);
9828     float        tanf(float);
9829     double      tanh(double);
9830     float        tanhlf(float);
9831     long double tanhl(long double);
9832     long double tanl(long double);
9833     double      tgamma(double);
9834     float        tgammaf(float);
9835     long double tgamma1(long double);
9836     double      trunc(double);
9837     float        truncf(float);
9838     long double trunc1(long double);
9839     XSI double    y0(double);
9840     double      y1(double);
9841     double      yn(int, double);

```

9842 The following external variable shall be defined:

```

9843 XSI extern int signgam;

```

9844 The behavior of each of the functions defined in <math.h> is specified in the System Interfaces
9845 volume of POSIX.1-2008 for all representable values of its input arguments, except where stated
9846 otherwise. Each function shall execute as if it were a single operation without generating any
9847 externally visible exceptional conditions.

APPLICATION USAGE

The FP_CONTRACT pragma can be used to allow (if the state is on) or disallow (if the state is off) the implementation to contract expressions. Each pragma can occur either outside external declarations or preceding all explicit declarations and statements inside a compound statement. When outside external declarations, the pragma takes effect from its occurrence until another FP_CONTRACT pragma is encountered, or until the end of the translation unit. When inside a compound statement, the pragma takes effect from its occurrence until another FP_CONTRACT pragma is encountered (including within a nested compound statement), or until the end of the compound statement; at the end of a compound statement the state for the pragma is restored to its condition just before the compound statement. If this pragma is used in any other context, the behavior is undefined. The default state (on or off) for the pragma is implementation-defined.

Applications should use FLT_MAX as described in the <float.h> header instead of the obsolescent MAXFLOAT.

RATIONALE

Before the ISO/IEC 9899:1999 standard, the math library was defined only for the floating type **double**. All the names formed by appending 'f' or 'l' to a name in <math.h> were reserved to allow for the definition of **float** and **long double** libraries; and the ISO/IEC 9899:1999 standard provides for all three versions of math functions.

The functions *ecvt()*, *fcvt()*, and *gcvt()* have been dropped from the ISO C standard since their capability is available through *sprintf()*.

FUTURE DIRECTIONS

None.

SEE ALSO

<float.h>, <stddef.h>, <sys/types.h>

XSH Section 2.2 (on page 472), *acos()*, *acosh()*, *asin()*, *asinh()*, *atan()*, *atan2()*, *atanh()*, *cbrt()*, *ceil()*, *copysign()*, *cos()*, *cosh()*, *erf()*, *erfc()*, *exp()*, *exp2()*, *expm1()*, *fabs()*, *fdim()*, *floor()*, *fma()*, *fmax()*, *fmin()*, *fmod()*, *fpclassify()*, *frexp()*, *hypot()*, *ilogb()*, *isfinite()*, *isgreater()*, *isgreaterequal()*, *isinf()*, *isless()*, *islessequal()*, *islessgreater()*, *isnan()*, *isnormal()*, *isunordered()*, *j0()*, *ldexp()*, *lgamma()*, *llrint()*, *llround()*, *log()*, *log10()*, *log1p()*, *log2()*, *logb()*, *lrint()*, *lround()*, *modf()*, *nan()*, *nearbyint()*, *nextafter()*, *pow()*, *remainder()*, *remquo()*, *rint()*, *round()*, *scalbln()*, *signbit()*, *sin()*, *sinh()*, *sqrt()*, *tan()*, *tanh()*, *tgamma()*, *trunc()*, *y0()*

CHANGE HISTORY

First released in Issue 1.

Issue 6

This reference page is updated to align with the ISO/IEC 9899:1999 standard.

IEEE Std 1003.1-2001/Cor 2-2004, item XBD/TC2/D6/21 is applied, making it clear that the meaning of the FP_FAST_FMA macro is unspecified if the macro is undefined.

Issue 7

ISO/IEC 9899:1999 standard, Technical Corrigendum 2 #47 (SD5-XBD-ERN-52) is applied, clarifying the wording of the FP_FAST_FMA macro.

The MAXFLOAT constant is marked obsolescent.

This reference page is clarified with respect to macros and symbolic constants.

9890 **NAME**

9891 monetary.h — monetary types

9892 **SYNOPSIS**

9893 #include <monetary.h>

9894 **DESCRIPTION**9895 The <monetary.h> header shall define the **locale_t** type as described in <locale.h>.9896 The <monetary.h> header shall define the **size_t** type as described in <stddef.h>.9897 The <monetary.h> header shall define the **ssize_t** type as described in <sys/types.h>.9898 The following shall be declared as functions and may also be defined as macros. Function
9899 prototypes shall be provided for use with ISO C standard compilers.

9900 ssize_t strfmon(char *restrict, size_t, const char *restrict, ...);

9901 ssize_t strfmon_l(char *restrict, size_t, locale_t,

9902 const char *restrict, ...);

9903 **APPLICATION USAGE**

9904 None.

9905 **RATIONALE**

9906 None.

9907 **FUTURE DIRECTIONS**

9908 None.

9909 **SEE ALSO**

9910 <locale.h>, <stddef.h>, <sys/types.h>

9911 XSH *strfmon()*9912 **CHANGE HISTORY**

9913 First released in Issue 4.

9914 **Issue 6**9915 The **restrict** keyword is added to the prototype for *strfmon()*.9916 **Issue 7**

9917 The <monetary.h> header is moved from the XSI option to the Base.

9918 The *strfmon_l()* function is added from The Open Group Technical Standard, 2006, Extended API
9919 Set Part 4.9920 A declaration for the **locale_t** type is added.

9921 **NAME**

9922 mqueue.h — message queues (**REALTIME**)

9923 **SYNOPSIS**

9924 MSG #include <mqueue.h>

9925 **DESCRIPTION**

9926 The <mqueue.h> header shall define the **mqd_t** type, which is used for message queue
9927 descriptors. This is not an array type.

9928 The <mqueue.h> header shall define the **pthread_attr_t**, **size_t**, and **ssize_t** types as described
9929 in <sys/types.h>.

9930 The <mqueue.h> header shall define the **struct timespec** structure as described in <time.h>.

9931 The tag **sigevent** shall be declared as naming an incomplete structure type, the contents of which
9932 are described in the <signal.h> header.

9933 The <mqueue.h> header shall define the **mq_attr** structure, which is used in getting and setting
9934 the attributes of a message queue. Attributes are initially set when the message queue is created.
9935 An **mq_attr** structure shall have at least the following fields:

9936 long mq_flags Message queue flags.
9937 long mq_maxmsg Maximum number of messages.
9938 long mq_msgsize Maximum message size.
9939 long mq_curmsgs Number of messages currently queued.

9940 The following shall be declared as functions and may also be defined as macros. Function
9941 prototypes shall be provided.

9942 int mq_close(mqd_t);
9943 int mq_getattr(mqd_t, struct mq_attr *);
9944 int mq_notify(mqd_t, const struct sigevent *);
9945 mqd_t mq_open(const char *, int, ...);
9946 ssize_t mq_receive(mqd_t, char *, size_t, unsigned *);
9947 int mq_send(mqd_t, const char *, size_t, unsigned);
9948 int mq_setattr(mqd_t, const struct mq_attr *restrict,
9949 struct mq_attr *restrict);
9950 ssize_t mq_timedreceive(mqd_t, char *restrict, size_t,
9951 unsigned *restrict, const struct timespec *restrict);
9952 int mq_timedsend(mqd_t, const char *, size_t, unsigned,
9953 const struct timespec *);
9954 int mq_unlink(const char *);

9955 Inclusion of the <mqueue.h> header may make visible symbols defined in the headers
9956 <fcntl.h>, <signal.h>, and <time.h>.

9957 **APPLICATION USAGE**

9958 None.

9959 **RATIONALE**

9960 None.

9961 **FUTURE DIRECTIONS**

9962 None.

SEE ALSO

[*<fcntl.h>*](#), [*<signal.h>*](#), [*<sys/types.h>*](#), [*<time.h>*](#)

XSH *mq_close()*, *mq_getattr()*, *mq_notify()*, *mq_open()*, *mq_receive()*, *mq_send()*, *mq_setattr()*,
mq_unlink()

CHANGE HISTORY

First released in Issue 5. Included for alignment with the POSIX Realtime Extension.

Issue 6

The **<mqqueue.h>** header is marked as part of the Message Passing option.

The *mq_timedreceive()* and *mq_timedsend()* functions are added for alignment with IEEE Std 1003.1d-1999.

The **restrict** keyword is added to the prototypes for *mq_setattr()* and *mq_timedreceive()*.

Issue 7

Type and structure declarations are added.

9976 **NAME**

9977 ndbm.h — definitions for ndbm database operations

9978 **SYNOPSIS**

9979 XSI `#include <ndbm.h>`

9980 **DESCRIPTION**

9981 The <ndbm.h> header shall define the **datum** type as a structure, which shall include at least the
9982 following members:

9983 `void *dptr` A pointer to the application's data.
9984 `size_t dsize` The size of the object pointed to by *dptr*.

9985 The <ndbm.h> header shall define the **size_t** type as described in <stddef.h>.

9986 The <ndbm.h> header shall define the **DBM** type.

9987 The <ndbm.h> header shall define the following symbolic constants as possible values for the
9988 *store_mode* argument to *dbm_store()*:

9989 **DBM_INSERT** Insertion of new entries only.
9990 **DBM_REPLACE** Allow replacing existing entries.

9991 The following shall be declared as functions and may also be defined as macros. Function
9992 prototypes shall be provided.

9993 `int dbm_clearerr(DBM *);`
9994 `void dbm_close(DBM *);`
9995 `int dbm_delete(DBM *, datum);`
9996 `int dbm_error(DBM *);`
9997 `datum dbm_fetch(DBM *, datum);`
9998 `datum dbm_firstkey(DBM *);`
9999 `datum dbm_nextkey(DBM *);`
10000 `DBM *dbm_open(const char *, int, mode_t);`
10001 `int dbm_store(DBM *, datum, datum, int);`

10002 The <ndbm.h> header shall define the **mode_t** type through **typedef**, as described in
10003 <sys/types.h>.

10004 **APPLICATION USAGE**

10005 None.

10006 **RATIONALE**

10007 None.

10008 **FUTURE DIRECTIONS**

10009 None.

10010 **SEE ALSO**

10011 <stddef.h>, <sys/types.h>

10012 XSH *dbm_clearerr()*

10013 **CHANGE HISTORY**

10014 First released in Issue 4, Version 2.

10015 **Issue 5**

10016 References to the definitions of **size_t** and **mode_t** are added to the DESCRIPTION.

10017 **Issue 7**

10018 This reference page is clarified with respect to macros and symbolic constants.

10019 **NAME**

10020 net/if.h — sockets local interfaces

10021 **SYNOPSIS**

10022 #include <net/if.h>

10023 **DESCRIPTION**

10024 The <net/if.h> header shall define the **if_nameindex** structure, which shall include at least the
10025 following members:

10026 unsigned if_index Numeric index of the interface.
10027 char *if_name Null-terminated name of the interface.

10028 The <net/if.h> header shall define the following symbolic constant for the length of a buffer
10029 containing an interface name (including the terminating NULL character):

10030 IF_NAMESIZE Interface name length.

10031 The following shall be declared as functions and may also be defined as macros. Function
10032 prototypes shall be provided.

10033 void if_freenameindex(struct if_nameindex *);
10034 char *if_indextoname(unsigned, char *);
10035 struct if_nameindex *if_nameindex(void);
10036 unsigned if_nametoindex(const char *);

10037 **APPLICATION USAGE**

10038 None.

10039 **RATIONALE**

10040 None.

10041 **FUTURE DIRECTIONS**

10042 None.

10043 **SEE ALSO**

10044 XSH [if_freenameindex\(\)](#), [if_indextoname\(\)](#), [if_nameindex\(\)](#), [if_nametoindex\(\)](#)

10045 **CHANGE HISTORY**

10046 First released in Issue 6. Derived from the XNS, Issue 5.2 specification.

10047 **Issue 7**

10048 This reference page is clarified with respect to macros and symbolic constants.

10049 **NAME**

10050 netdb.h — definitions for network database operations

10051 **SYNOPSIS**

10052 #include <netdb.h>

10053 **DESCRIPTION**10054 The <netdb.h> header may define the **in_port_t** type and the **in_addr_t** type as described in
10055 <netinet/in.h>.10056 The <netdb.h> header shall define the **hostent** structure, which shall include at least the
10057 following members:

10058	char	*h_name	Official name of the host.
10059	char	**h_aliases	A pointer to an array of pointers to 10060 alternative host names, terminated by a 10061 null pointer.
10062	int	h_addrtype	Address type.
10063	int	h_length	The length, in bytes, of the address.
10064	char	**h_addr_list	A pointer to an array of pointers to network 10065 addresses (in network byte order) for the host, 10066 terminated by a null pointer.

10067 The <netdb.h> header shall define the **netent** structure, which shall include at least the
10068 following members:

10069	char	*n_name	Official, fully-qualified (including the 10070 domain) name of the host.
10071	char	**n_aliases	A pointer to an array of pointers to 10072 alternative network names, terminated by a 10073 null pointer.
10074	int	n_addrtype	The address type of the network.
10075	uint32_t	n_net	The network number, in host byte order.

10076 The <netdb.h> header shall define the **uint32_t** type as described in <inttypes.h>.10077 The <netdb.h> header shall define the **protoent** structure, which shall include at least the
10078 following members:

10079	char	*p_name	Official name of the protocol.
10080	char	**p_aliases	A pointer to an array of pointers to 10081 alternative protocol names, terminated by 10082 a null pointer.
10083	int	p_proto	The protocol number.

10084 The <netdb.h> header shall define the **servent** structure, which shall include at least the
10085 following members:

10086	char	*s_name	Official name of the service.
10087	char	**s_aliases	A pointer to an array of pointers to 10088 alternative service names, terminated by 10089 a null pointer.
10090	int	s_port	A value which, when converted to uint16_t , 10091 yields the port number in network byte order 10092 at which the service resides.
10093	char	*s_proto	The name of the protocol to use when 10094 contacting the service.

10095 The <netdb.h> header shall define the IPPORT_RESERVED symbolic constant with the value of
10096 the highest reserved Internet port number.

10097 Address Information Structure

10098 The <netdb.h> header shall define the **addrinfo** structure, which shall include at least the
10099 following members:

10100	int	ai_flags	Input flags.
10101	int	ai_family	Address family of socket.
10102	int	ai_socktype	Socket type.
10103	int	ai_protocol	Protocol of socket.
10104	socklen_t	ai_addrlen	Length of socket address.
10105	struct sockaddr	*ai_addr	Socket address of socket.
10106	char	*ai_canonname	Canonical name of service location.
10107	struct addrinfo	*ai_next	Pointer to next in list.

10108 The <netdb.h> header shall define the following symbolic constants that evaluate to bitwise-
10109 distinct integer constants for use in the *flags* field of the **addrinfo** structure:

10110	AI_PASSIVE	Socket address is intended for <i>bind()</i> .
10111	AI_CANONNAME	Request for canonical name.
10112	AI_NUMERICHOST	Return numeric host address as name.
10113	AI_NUMERICSERV	Inhibit service name resolution.
10114	AI_V4MAPPED	If no IPv6 addresses are found, query for IPv4 addresses and return them 10115 to the caller as IPv4-mapped IPv6 addresses.
10116	AI_ALL	Query for both IPv4 and IPv6 addresses.
10117	AI_ADDRCONFIG	Query for IPv4 addresses only when an IPv4 address is configured; query 10118 for IPv6 addresses only when an IPv6 address is configured.

10119 The <netdb.h> header shall define the following symbolic constants that evaluate to bitwise-
10120 distinct integer constants for use in the *flags* argument to *getnameinfo()*:

10121	NI_NOFQDN	Only the nodename portion of the FQDN is returned for local hosts.
10122	NI_NUMERICHOST	The numeric form of the node's address is returned instead of its name.
10123	NI_NAMEREQD	Return an error if the node's name cannot be located in the database.
10124	NI_NUMERICSERV	The numeric form of the service address is returned instead of its name.
10125	NI_NUMERICSCOPE	
10126		For IPv6 addresses, the numeric form of the scope identifier is returned 10127 instead of its name.
10128	NI_DGRAM	Indicates that the service is a datagram service (SOCK_DGRAM).

10129 **Address Information Errors**

10130 The <netdb.h> header shall define the following symbolic constants for use as error values for
 10131 *getaddrinfo()* and *getnameinfo()*. The values shall be suitable for use in **#if** preprocessing
 10132 directives.

10133	EAI_AGAIN	The name could not be resolved at this time. Future attempts may
10134		succeed.
10135	EAI_BADFLAGS	The flags had an invalid value.
10136	EAI_FAIL	A non-recoverable error occurred.
10137	EAI_FAMILY	The address family was not recognized or the address length was invalid
10138		for the specified family.
10139	EAI_MEMORY	There was a memory allocation failure.
10140	EAI_NONAME	The name does not resolve for the supplied parameters.
10141		NI_NAMEREQD is set and the host's name cannot be located, or both
10142		<i>nodename</i> and <i>servname</i> were null.
10143	EAI_SERVICE	The service passed was not recognized for the specified socket type.
10144	EAI_SOCKTYPE	The intended socket type was not recognized.
10145	EAI_SYSTEM	A system error occurred. The error code can be found in <i>errno</i> .
10146	EAI_OVERFLOW	An argument buffer overflowed.

10147 The following shall be declared as functions and may also be defined as macros. Function
 10148 prototypes shall be provided.

10149	void	<i>endhostent</i> (void);
10150	void	<i>endnetent</i> (void);
10151	void	<i>endprotoent</i> (void);
10152	void	<i>endservent</i> (void);
10153	void	<i>freeaddrinfo</i> (struct <i>addrinfo</i> *);
10154	const char	* <i>gai_strerror</i> (int);
10155	int	<i>getaddrinfo</i> (const char *restrict, const char *restrict,
10156		const struct <i>addrinfo</i> *restrict,
10157		struct <i>addrinfo</i> **restrict);
10158	struct <i>hostent</i>	* <i>gethostent</i> (void);
10159	int	<i>getnameinfo</i> (const struct <i>sockaddr</i> *restrict, <i>socklen_t</i> ,
10160		char *restrict, <i>socklen_t</i> , char *restrict,
10161		<i>socklen_t</i> , int);
10162	struct <i>netent</i>	* <i>getnetbyaddr</i> (<i>uint32_t</i> , int);
10163	struct <i>netent</i>	* <i>getnetbyname</i> (const char *);
10164	struct <i>netent</i>	* <i>getnetent</i> (void);
10165	struct <i>protoent</i>	* <i>getprotobyname</i> (const char *);
10166	struct <i>protoent</i>	* <i>getprotobynumber</i> (int);
10167	struct <i>protoent</i>	* <i>getprotoent</i> (void);
10168	struct <i>servent</i>	* <i>getservbyname</i> (const char *, const char *);
10169	struct <i>servent</i>	* <i>getservbyport</i> (int, const char *);
10170	struct <i>servent</i>	* <i>getservent</i> (void);
10171	void	<i>sethostent</i> (int);
10172	void	<i>setnetent</i> (int);
10173	void	<i>setprotoent</i> (int);

10174 void setservernt(int);

10175 The <netdb.h> header shall define the **socklen_t** type through **typedef**, as described in

10176 <sys/socket.h>.

10177 Inclusion of the <netdb.h> header may also make visible all symbols from <netinet/in.h>,

10178 <sys/socket.h>, and <inttypes.h>.

10179 **APPLICATION USAGE**

10180 None.

10181 **RATIONALE**

10182 None.

10183 **FUTURE DIRECTIONS**

10184 None.

10185 **SEE ALSO**

10186 <inttypes.h>, <netinet/in.h>, <sys/socket.h>

10187 XSH *bind()*, *endhostent()*, *endnetent()*, *endprotoent()*, *endservent()*, *freeaddrinfo()*, *gai_strerror()*,

10188 *getnameinfo()*

10189 **CHANGE HISTORY**

10190 First released in Issue 6. Derived from the XNS, Issue 5.2 specification.

10191 The Open Group Base Resolution bwg2001-009 is applied, which changes the return type for

10192 *gai_strerror()* from **char *** to **const char ***. This is for coordination with the IPnG Working Group.

10193 IEEE Std 1003.1-2001/Cor 1-2002, item XBD/TC1/D6/11 is applied, adding a description of the

10194 NI_NUMERICSCOPE macro and correcting the *getnameinfo()* function prototype. These changes

10195 are for alignment with IPv6.

10196 **Issue 7**

10197 SD5-XBD-ERN-14 is applied, changing the description of the *s_port* member of the **servent**

10198 structure.

10199 The obsolescent *h_errno* external integer, and the obsolescent *gethostbyaddr()* and *gethostbyname()*

10200 functions are removed, along with the HOST_NOT_FOUND, NO_DATA, NO_RECOVERY, and

10201 TRY_AGAIN macros.

10202 This reference page is clarified with respect to macros and symbolic constants.

10203 **NAME**

10204 netinet/in.h — Internet address family

10205 **SYNOPSIS**

10206 #include <netinet/in.h>

10207 **DESCRIPTION**

10208 The <netinet/in.h> header shall define the following types:

10209 **in_port_t** Equivalent to the type **uint16_t** as described in <inttypes.h>.10210 **in_addr_t** Equivalent to the type **uint32_t** as described in <inttypes.h>.10211 The <netinet_in.h> header shall define the **sa_family_t** type as described in <sys/socket.h>.10212 The <netinet_in.h> header shall define the **uint8_t** and **uint32_t** types as described in
10213 <inttypes.h>. Inclusion of the <netinet/in.h> header may also make visible all symbols from
10214 <inttypes.h> and <sys/socket.h>.10215 The <netinet/in.h> header shall define the **in_addr** structure, which shall include at least the
10216 following member:

10217 in_addr_t s_addr

10218 The <netinet/in.h> header shall define the **sockaddr_in** structure, which shall include at least
10219 the following members:10220 sa_family_t sin_family AF_INET.
10221 in_port_t sin_port Port number.
10222 struct in_addr sin_addr IP address.10223 The *sin_port* and *sin_addr* members shall be in network byte order.10224 The **sockaddr_in** structure is used to store addresses for the Internet address family. Pointers to
10225 this type shall be cast by applications to **struct sockaddr *** for use with socket functions.10226 **IPv6** The <netinet/in.h> header shall define the **in6_addr** structure, which shall include at least the
10227 following member:

10228 uint8_t s6_addr[16]

10229 This array is used to contain a 128-bit IPv6 address, stored in network byte order.

10230 The <netinet/in.h> header shall define the **sockaddr_in6** structure, which shall include at least
10231 the following members:10232 sa_family_t sin6_family AF_INET6.
10233 in_port_t sin6_port Port number.
10234 uint32_t sin6_flowinfo IPv6 traffic class and flow information.
10235 struct in6_addr sin6_addr IPv6 address.
10236 uint32_t sin6_scope_id Set of interfaces for a scope.10237 The *sin6_port* and *sin6_addr* members shall be in network byte order.10238 The **sockaddr_in6** structure shall be set to zero by an application prior to using it, since
10239 implementations are free to have additional, implementation-defined fields in **sockaddr_in6**.10240 The *sin6_scope_id* field is a 32-bit integer that identifies a set of interfaces as appropriate for the
10241 scope of the address carried in the *sin6_addr* field. For a link scope *sin6_addr*, the application
10242 shall ensure that *sin6_scope_id* is a link index. For a site scope *sin6_addr*, the application shall
10243 ensure that *sin6_scope_id* is a site index. The mapping of *sin6_scope_id* to an interface or set of
10244 interfaces is implementation-defined.

10245		The <netinet/in.h> header shall declare the following external variable:
10246		<code>const struct in6_addr in6addr_any</code>
10247		This variable is initialized by the system to contain the wildcard IPv6 address. The
10248		<netinet/in.h> header also defines the IN6ADDR_ANY_INIT macro. This macro must be
10249		constant at compile time and can be used to initialize a variable of type struct in6_addr to the
10250		IPv6 wildcard address.
10251		The <netinet/in.h> header shall declare the following external variable:
10252		<code>const struct in6_addr in6addr_loopback</code>
10253		This variable is initialized by the system to contain the loopback IPv6 address. The
10254		<netinet/in.h> header also defines the IN6ADDR_LOOPBACK_INIT macro. This macro must be
10255		constant at compile time and can be used to initialize a variable of type struct in6_addr to the
10256		IPv6 loopback address.
10257		The <netinet/in.h> header shall define the ipv6_mreq structure, which shall include at least the
10258		following members:
10259		<code>struct in6_addr</code> <code>ipv6mr_multiaddr</code> IPv6 multicast address.
10260		<code>unsigned</code> <code>ipv6mr_interface</code> Interface index.
10261		The <netinet/in.h> header shall define the following symbolic constants for use as values of the
10262		<i>level</i> argument of <i>getsockopt()</i> and <i>setsockopt()</i> :
10263		IPPROTO_IP Internet protocol.
10264	IP6	IPPROTO_IPV6 Internet Protocol Version 6.
10265		IPPROTO_ICMP Control message protocol.
10266	RS	IPPROTO_RAW Raw IP Packets Protocol.
10267		IPPROTO_TCP Transmission control protocol.
10268		IPPROTO_UDP User datagram protocol.
10269		The <netinet/in.h> header shall define the following symbolic constants for use as destination
10270		addresses for <i>connect()</i> , <i>sendmsg()</i> , and <i>sendto()</i> :
10271		INADDR_ANY IPv4 local host address.
10272		INADDR_BROADCAST IPv4 broadcast address.
10273		The <netinet/in.h> header shall define the following symbolic constant, with the value
10274		specified, to help applications declare buffers of the proper size to store IPv4 addresses in string
10275		form:
10276		INET_ADDRSTRLEN 16. Length of the string form for IP.
10277		The <i>htonl()</i> , <i>htons()</i> , <i>ntohl()</i> , and <i>ntohs()</i> functions shall be available as described in
10278		<arpa/inet.h>. Inclusion of the <netinet/in.h> header may also make visible all symbols from
10279		<arpa/inet.h>.
10280	IP6	The <netinet/in.h> header shall define the following symbolic constant, with the value
10281		specified, to help applications declare buffers of the proper size to store IPv6 addresses in string
10282		form:

10283		INET6_ADDRSTRLEN	46. Length of the string form for IPv6.
10284	IP6	The <netinet/in.h> header shall define the following symbolic constants, with distinct integer values, for use in the <i>option_name</i> argument in the <i>getsockopt()</i> or <i>setsockopt()</i> functions at protocol level IPPROTO_IPV6:	
10285			
10286			
10287		IPV6_JOIN_GROUP	Join a multicast group.
10288		IPV6_LEAVE_GROUP	Quit a multicast group.
10289		IPV6_MULTICAST_HOPS	Multicast hop limit.
10290			
10291		IPV6_MULTICAST_IF	Interface to use for outgoing multicast packets.
10292		IPV6_MULTICAST_LOOP	Multicast packets are delivered back to the local application.
10293			
10294		IPV6_UNICAST_HOPS	Unicast hop limit.
10295		IPV6_V6ONLY	Restrict AF_INET6 socket to IPv6 communications only.
10296		The <netinet/in.h> header shall define the following macros that test for special IPv6 addresses. Each macro is of type int and takes a single argument of type const struct in6_addr * :	
10297			
10298		IN6_IS_ADDR_UNSPECIFIED	Unspecified address.
10299			
10300		IN6_IS_ADDR_LOOPBACK	Loopback address.
10301			
10302		IN6_IS_ADDR_MULTICAST	Multicast address.
10303			
10304		IN6_IS_ADDR_LINKLOCAL	Unicast link-local address.
10305			
10306		IN6_IS_ADDR_SITELOCAL	Unicast site-local address.
10307			
10308		IN6_IS_ADDR_V4MAPPED	IPv4 mapped address.
10309			
10310		IN6_IS_ADDR_V4COMPAT	IPv4-compatible address.
10311			
10312		IN6_IS_ADDR_MC_NODELOCAL	Multicast node-local address.
10313			
10314		IN6_IS_ADDR_MC_LINKLOCAL	Multicast link-local address.
10315			
10316		IN6_IS_ADDR_MC_SITELOCAL	Multicast site-local address.
10317			
10318		IN6_IS_ADDR_MC_ORGLOCAL	Multicast organization-local address.
10319			
10320		IN6_IS_ADDR_MC_GLOBAL	Multicast global address.
10321			

10322 **APPLICATION USAGE**

10323 None.

10324 **RATIONALE**

10325 None.

10326 **FUTURE DIRECTIONS**

10327 None.

10328 **SEE ALSO**

10329 Section 4.9 (on page 110), <arpa/inet.h>, <inttypes.h>, <sys/socket.h>

10330 XSH *connect()*, *getsockopt()*, *htonl()*, *sendmsg()*, *sendto()*, *setsockopt()*10331 **CHANGE HISTORY**

10332 First released in Issue 6. Derived from the XNS, Issue 5.2 specification.

10333 The *sin_zero* member was removed from the **sockaddr_in** structure as per The Open Group Base
10334 Resolution bwg2001-004.10335 IEEE Std 1003.1-2001/Cor 1-2002, item XBD/TC1/D6/12 is applied, adding **const** qualifiers to
10336 the *in6addr_any* and *in6addr_loopback* external variables.10337 IEEE Std 1003.1-2001/Cor 2-2004, item XBD/TC2/D6/22 is applied, making it clear which
10338 structure members are in network byte order.10339 **Issue 7**

10340 This reference page is clarified with respect to macros and symbolic constants.

10341 POSIX.1-2008, Technical Corrigendum 1, XBD/TC1-2008/0061 [355] is applied.

+

10342 **NAME**

10343 netinet/tcp.h — definitions for the Internet Transmission Control Protocol (TCP)

10344 **SYNOPSIS**

10345 #include <netinet/tcp.h>

10346 **DESCRIPTION**

10347 The <netinet/tcp.h> header shall define the following symbolic constant for use as a socket
10348 option at the IPPROTO_TCP level:

10349 TCP_NODELAY Avoid coalescing of small segments.

10350 The implementation need not allow the value of the option to be set via *setsockopt()* or retrieved
10351 via *getsockopt()*.

10352 **APPLICATION USAGE**

10353 None.

10354 **RATIONALE**

10355 None.

10356 **FUTURE DIRECTIONS**

10357 None.

10358 **SEE ALSO**

10359 <sys/socket.h>

10360 XSH *getsockopt()*, *setsockopt()*

10361 **CHANGE HISTORY**

10362 First released in Issue 6. Derived from the XNS, Issue 5.2 specification.

10363 **Issue 7**

10364 This reference page is clarified with respect to macros and symbolic constants.

10365 **NAME**

10366 nl_types.h — data types

10367 **SYNOPSIS**

10368 #include <nl_types.h>

10369 **DESCRIPTION**

10370 The <nl_types.h> header shall define at least the following types:

10371 **nl_catd** Used by the message catalog functions *catopen()*, *catgets()*, and *catclose()*
10372 to identify a catalog descriptor.

10373 **nl_item** Used by *nl_langinfo()* to identify items of *langinfo* data. Values of objects
10374 of type **nl_item** are defined in <langinfo.h>.

10375 The <nl_types.h> header shall define at least the following symbolic constants:

10376 **NL_SETD** Used by *gencat* when no *\$set* directive is specified in a message text source
10377 file. This constant can be passed as the value of *set_id* on subsequent calls
10378 to *catgets()* (that is, to retrieve messages from the default message set).
10379 The value of **NL_SETD** is implementation-defined.

10380 **NL_CAT_LOCALE** Value that must be passed as the *oflag* argument to *catopen()* to ensure that
10381 message catalog selection depends on the *LC_MESSAGES* locale category,
10382 rather than directly on the *LANG* environment variable.

10383 The following shall be declared as functions and may also be defined as macros. Function
10384 prototypes shall be provided.

10385 int catclose(nl_catd);
10386 char *catgets(nl_catd, int, int, const char *);
10387 nl_catd catopen(const char *, int);

10388 **APPLICATION USAGE**

10389 None.

10390 **RATIONALE**

10391 None.

10392 **FUTURE DIRECTIONS**

10393 None.

10394 **SEE ALSO**

10395 <langinfo.h>

10396 XSH *catclose()*, *catgets()*, *catopen()*, *nl_langinfo()*

10397 XCU *gencat*

10398 **CHANGE HISTORY**

10399 First released in Issue 2.

10400 **Issue 7**

10401 The <nl_types.h> header is moved from the XSI option to the Base.

10402 This reference page is clarified with respect to macros and symbolic constants.

10403 **NAME**

10404 poll.h — definitions for the poll() function

10405 **SYNOPSIS**

10406 #include <poll.h>

10407 **DESCRIPTION**10408 The <poll.h> header shall define the **pollfd** structure, which shall include at least the following
10409 members:

10410 int fd The following descriptor being polled.

10411 short events The input event flags (see below).

10412 short revents The output event flags (see below).

10413 The <poll.h> header shall define the following type through **typedef**:10414 **nfds_t** An unsigned integer type used for the number of file descriptors.10415 The implementation shall support one or more programming environments in which the width
10416 of **nfds_t** is no greater than the width of type **long**. The names of these programming
10417 environments can be obtained using the *confstr()* function or the *getconf* utility.10418 The <poll.h> header shall define the following symbolic constants, zero or more of which may
10419 be OR'ed together to form the *events* or *revents* members in the **pollfd** structure:

10420 POLLIN Data other than high-priority data may be read without blocking.

10421 POLLRDNORM Normal data may be read without blocking.

10422 POLLRDBAND Priority data may be read without blocking.

10423 POLLPRI High priority data may be read without blocking.

10424 POLLOUT Normal data may be written without blocking.

10425 POLLWRNORM Equivalent to POLLOUT.

10426 POLLWRBAND Priority data may be written.

10427 POLLERR An error has occurred (*revents* only).10428 POLLHUP Device has been disconnected (*revents* only).10429 POLLNVAL Invalid *fd* member (*revents* only).10430 The significance and semantics of normal, priority, and high-priority data are file and device-
10431 specific.10432 The following shall be declared as a function and may also be defined as a macro. A function
10433 prototype shall be provided.

10434 int poll(struct pollfd [], nfds_t, int);

10435 **APPLICATION USAGE**

10436 None.

10437 **RATIONALE**

10438 None.

10439 **FUTURE DIRECTIONS**

10440 None.

10441 **SEE ALSO**

10442 XSH *confstr()*, *poll()*

10443 XCU *getconf*

10444 **CHANGE HISTORY**

10445 First released in Issue 4, Version 2.

10446 **Issue 6**

10447 The description of the symbolic constants is updated to match the *poll()* function.

10448 Text related to STREAMS has been moved to the *poll()* reference page.

10449 A note is added to the DESCRIPTION regarding the significance and semantics of normal,
10450 priority, and high-priority data.

10451 **Issue 7**

10452 The <poll.h> header is moved from the XSI option to the Base.

10453 **NAME**

10454 pthread.h — threads

10455 **SYNOPSIS**

10456 #include <pthread.h>

10457 **DESCRIPTION**

10458 The <pthread.h> header shall define the following symbolic constants:

10459 PTHREAD_BARRIER_SERIAL_THREAD
 10460 PTHREAD_CANCEL_ASYNCHRONOUS
 10461 PTHREAD_CANCEL_ENABLE
 10462 PTHREAD_CANCEL_DEFERRED
 10463 PTHREAD_CANCEL_DISABLE
 10464 PTHREAD_CANCELED
 10465 PTHREAD_CREATE_DETACHED
 10466 PTHREAD_CREATE_JOINABLE
 10467 TPS PTHREAD_EXPLICIT_SCHED
 10468 PTHREAD_INHERIT_SCHED
 10469 PTHREAD_MUTEX_DEFAULT
 10470 PTHREAD_MUTEX_ERRORCHECK
 10471 PTHREAD_MUTEX_NORMAL
 10472 PTHREAD_MUTEX_RECURSIVE
 10473 PTHREAD_MUTEX_ROBUST
 10474 PTHREAD_MUTEX_STALLED
 10475 PTHREAD_ONCE_INIT
 10476 RPI | TPI PTHREAD_PRIO_INHERIT
 10477 MC1 PTHREAD_PRIO_NONE
 10478 RPP | TPP PTHREAD_PRIO_PROTECT
 10479 PTHREAD_PROCESS_SHARED
 10480 PTHREAD_PROCESS_PRIVATE
 10481 TPS PTHREAD_SCOPE_PROCESS
 10482 PTHREAD_SCOPE_SYSTEM

10483 The <pthread.h> header shall define the following compile-time constant expressions valid as
 10484 initializers for the following types:

	Name	Initializer for Type
10485	PTHREAD_COND_INITIALIZER	pthread_cond_t
10486	PTHREAD_MUTEX_INITIALIZER	pthread_mutex_t
10487	PTHREAD_RWLOCK_INITIALIZER	pthread_rwlock_t
10488		

10489 The <pthread.h> header shall define the pthread_attr_t, pthread_barrier_t,
 10490 pthread_barrierattr_t, pthread_cond_t, pthread_condattr_t, pthread_key_t, pthread_mutex_t,
 10491 pthread_mutexattr_t, pthread_once_t, pthread_rwlock_t, pthread_rwlockattr_t,
 10492 pthread_spinlock_t, and pthread_t types as described in <sys/types.h>.

10493 The following shall be declared as functions and may also be defined as macros. Function
 10494 prototypes shall be provided.

```
10495 int pthread_atfork(void (*) (void), void (*) (void),
10496                  void (*) (void));
10497 int pthread_attr_destroy(pthread_attr_t *);
10498 int pthread_attr_getdetachstate(const pthread_attr_t *, int *);
10499 int pthread_attr_getguardsize(const pthread_attr_t *restrict,
```



```

10500         size_t *restrict);
10501 TPS      int  pthread_attr_getinheritsched(const pthread_attr_t *restrict,
10502         int *restrict);
10503         int  pthread_attr_getschedparam(const pthread_attr_t *restrict,
10504         struct sched_param *restrict);
10505 TPS      int  pthread_attr_getschedpolicy(const pthread_attr_t *restrict,
10506         int *restrict);
10507         int  pthread_attr_getscope(const pthread_attr_t *restrict,
10508         int *restrict);
10509 TSA TSS  int  pthread_attr_getstack(const pthread_attr_t *restrict,
10510         void **restrict, size_t *restrict);
10511 TSS      int  pthread_attr_getstacksize(const pthread_attr_t *restrict,
10512         size_t *restrict);
10513         int  pthread_attr_init(pthread_attr_t *);
10514         int  pthread_attr_setdetachstate(pthread_attr_t *, int);
10515         int  pthread_attr_setguardsize(pthread_attr_t *, size_t);
10516 TPS      int  pthread_attr_setinheritsched(pthread_attr_t *, int);
10517         int  pthread_attr_setschedparam(pthread_attr_t *restrict,
10518         const struct sched_param *restrict);
10519 TPS      int  pthread_attr_setschedpolicy(pthread_attr_t *, int);
10520         int  pthread_attr_setscope(pthread_attr_t *, int);
10521 TSA TSS  int  pthread_attr_setstack(pthread_attr_t *, void *, size_t);
10522 TSS      int  pthread_attr_setstacksize(pthread_attr_t *, size_t);
10523         int  pthread_barrier_destroy(pthread_barrier_t *);
10524         int  pthread_barrier_init(pthread_barrier_t *restrict,
10525         const pthread_barrierattr_t *restrict, unsigned);
10526         int  pthread_barrier_wait(pthread_barrier_t *);
10527         int  pthread_barrierattr_destroy(pthread_barrierattr_t *);
10528 TSH      int  pthread_barrierattr_getpshared(
10529         const pthread_barrierattr_t *restrict, int *restrict);
10530         int  pthread_barrierattr_init(pthread_barrierattr_t *);
10531 TSH      int  pthread_barrierattr_setpshared(pthread_barrierattr_t *, int);
10532         int  pthread_cancel(pthread_t);
10533         void  pthread_cleanup_pop(int);
10534         void  pthread_cleanup_push(void (*)(void*), void *);
10535         int  pthread_cond_broadcast(pthread_cond_t *);
10536         int  pthread_cond_destroy(pthread_cond_t *);
10537         int  pthread_cond_init(pthread_cond_t *restrict,
10538         const pthread_condattr_t *restrict);
10539         int  pthread_cond_signal(pthread_cond_t *);
10540         int  pthread_cond_timedwait(pthread_cond_t *restrict,
10541         pthread_mutex_t *restrict, const struct timespec *restrict);
10542         int  pthread_cond_wait(pthread_cond_t *restrict,
10543         pthread_mutex_t *restrict);
10544         int  pthread_condattr_destroy(pthread_condattr_t *);
10545         int  pthread_condattr_getclock(const pthread_condattr_t *restrict,
10546         clockid_t *restrict);
10547 TSH      int  pthread_condattr_getpshared(const pthread_condattr_t *restrict,
10548         int *restrict);
10549         int  pthread_condattr_init(pthread_condattr_t *);
10550         int  pthread_condattr_setclock(pthread_condattr_t *, clockid_t);

```

```

10551 TSH      int  pthread_condattr_setpshared(pthread_condattr_t *, int);
10552          int  pthread_create(pthread_t *restrict, const pthread_attr_t *restrict,
10553                                void *(*)(void*), void *restrict);
10554          int  pthread_detach(pthread_t);
10555          int  pthread_equal(pthread_t, pthread_t);
10556          void  pthread_exit(void *);
10557 OB XSI    int  pthread_getconcurrency(void);
10558 TCT       int  pthread_getcpuclockid(pthread_t, clockid_t *);
10559 TPS       int  pthread_getschedparam(pthread_t, int *restrict,
10560                                         struct sched_param *restrict);
10561          void *pthread_getspecific(pthread_key_t);
10562          int  pthread_join(pthread_t, void **);
10563          int  pthread_key_create(pthread_key_t *, void (*)(void*));
10564          int  pthread_key_delete(pthread_key_t);
10565          int  pthread_mutex_consistent(pthread_mutex_t *);
10566          int  pthread_mutex_destroy(pthread_mutex_t *);
10567 RPP|TPP   int  pthread_mutex_getprioceiling(const pthread_mutex_t *restrict,
10568                                             int *restrict);
10569          int  pthread_mutex_init(pthread_mutex_t *restrict,
10570                                   const pthread_mutexattr_t *restrict);
10571          int  pthread_mutex_lock(pthread_mutex_t *);
10572 RPP|TPP   int  pthread_mutex_setprioceiling(pthread_mutex_t *restrict, int,
10573                                             int *restrict);
10574          int  pthread_mutex_timedlock(pthread_mutex_t *restrict,
10575                                         const struct timespec *restrict);
10576          int  pthread_mutex_trylock(pthread_mutex_t *);
10577          int  pthread_mutex_unlock(pthread_mutex_t *);
10578          int  pthread_mutexattr_destroy(pthread_mutexattr_t *);
10579 RPP|TPP   int  pthread_mutexattr_getprioceiling(
10580                                         const pthread_mutexattr_t *restrict, int *restrict);
10581 MC1       int  pthread_mutexattr_getprotocol(const pthread_mutexattr_t *restrict,
10582                                             int *restrict);
10583 TSH       int  pthread_mutexattr_getpshared(const pthread_mutexattr_t *restrict,
10584                                             int *restrict);
10585          int  pthread_mutexattr_getrobust(const pthread_mutexattr_t *restrict,
10586                                             int *restrict);
10587          int  pthread_mutexattr_gettype(const pthread_mutexattr_t *restrict,
10588                                             int *restrict);
10589          int  pthread_mutexattr_init(pthread_mutexattr_t *);
10590 RPP|TPP   int  pthread_mutexattr_setprioceiling(pthread_mutexattr_t *, int);
10591 MC1       int  pthread_mutexattr_setprotocol(pthread_mutexattr_t *, int);
10592 TSH       int  pthread_mutexattr_setpshared(pthread_mutexattr_t *, int);
10593          int  pthread_mutexattr_setrobust(pthread_mutexattr_t *, int);
10594          int  pthread_mutexattr_settype(pthread_mutexattr_t *, int);
10595          int  pthread_once(pthread_once_t *, void (*)(void));
10596          int  pthread_rwlock_destroy(pthread_rwlock_t *);
10597          int  pthread_rwlock_init(pthread_rwlock_t *restrict,
10598                                   const pthread_rwlockattr_t *restrict);
10599          int  pthread_rwlock_rdlock(pthread_rwlock_t *);
10600          int  pthread_rwlock_timedrdlock(pthread_rwlock_t *restrict,
10601                                           const struct timespec *restrict);
10602          int  pthread_rwlock_timedwrlock(pthread_rwlock_t *restrict,

```

```

10603         const struct timespec *restrict);
10604     int    pthread_rwlock_tryrdlock(pthread_rwlock_t *);
10605     int    pthread_rwlock_trywrlock(pthread_rwlock_t *);
10606     int    pthread_rwlock_unlock(pthread_rwlock_t *);
10607     int    pthread_rwlock_wrlock(pthread_rwlock_t *);
10608     int    pthread_rwlockattr_destroy(pthread_rwlockattr_t *);
10609     TSH    int    pthread_rwlockattr_getpshared(
10610             const pthread_rwlockattr_t *restrict, int *restrict);
10611     int    pthread_rwlockattr_init(pthread_rwlockattr_t *);
10612     TSH    int    pthread_rwlockattr_setpshared(pthread_rwlockattr_t *, int);
10613     pthread_t
10614         pthread_self(void);
10615     int    pthread_setcancelstate(int, int *);
10616     int    pthread_setcanceltype(int, int *);
10617     OB XSI int    pthread_setconcurrency(int);
10618     TPS    int    pthread_setschedparam(pthread_t, int,
10619             const struct sched_param *);
10620     int    pthread_setschedprio(pthread_t, int);
10621     int    pthread_setspecific(pthread_key_t, const void *);
10622     int    pthread_spin_destroy(pthread_spinlock_t *);
10623     int    pthread_spin_init(pthread_spinlock_t *, int);
10624     int    pthread_spin_lock(pthread_spinlock_t *);
10625     int    pthread_spin_trylock(pthread_spinlock_t *);
10626     int    pthread_spin_unlock(pthread_spinlock_t *);
10627     void    pthread_testcancel(void);

10628     Inclusion of the <pthread.h> header shall make symbols defined in the headers <sched.h> and
10629     <time.h> visible.

```

APPLICATION USAGE

None.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

<sched.h>, <sys/types.h>, <time.h>

XSH *pthread_atfork()*, *pthread_attr_destroy()*, *pthread_attr_getdetachstate()*,
pthread_attr_getguardsize(), *pthread_attr_getinheritsched()*, *pthread_attr_getschedparam()*,
pthread_attr_getschedpolicy(), *pthread_attr_getscope()*, *pthread_attr_getstack()*,
pthread_attr_getstacksize(), *pthread_barrier_destroy()*, *pthread_barrier_wait()*,
pthread_barrierattr_destroy(), *pthread_barrierattr_getpshared()*, *pthread_cancel()*,
pthread_cleanup_pop(), *pthread_cond_broadcast()*, *pthread_cond_destroy()*, *pthread_cond_timedwait()*,
pthread_condattr_destroy(), *pthread_condattr_getclock()*, *pthread_condattr_getpshared()*,
pthread_create(), *pthread_detach()*, *pthread_equal()*, *pthread_exit()*, *pthread_getconcurrency()*,
pthread_getcpuclockid(), *pthread_getschedparam()*, *pthread_getspecific()*, *pthread_join()*,
pthread_key_create(), *pthread_key_delete()*, *pthread_mutex_consistent()*, *pthread_mutex_destroy()*,
pthread_mutex_getprioceiling(), *pthread_mutex_lock()*, *pthread_mutex_timedlock()*,
pthread_mutexattr_destroy(), *pthread_mutexattr_getprioceiling()*, *pthread_mutexattr_getprotocol()*,
pthread_mutexattr_getpshared(), *pthread_mutexattr_getrobust()*, *pthread_mutexattr_gettype()*,
pthread_once(), *pthread_rwlock_destroy()*, *pthread_rwlock_rdlock()*, *pthread_rwlock_timedrdlock()*,

*pthread_rwlock_timedwrlock(), pthread_rwlock_trywrlock(), pthread_rwlock_unlock(),
pthread_rwlockattr_destroy(), pthread_rwlockattr_getpshared(), pthread_self(),
pthread_setcancelstate(), pthread_setschedprio(), pthread_spin_destroy(), pthread_spin_lock(),
pthread_spin_unlock()*

CHANGE HISTORY

First released in Issue 5. Included for alignment with the POSIX Threads Extension.

Issue 6

The RTT margin markers are broken out into their POSIX options.

The Open Group Corrigendum U021/9 is applied, correcting the prototype for the *pthread_cond_wait()* function.

The Open Group Corrigendum U026/2 is applied, correcting the prototype for the *pthread_setschedparam()* function so that its second argument is of type **int**.

The *pthread_getcpuclockid()* and *pthread_mutex_timedlock()* functions are added for alignment with IEEE Std 1003.1d-1999.

The following functions are added for alignment with IEEE Std 1003.1j-2000:

*pthread_barrier_destroy(), pthread_barrier_init(), pthread_barrier_wait(),
pthread_barrierattr_destroy(), pthread_barrierattr_getpshared(), pthread_barrierattr_init(),
pthread_barrierattr_setpshared(), pthread_condattr_getclock(), pthread_condattr_setclock(),
pthread_rwlock_timedrdlock(), pthread_rwlock_timedwrlock(), pthread_spin_destroy(),
pthread_spin_init(), pthread_spin_lock(), pthread_spin_trylock(), and pthread_spin_unlock().*

PTHREAD_RWLOCK_INITIALIZER is removed for alignment with IEEE Std 1003.1j-2000.

Functions previously marked as part of the Read-Write Locks option are now moved to the Threads option.

The **restrict** keyword is added to the prototypes for *pthread_attr_getguardsize(),
pthread_attr_getinheritsched(), pthread_attr_getschedparam(), pthread_attr_getschedpolicy(),
pthread_attr_getscope(), pthread_attr_getstackaddr(), pthread_attr_getstacksize(),
pthread_attr_setschedparam(), pthread_barrier_init(), pthread_barrierattr_getpshared(),
pthread_cond_init(), pthread_cond_signal(), pthread_cond_timedwait(), pthread_cond_wait(),
pthread_condattr_getclock(), pthread_condattr_getpshared(), pthread_create(),
pthread_getschedparam(), pthread_mutex_getprioceiling(), pthread_mutex_init(),
pthread_mutex_setprioceiling(), pthread_mutexattr_getprioceiling(), pthread_mutexattr_getprotocol(),
pthread_mutexattr_getpshared(), pthread_mutexattr_gettype(), pthread_rwlock_init(),
pthread_rwlock_timedrdlock(), pthread_rwlock_timedwrlock(), pthread_rwlockattr_getpshared(), and
pthread_sigmask().*

IEEE PASC Interpretation 1003.1 #86 is applied, allowing the symbols from <sched.h> and <time.h> to be made visible when <pthread.h> is included. Previously this was an XSI option.

IEEE PASC Interpretation 1003.1c #42 is applied, removing the requirement for prototypes for the *pthread_kill()* and *pthread_sigmask()* functions. These are required to be in the <signal.h> header. They are allowed here through the name space rules.

IEEE PASC Interpretation 1003.1 #96 is applied, adding the *pthread_setschedprio()* function.

IEEE Std 1003.1-2001/Cor 1-2002, item XBD/TC1/D6/13 is applied, correcting shading errors that were in contradiction with the System Interfaces volume of POSIX.1-2008.

Issue 7

SD5-XBD-ERN-55 is applied, adding the **restrict** keyword to the *pthread_mutex_timedlock()* function prototype.

SD5-XBD-ERN-62 is applied.

Austin Group Interpretation 1003.1-2001 #048 is applied, reinstating the PTHREAD_RWLOCK_INITIALIZER symbol.

The <pthread.h> header is moved from the Threads option to the Base.

The following extended mutex types are moved from the XSI option to the Base:

```
PTHREAD_MUTEX_NORMAL
PTHREAD_MUTEX_ERRORCHECK
PTHREAD_MUTEX_RECURSIVE
PTHREAD_MUTEX_DEFAULT
```

The PTHREAD_MUTEX_ROBUST and PTHREAD_MUTEX_STALLED symbols and the *pthread_mutex_consistent()*, *pthread_mutexattr_getrobust()*, and *pthread_mutexattr_setrobust()* functions are added from The Open Group Technical Standard, 2006, Extended API Set Part 2.

Functionality relating to the Thread Priority Protection and Thread Priority Inheritance options is changed to be Non-Robust Mutex or Robust Mutex Priority Protection and Non-Robust Mutex or Robust Mutex Priority Inheritance, respectively.

This reference page is clarified with respect to macros and symbolic constants.

10713 **NAME**

10714 pwd.h — password structure

10715 **SYNOPSIS**

10716 #include <pwd.h>

10717 **DESCRIPTION**10718 The <pwd.h> header shall define the **struct passwd**, structure, which shall include at least the
10719 following members:

10720	char	*pw_name	User's login name.
10721	uid_t	pw_uid	Numerical user ID.
10722	gid_t	pw_gid	Numerical group ID.
10723	char	*pw_dir	Initial working directory.
10724	char	*pw_shell	Program to use as shell.

10725 The <pwd.h> header shall define the **gid_t**, **uid_t**, and **size_t** types as described in
10726 <sys/types.h>.10727 The following shall be declared as functions and may also be defined as macros. Function
10728 prototypes shall be provided.

10729	XSI	void	endpwent(void);
10730		struct passwd *	getpwent(void);
10731		struct passwd *	getpwnam(const char *);
10732		int	getpwnam_r(const char *, struct passwd *, char *,
10733			size_t, struct passwd **);
10734		struct passwd *	getpwuid(uid_t);
10735		int	getpwuid_r(uid_t, struct passwd *, char *,
10736			size_t, struct passwd **);
10737	XSI	void	setpwent(void);

10738 **APPLICATION USAGE**

10739 None.

10740 **RATIONALE**

10741 None.

10742 **FUTURE DIRECTIONS**

10743 None.

10744 **SEE ALSO**

10745 <sys/types.h>

10746 XSH *endpwent()*, *getpwnam()*, *getpwuid()*10747 **CHANGE HISTORY**

10748 First released in Issue 1.

10749 **Issue 5**

10750 The DESCRIPTION is updated for alignment with the POSIX Threads Extension.

10751 **Issue 6**10752 The following new requirements on POSIX implementations derive from alignment with the
10753 Single UNIX Specification:

- 10754
- The **gid_t** and **uid_t** types are mandated.

- 10755 • The `getpwnam_r()` and `getpwuid_r()` functions are marked as part of the Thread-Safe
- 10756 Functions option.
- 10757 **Issue 7**
- 10758 SD5-XBD-ERN-56 is applied, adding a reference to <sys/types.h> for the `size_t` type.

10759 **NAME**

10760 regex.h — regular expression matching types

10761 **SYNOPSIS**

10762 #include <regex.h>

10763 **DESCRIPTION**10764 The <regex.h> header shall define the structures and symbolic constants used by the *regcomp()*,
10765 *regexexec()*, *regerror()*, and *regfree()* functions.10766 The <regex.h> header shall define the **regex_t** structure type, which shall include at least the
10767 following member:

10768 size_t re_nsub Number of parenthesized subexpressions.

10769 The <regex.h> header shall define the **size_t** type as described in <sys/types.h>.10770 The <regex.h> header shall define the **regoff_t** type as a signed integer type that can hold the
10771 largest value that can be stored in either a **ptrdiff_t** type or a **ssize_t** type.10772 The <regex.h> header shall define the **regmatch_t** structure type, which shall include at least the
10773 following members:10774 regoff_t rm_so Byte offset from start of string
10775 to start of substring.10776 regoff_t rm_eo Byte offset from start of string of the
10777 first character after the end of substring.10778 The <regex.h> header shall define the following symbolic constants for the *cflags* parameter to
10779 the *regcomp()* function:

10780 REG_EXTENDED Use Extended Regular Expressions.

10781 REG_ICASE Ignore case in match.

10782 REG_NOSUB Report only success or fail in *regexexec()*.

10783 REG_NEWLINE Change the handling of <newline>.

10784 The <regex.h> header shall define the following symbolic constants for the *eflags* parameter to
10785 the *regexexec()* function:10786 REG_NOTBOL The <circumflex> character ('^'), when taken as a special character, does
10787 not match the beginning of *string*.10788 REG_NOTEOL The <dollar-sign> ('\$'), when taken as a special character, does not
10789 match the end of *string*.

10790 The <regex.h> header shall define the following symbolic constants as error return values:

10791 REG_NOMATCH *regexexec()* failed to match.

10792 REG_BADPAT Invalid regular expression.

10793 REG_ECOLLATE Invalid collating element referenced.

10794 REG_ETYPE Invalid character class type referenced.

10795 REG_EESCAPE Trailing <backslash> character in pattern.

10796 REG_ESUBREG Number in *\digit* invalid or in error.

10797	REG_EBRACK	"[]" imbalance.
10798	REG_EPAREN	"\(\)" or "()" imbalance.
10799	REG_EBRACE	"\{\}" imbalance.
10800	REG_BADBR	Content of "\{\}" invalid: not a number, number too large, more than
10801		two numbers, first larger than second.
10802	REG_ERANGE	Invalid endpoint in range expression.
10803	REG_ESPACE	Out of memory.
10804	REG_BADRPT	'?', '*', or '+' not preceded by valid regular expression.
10805	The following shall be declared as functions and may also be defined as macros. Function	
10806	prototypes shall be provided.	
10807	<code>int regcomp(regex_t *restrict, const char *restrict, int);</code>	
10808	<code>size_t regerror(int, const regex_t *restrict, char *restrict, size_t);</code>	
10809	<code>int regexec(const regex_t *restrict, const char *restrict, size_t,</code>	
10810	<code>regmatch_t [restrict], int);</code>	
10811	<code>void regfree(regex_t *);</code>	
10812	The implementation may define additional macros or constants using names beginning with	
10813	REG_.	
10814	APPLICATION USAGE	
10815	None.	
10816	RATIONALE	
10817	None.	
10818	FUTURE DIRECTIONS	
10819	None.	
10820	SEE ALSO	
10821	<sys/types.h>	
10822	XSH regcomp()	
10823	CHANGE HISTORY	
10824	First released in Issue 4.	
10825	Originally derived from the ISO POSIX-2 standard.	
10826	Issue 6	
10827	The REG_ENOSYS constant is marked obsolescent.	
10828	The restrict keyword is added to the prototypes for <i>regcomp()</i> , <i>regerror()</i> , and <i>regexec()</i> .	
10829	A statement is added that the size_t type is defined as described in <sys/types.h> .	
10830	Issue 7	
10831	SD5-XBD-ERN-60 is applied.	
10832	The obsolescent REG_ENOSYS constant is removed.	
10833	This reference page is clarified with respect to macros and symbolic constants.	

10834 **NAME**

10835 sched.h — execution scheduling

10836 **SYNOPSIS**

10837 #include <sched.h>

10838 **DESCRIPTION**10839 PS The <sched.h> header shall define the **pid_t** type as described in <sys/types.h>.10840 SS|TSP The <sched.h> header shall define the **time_t** type as described in <sys/types.h>.10841 The <sched.h> header shall define the **timespec** structure as described in <time.h>.

10842 The <sched.h> header shall define the **sched_param** structure, which shall include the
 10843 scheduling parameters required for implementation of each supported scheduling policy. This
 10844 structure shall include at least the following member:

10845 int sched_priority Process or thread execution scheduling priority.

10846 SS|TSP The **sched_param** structure defined in <sched.h> shall include the following members in
 10847 addition to those specified above:

10848 int sched_ss_low_priority Low scheduling priority for
 10849 sporadic server.

10850 struct timespec sched_ss_repl_period Replenishment period for
 10851 sporadic server.

10852 struct timespec sched_ss_init_budget Initial budget for sporadic server.

10853 int sched_ss_max_repl Maximum pending replenishments for
 10854 sporadic server.

10855 Each process or thread is controlled by an associated scheduling policy and priority. Associated
 10856 with each policy is a priority range. Each policy definition specifies the minimum priority range
 10857 for that policy. The priority ranges for each policy may overlap the priority ranges of other
 10858 policies.

10859 Four scheduling policies are defined; others may be defined by the implementation. The four
 10860 standard policies are indicated by the values of the following symbolic constants:

10861 PS|TPS **SCHED_FIFO** First in-first out (FIFO) scheduling policy.

10862 PS|TPS **SCHED_RR** Round robin scheduling policy.

10863 SS|TSP **SCHED_SPORADIC** Sporadic server scheduling policy.

10864 PS|TPS **SCHED_OTHER** Another scheduling policy.

10865 The values of these constants are distinct.

10866 The following shall be declared as functions and may also be defined as macros. Function
 10867 prototypes shall be provided.

```
10868 PS|TPS int sched_get_priority_max(int);
10869 int sched_get_priority_min(int);
10870 PS int sched_getparam(pid_t, struct sched_param *);
10871 int sched_getscheduler(pid_t);
10872 PS|TPS int sched_rr_get_interval(pid_t, struct timespec *);
10873 PS int sched_setparam(pid_t, const struct sched_param *);
10874 int sched_setscheduler(pid_t, int, const struct sched_param *);
10875 int sched_yield(void);
```

10876 Inclusion of the <sched.h> header may make visible all symbols from the <time.h> header.

10877 **APPLICATION USAGE**

10878 None.

10879 **RATIONALE**

10880 None.

10881 **FUTURE DIRECTIONS**

10882 None.

10883 **SEE ALSO**

10884 <sys/types.h>, <time.h>

10885 XSH *sched_get_priority_max()*, *sched_getparam()*, *sched_getscheduler()*, *sched_rr_get_interval()*,
 10886 *sched_setparam()*, *sched_setscheduler()*, *sched_yield()*

10887 **CHANGE HISTORY**

10888 First released in Issue 5. Included for alignment with the POSIX Realtime Extension.

10889 **Issue 6**

10890 The <sched.h> header is marked as part of the Process Scheduling option.

10891 Sporadic server members are added to the **sched_param** structure, and the SCHED_SPORADIC
 10892 scheduling policy is added for alignment with IEEE Std 1003.1d-1999.

10893 IEEE PASC Interpretation 1003.1 #108 is applied, correcting the **sched_param** structure whose
 10894 members *sched_ss_repl_period* and *sched_ss_init_budget* should be type **struct timespec** and not
 10895 **timespec**.

10896 Symbols from <time.h> may be made visible when <sched.h> is included.

10897 IEEE Std 1003.1-2001/Cor 1-2002, items XSH/TC1/D6/52 and XSH/TC1/D6/53 are applied,
 10898 aligning the function prototype shading and margin codes with the System Interfaces volume of
 10899 IEEE Std 1003.1-2001.

10900 IEEE Std 1003.1-2001/Cor 2-2004, item XBD/TC2/D6/23 is applied, updating the
 10901 DESCRIPTION to differentiate between thread and process execution.

10902 **Issue 7**

10903 SD5-XBD-ERN-13 is applied.

10904 Austin Group Interpretation 1003.1-2001 #064 is applied, correcting the options markings.

10905 The <sched.h> headers is moved from the Threads option to the Base.

10906 Declarations for the **pid_t** and **time_t** types and the **timespec** structure are added.

10907 **NAME**

10908 search.h — search tables

10909 **SYNOPSIS**10910 XSI `#include <search.h>`10911 **DESCRIPTION**

10912 The <search.h> header shall define the **ENTRY** type for structure **entry** which shall include the
 10913 following members:

```
10914 char    *key
10915 void     *data
```

10916 and shall define **ACTION** and **VISIT** as enumeration data types through type definitions as
 10917 follows:

```
10918 enum { FIND, ENTER } ACTION;
10919 enum { preorder, postorder, endorder, leaf } VISIT;
```

10920 The <search.h> header shall define the **size_t** type as described in <sys/types.h>.

10921 The following shall be declared as functions and may also be defined as macros. Function
 10922 prototypes shall be provided.

```
10923 int      hcreate(size_t);
10924 void     hdestroy(void);
10925 ENTRY *hsearch(ENTRY, ACTION);
10926 void     insque(void *, void *);
10927 void     *lfind(const void *, const void *, size_t *,
10928               size_t, int (*)(const void *, const void *));
10929 void     *lsearch(const void *, void *, size_t *,
10930               size_t, int (*)(const void *, const void *));
10931 void     remque(void *);
10932 void     *tdelete(const void *restrict, void **restrict,
10933               int (*)(const void *, const void *));
10934 void     *tfind(const void *, void *const *,
10935               int (*)(const void *, const void *));
10936 void     *tsearch(const void *, void **,
10937               int (*)(const void *, const void *));
10938 void     twalk(const void *,
10939               void (*)(const void *, VISIT, int ));
```

10940 **APPLICATION USAGE**

10941 None.

10942 **RATIONALE**

10943 None.

10944 **FUTURE DIRECTIONS**

10945 None.

10946 **SEE ALSO**10947 [<sys/types.h>](#)10948 XSH [hcreate\(\)](#), [insque\(\)](#), [lsearch\(\)](#), [tdelete\(\)](#)

10949 **CHANGE HISTORY**

10950 First released in Issue 1. Derived from Issue 1 of the SVID.

10951 **Issue 6**

10952 The Open Group Corrigendum U021/6 is applied, updating the prototypes for *tdelete()* and
10953 *tsearch()*.

10954 The **restrict** keyword is added to the prototype for *tdelete()*.

10955 **NAME**

10956 semaphore.h — semaphores

10957 **SYNOPSIS**

10958 #include <semaphore.h>

10959 **DESCRIPTION**

10960 The <semaphore.h> header shall define the **sem_t** type, used in performing semaphore
 10961 operations. The semaphore may be implemented using a file descriptor, in which case
 10962 applications are able to open up at least a total of {OPEN_MAX} files and semaphores.

10963 The <semaphore.h> header shall define the symbolic constant SEM_FAILED which shall have
 10964 type **sem_t** *.

10965 The following shall be declared as functions and may also be defined as macros. Function
 10966 prototypes shall be provided.

```
10967 int      sem_close(sem_t *);
10968 int      sem_destroy(sem_t *);
10969 int      sem_getvalue(sem_t *restrict, int *restrict);
10970 int      sem_init(sem_t *, int, unsigned);
10971 sem_t *sem_open(const char *, int, ...);
10972 int      sem_post(sem_t *);
10973 int      sem_timedwait(sem_t *restrict, const struct timespec *restrict);
10974 int      sem_trywait(sem_t *);
10975 int      sem_unlink(const char *);
10976 int      sem_wait(sem_t *);
```

10977 Inclusion of the <semaphore.h> header may make visible symbols defined in the <fcntl.h> and
 10978 <time.h> headers.

10979 **APPLICATION USAGE**

10980 None.

10981 **RATIONALE**

10982 None.

10983 **FUTURE DIRECTIONS**

10984 None.

10985 **SEE ALSO**

10986 <fcntl.h>, <sys/types.h>, <time.h>

10987 XSH *sem_close()*, *sem_destroy()*, *sem_getvalue()*, *sem_init()*, *sem_open()*, *sem_post()*,
 10988 *sem_timedwait()*, *sem_trywait()*, *sem_unlink()*

10989 **CHANGE HISTORY**

10990 First released in Issue 5. Included for alignment with the POSIX Realtime Extension.

10991 **Issue 6**

10992 The <semaphore.h> header is marked as part of the Semaphores option.

10993 The Open Group Corrigendum U021/3 is applied, adding a description of SEM_FAILED.

10994 The *sem_timedwait()* function is added for alignment with IEEE Std 1003.1d-1999.10995 The **restrict** keyword is added to the prototypes for *sem_getvalue()* and *sem_timedwait()*.

10996 **Issue 7**

10997 SD5-XBD-ERN-57 is applied, allowing the header to make visible symbols from the <time.h>
10998 header.

10999 The <semaphore.h> header is moved from the Semaphores option to the Base.

11000 This reference page is clarified with respect to macros and symbolic constants.

11001 **NAME**

11002 setjmp.h — stack environment declarations

11003 **SYNOPSIS**

11004 #include <setjmp.h>

11005 **DESCRIPTION**

11006 CX Some of the functionality described on this reference page extends the ISO C standard.
 11007 Applications shall define the appropriate feature test macro (see XSH [Section 2.2](#), on page 472) to
 11008 enable the visibility of these symbols in this header.

11009 CX The <setjmp.h> header shall define the array types **jmp_buf** and **sigjmp_buf**.

11010 The following shall be declared as functions and may also be defined as macros. Function
 11011 prototypes shall be provided.

11012 OB XSI void _longjmp(jmp_buf, int);

11013 void longjmp(jmp_buf, int);

11014 CX void siglongjmp(sigjmp_buf, int);

11015 The following may be declared as functions, or defined as macros, or both. If functions are
 11016 declared, function prototypes shall be provided.

11017 OB XSI int _setjmp(jmp_buf);

11018 int setjmp(jmp_buf);

11019 CX int sigsetjmp(sigjmp_buf, int);

11020 **APPLICATION USAGE**

11021 None.

11022 **RATIONALE**

11023 None.

11024 **FUTURE DIRECTIONS**

11025 None.

11026 **SEE ALSO**11027 XSH [Section 2.2](#) (on page 472), [_longjmp\(\)](#), [longjmp\(\)](#), [setjmp\(\)](#), [siglongjmp\(\)](#), [sigsetjmp\(\)](#)11028 **CHANGE HISTORY**

11029 First released in Issue 1.

11030 **Issue 6**

11031 Extensions beyond the ISO C standard are marked.

11032 **Issue 7**

11033 SD5-XBD-ERN-6 is applied.

11034 **NAME**

11035 signal.h — signals

11036 **SYNOPSIS**

11037 #include <signal.h>

11038 **DESCRIPTION**

11039 CX Some of the functionality described on this reference page extends the ISO C standard.
11040 Applications shall define the appropriate feature test macro (see XSH [Section 2.2](#), on page 472) to
11041 enable the visibility of these symbols in this header.

11042 The <signal.h> header shall define the following macros, which shall expand to constant
11043 expressions with distinct values that have a type compatible with the second argument to, and
11044 the return value of, the *signal()* function, and whose values shall compare unequal to the
11045 address of any declarable function.

11046 SIG_DFL Request for default signal handling.

11047 SIG_ERR Return value from *signal()* in case of error.

11048 OB XSI SIG_HOLD Request that signal be held.

11049 SIG_IGN Request that signal be ignored.

11050 CX The <signal.h> header shall define the **pthread_t**, **size_t**, and **uid_t** types as described in
11051 <sys/types.h>.

11052 The <signal.h> header shall define the **timespec** structure as described in <time.h>.

11053 The <signal.h> header shall define the following data types:

11054 **sig_atomic_t** Possibly volatile-qualified integer type of an object that can be accessed as
11055 an atomic entity, even in the presence of asynchronous interrupts.

11056 CX **sigset_t** Integer or structure type of an object used to represent sets of signals.

11057 CX **pid_t** As described in <sys/types.h>.

11058 CX The <signal.h> header shall define the **pthread_attr_t** type as described in <sys/types.h>.

11059 The <signal.h> header shall define the **sigevent** structure, which shall include at least the
11060 following members:

11061	int	sigev_notify	Notification type.
11062	int	sigev_signo	Signal number.
11063	union sigval	sigev_value	Signal value.
11064	void	(*sigev_notify_function) (union sigval)	Notification function.
11065			Notification attributes.
11066	pthread_attr_t *	sigev_notify_attributes	

11067 The <signal.h> header shall define the following symbolic constants for the values of
11068 *sigev_notify*:

11069 SIGEV_NONE No asynchronous notification is delivered when the event of interest
11070 occurs.

11071 SIGEV_SIGNAL A queued signal, with an application-defined value, is generated when
11072 the event of interest occurs.

11073 SIGEV_THREAD A notification function is called to perform notification.

11074 The **signal** union shall be defined as:

11075 `int sival_int` Integer signal value.

11076 `void *sival_ptr` Pointer signal value.

11077 The <**signal.h**> header shall declare the SIGRTMIN and SIGRTMAX macros, which shall expand

11078 to positive integer expressions with type **int**, but which need not be constant expressions. These

11079 macros specify a range of signal numbers that are reserved for application use and for which the

11080 realtime signal behavior specified in this volume of POSIX.1-2008 is supported. The signal

11081 numbers in this range do not overlap any of the signals specified in the following table.

11082 The range SIGRTMIN through SIGRTMAX inclusive shall include at least {RTSIG_MAX} signal

11083 numbers.

11084 It is implementation-defined whether realtime signal behavior is supported for other signals.

11085 The <**signal.h**> header shall define the following macros that are used to refer to the signals that

11086 occur in the system. Signals defined here begin with the letters SIG followed by an uppercase

11087 letter. The macros shall expand to positive integer constant expressions with type **int** and

11088 distinct values. The value 0 is reserved for use as the null signal (see *kill()*). Additional

11089 implementation-defined signals may occur in the system.

11090 CX The ISO C standard only requires the signal names SIGABRT, SIGFPE, SIGILL, SIGINT,

11091 SIGSEGV, and SIGTERM to be defined.

11092 The following signals shall be supported on all implementations (default actions are explained

11093 below the table):

	Signal	Default Action	Description
11094			
11095	SIGABRT	A	Process abort signal.
11096	SIGALRM	T	Alarm clock.
11097	SIGBUS	A	Access to an undefined portion of a memory object.
11098	SIGCHLD	I	Child process terminated, stopped,
11099	XSI		or continued.
11100	SIGCONT	C	Continue executing, if stopped.
11101	SIGFPE	A	Erroneous arithmetic operation.
11102	SIGHUP	T	Hangup.
11103	SIGILL	A	Illegal instruction.
11104	SIGINT	T	Terminal interrupt signal.
11105	SIGKILL	T	Kill (cannot be caught or ignored).
11106	SIGPIPE	T	Write on a pipe with no one to read it.
11107	SIGQUIT	A	Terminal quit signal.
11108	SIGSEGV	A	Invalid memory reference.
11109	SIGSTOP	S	Stop executing (cannot be caught or ignored).
11110	SIGTERM	T	Termination signal.
11111	SIGTSTP	S	Terminal stop signal.
11112	SIGTTIN	S	Background process attempting read.
11113	SIGTTOU	S	Background process attempting write.
11114	SIGUSR1	T	User-defined signal 1.
11115	SIGUSR2	T	User-defined signal 2.
11116	OB XSR	T	Pollable event.
11117	OB XSI	T	Profiling timer expired.
11118	XSI	A	Bad system call.
11119		A	Trace/breakpoint trap.
11120	SIGURG	I	High bandwidth data is available at a socket.
11121	XSI	T	Virtual timer expired.
11122		A	CPU time limit exceeded.
11123		A	File size limit exceeded.

11124 The default actions are as follows:

- 11125 T Abnormal termination of the process.
- 11126 XSI A Abnormal termination of the process with additional actions.
- 11127 I Ignore the signal.
- 11128 S Stop the process.
- 11129 C Continue the process, if it is stopped; otherwise, ignore the signal.

11130 The effects on the process in each case are described in XSH [Section 2.4.3](#) (on page 490).

11131 CX The <signal.h> header shall declare the **sigaction** structure, which shall include at least the

11132 following members:

```

11133 void    (*sa_handler)(int)    Pointer to a signal-catching function
11134                                     or one of the SIG_IGN or SIG_DFL.
11135 sigset_t sa_mask              Set of signals to be blocked during execution
11136                                     of the signal handling function.
11137 int      sa_flags              Special flags.
11138 void    (*sa_sigaction)(int, siginfo_t *, void *)
11139                                     Pointer to a signal-catching function.
```

11140	CX	The storage occupied by <i>sa_handler</i> and <i>sa_sigaction</i> may overlap, and a conforming application shall not use both simultaneously.	
11141			
11142		The <signal.h> header shall define the following macros which shall expand to integer constant expressions that need not be usable in #if preprocessing directives:	
11143			
11144	CX	SIG_BLOCK	The resulting set is the union of the current set and the signal set pointed to by the argument <i>set</i> .
11145			
11146	CX	SIG_UNBLOCK	The resulting set is the intersection of the current set and the complement of the signal set pointed to by the argument <i>set</i> .
11147			
11148	CX	SIG_SETMASK	The resulting set is the signal set pointed to by the argument <i>set</i> .
11149		The <signal.h> header shall also define the following symbolic constants:	
11150	CX	SA_NOCLDSTOP	Do not generate SIGCHLD when children stop
11151	XSI		or stopped children continue.
11152	XSI	SA_ONSTACK	Causes signal delivery to occur on an alternate stack.
11153	CX	SA_RESETHAND	Causes signal dispositions to be set to SIG_DFL on entry to signal handlers.
11154			
11155	CX	SA_RESTART	Causes certain functions to become restartable.
11156	CX	SA_SIGINFO	Causes extra information to be passed to signal handlers at the time of receipt of a signal.
11157			
11158	CX	SA_NOCLDWAIT	Causes implementations not to create zombie processes on child death.
11159	CX	SA_NODEFER	Causes signal not to be automatically blocked on entry to signal handler.
11160	XSI	SS_ONSTACK	Process is executing on an alternate signal stack.
11161	XSI	SS_DISABLE	Alternate signal stack is disabled.
11162	XSI	MINSIGSTKSZ	Minimum stack size for a signal handler.
11163	XSI	SIGSTKSZ	Default size in bytes for the alternate signal stack.
11164	CX	The <signal.h> header shall define the mcontext_t type through typedef .	
11165	CX	The <signal.h> header shall define the ucontext_t type as a structure that shall include at least the following members:	
11166			
11167		ucontext_t *uc_link	Pointer to the context that is resumed when this context returns.
11168			
11169		sigset_t uc_sigmask	The set of signals that are blocked when this context is active.
11170			
11171		stack_t uc_stack	The stack used by this context.
11172		mcontext_t uc_mcontext	A machine-specific representation of the saved context.
11173			
11174		The <signal.h> header shall define the stack_t type as a structure, which shall include at least the following members:	
11175			
11176		void *ss_sp	Stack base or pointer.
11177		size_t ss_size	Stack size.
11178		int ss_flags	Flags.

11179 CX The <signal.h> header shall define the **siginfo_t** type as a structure, which shall include at least
11180 the following members:

11181	CX	int	si_signo	Signal number.
11182		int	si_code	Signal code.
11183	XSI	int	si_errno	If non-zero, an <i>errno</i> value associated with
11184				this signal, as described in <errno.h>.
11185	CX	pid_t	si_pid	Sending process ID.
11186		uid_t	si_uid	Real user ID of sending process.
11187		void	*si_addr	Address of faulting instruction.
11188		int	si_status	Exit value or signal.
11189	OB XSR	long	si_band	Band event for SIGPOLL.
11190	CX	union sigval	si_value	Signal value.

11191 CX The <signal.h> header shall define the symbolic constants in the **Code** column of the following
11192 table for use as values of *si_code* that are signal-specific or non-signal-specific reasons why the
11193 signal was generated.

		Signal	Code	Reason
11194				
11195	CX	SIGILL	ILL_ILLOPC	Illegal opcode.
11196			ILL_ILLOPN	Illegal operand.
11197			ILL_ILLADR	Illegal addressing mode.
11198			ILL_ILLTRP	Illegal trap.
11199			ILL_PRVOPC	Privileged opcode.
11200			ILL_PRVREG	Privileged register.
11201			ILL_COPROC	Coprocessor error.
11202			ILL_BADSTK	Internal stack error.
11203		SIGFPE	FPE_INTDIV	Integer divide by zero.
11204			FPE_INTOVF	Integer overflow.
11205			FPE_FLTDIV	Floating-point divide by zero.
11206			FPE_FLTOVF	Floating-point overflow.
11207			FPE_FLTUND	Floating-point underflow.
11208			FPE_FLTRES	Floating-point inexact result.
11209			FPE_FLTINV	Invalid floating-point operation.
11210			FPE_FLTSUB	Subscript out of range.
11211		SIGSEGV	SEGV_MAPERR	Address not mapped to object.
11212			SEGV_ACCERR	Invalid permissions for mapped object.
11213		SIGBUS	BUS_ADRALN	Invalid address alignment.
11214			BUS_ADRERR	Nonexistent physical address.
11215			BUS_OBJERR	Object-specific hardware error.
11216	XSI	SIGTRAP	TRAP_BRKPT	Process breakpoint.
11217			TRAP_TRACE	Process trace trap.
11218	CX	SIGCHLD	CLD_EXITED	Child has exited.
11219			CLD_KILLED	Child has terminated abnormally and did not create a core file.
11220			CLD_DUMPED	Child has terminated abnormally and created a core file.
11221			CLD_TRAPPED	Traced child has trapped.
11222			CLD_STOPPED	Child has stopped.
11223			CLD_CONTINUED	Stopped child has continued.
11224	OB XSR	SIGPOLL	POLL_IN	Data input available.
11225			POLL_OUT	Output buffers available.
11226			POLL_MSG	Input message available.
11227			POLL_ERR	I/O error.
11228			POLL_PRI	High priority input available.
11229			POLL_HUP	Device disconnected.
11230	CX	Any	SI_USER	Signal sent by <i>kill()</i> .
11231			SI_QUEUE	Signal sent by <i>sigqueue()</i> .
11232			SI_TIMER	Signal generated by expiration of a timer set by <i>timer_settime()</i> .
11233			SI_ASYNCIO	Signal generated by completion of an asynchronous I/O request.
11234				
11235			SI_MESGQ	Signal generated by arrival of a message on an empty message queue
11236				
11237	CX	Implementations may support additional <i>si_code</i> values not included in this list, may generate values included in this list under circumstances other than those described in this list, and may contain extensions or limitations that prevent some values from being generated. Implementations do not generate a different value from the ones described in this list for circumstances described in this list.		
11238				
11239				
11240				
11241				

11242 CX In addition, the following signal-specific information shall be available:

Signal	Member	Value
SIGILL SIGFPE	void * <i>si_addr</i>	Address of faulting instruction.
SIGSEGV SIGBUS	void * <i>si_addr</i>	Address of faulting memory reference.
SIGCHLD	pid_t <i>si_pid</i> int <i>si_status</i> uid_t <i>si_uid</i>	Child process ID. Exit value or signal. Real user ID of the process that sent the signal.
SIGPOLL	long <i>si_band</i>	Band event for POLL_IN, POLL_OUT, or POLL_MSG

11252 For some implementations, the value of *si_addr* may be inaccurate.

11253 The following shall be declared as functions and may also be defined as macros. Function
11254 prototypes shall be provided.

```

11255 CX int kill(pid_t, int);
11256 XSI int killpg(pid_t, int);
11257 CX void psiginfo(const siginfo_t *, const char *);
11258 void psignal(int, const char *);
11259 int pthread_kill(pthread_t, int);
11260 int pthread_sigmask(int, const sigset_t *restrict,
11261 sigset_t *restrict);
11262 int raise(int);
11263 CX int sigaction(int, const struct sigaction *restrict,
11264 struct sigaction *restrict);
11265 int sigaddset(sigset_t *, int);
11266 XSI int sigaltstack(const stack_t *restrict, stack_t *restrict);
11267 CX int sigdelset(sigset_t *, int);
11268 int sigemptyset(sigset_t *);
11269 int sigfillset(sigset_t *);
11270 OB XSI int sighold(int);
11271 int sigignore(int);
11272 int siginterrupt(int, int);
11273 CX int sigismember(const sigset_t *, int);
11274 void (*signal(int, void (*)(int)))(int);
11275 OB XSI int sigpause(int);
11276 CX int sigpending(sigset_t *);
11277 int sigprocmask(int, const sigset_t *restrict, sigset_t *restrict);
11278 int sigqueue(pid_t, int, const union sigval);
11279 OB XSI int sigrelse(int);
11280 void (*sigset(int, void (*)(int)))(int);
11281 CX int sigsuspend(const sigset_t *);
11282 int sigtimedwait(const sigset_t *restrict, siginfo_t *restrict,
11283 const struct timespec *restrict);
11284 int sigwait(const sigset_t *restrict, int *restrict);
11285 int sigwaitinfo(const sigset_t *restrict, siginfo_t *restrict);

```

11286 CX Inclusion of the <signal.h> header may make visible all symbols from the <time.h> header.

APPLICATION USAGE

On systems not supporting the XSI option, the *si_pid* and *si_uid* members of **siginfo_t** are only required to be valid when *si_code* is SI_USER or SI_QUEUE. On XSI-conforming systems, they are also valid for all *si_code* values less than or equal to 0; however, it is unspecified whether SI_USER and SI_QUEUE have values less than or equal to zero, and therefore XSI applications should check whether *si_code* has the value SI_USER or SI_QUEUE or is less than or equal to 0 to tell whether *si_pid* and *si_uid* are valid.

RATIONALE

None.

FUTURE DIRECTIONS

The SIGPOLL and SIGPROF signals may be removed in a future version.

SEE ALSO

[<errno.h>](#), [<stropts.h>](#), [<sys/types.h>](#), [<time.h>](#)

XSH Section 2.2 (on page 472), *alarm()*, *ioctl()*, *kill()*, *killpg()*, *psiginfo()*, *pthread_kill()*, *pthread_sigmask()*, *raise()*, *sigaction()*, *sigaddset()*, *sigaltstack()*, *sigdelset()*, *sigemptyset()*, *sigfillset()*, *sighold()*, *siginterrupt()*, *sigismember()*, *signal()*, *sigpending()*, *sigqueue()*, *sigsuspend()*, *sigtimedwait()*, *sigwait()*, *timer_create()*, *wait()*, *waitid()*

CHANGE HISTORY

First released in Issue 1.

Issue 5

The DESCRIPTION is updated for alignment with the POSIX Realtime Extension and the POSIX Threads Extension.

The default action for SIGURG is changed from i to iii. The function prototype for *sigmask()* is removed.

Issue 6

The Open Group Corrigendum U035/2 is applied. In the DESCRIPTION, the wording for abnormal termination is clarified.

The Open Group Corrigendum U028/8 is applied, correcting the prototype for the *sigset()* function.

The Open Group Corrigendum U026/3 is applied, correcting the type of the *sigev_notify_function* function member of the **sigevent** structure.

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The SIGCHLD, SIGCONT, SIGSTOP, SIGTSTP, SIGTTIN, and SIGTTOU signals are now mandated. This is also a FIPS requirement.
- The **pid_t** definition is mandated.

The RT markings are changed to RTS to denote that the semantics are part of the Realtime Signals Extension option.

The **restrict** keyword is added to the prototypes for *sigaction()*, *sigaltstack()*, *sigprocmask()*, *sigtimedwait()*, *sigwait()*, and *sigwaitinfo()*.

IEEE PASC Interpretation 1003.1 #85 is applied, adding the statement that symbols from [<time.h>](#) may be made visible when [<signal.h>](#) is included.

Extensions beyond the ISO C standard are marked.

11330	IEEE Std 1003.1-2001/Cor 1-2002, item XBD/TC1/D6/14 is applied, changing the descriptive
11331	text for members of the sigaction structure.
11332	IEEE Std 1003.1-2001/Cor 1-2002, item XBD/TC1/D6/15 is applied, correcting the definition of
11333	the <i>sa_sigaction</i> member of the sigaction structure.
11334	IEEE Std 1003.1-2001/Cor 2-2004, item XBD/TC2/D6/24 is applied, reworking the ordering of
11335	the siginfo_t type structure in the DESCRIPTION. This is an editorial change and no normative
11336	change is intended.
11337	Issue 7
11338	SD5-XBD-ERN-5 is applied.
11339	SD5-XBD-ERN-39 is applied, removing the sigstack structure which should have been removed
11340	at the same time as the LEGACY <i>sigstack()</i> function.
11341	SD5-XBD-ERN-56 is applied, adding a reference to <sys/types.h> for the size_t type.
11342	Austin Group Interpretation 1003.1-2001 #034 is applied.
11343	The ucontext_t and mcontext_t structures are added here from the obsolescent <ucontext.h>
11344	header.
11345	The <i>psiginfo()</i> and <i>psignal()</i> functions are added from The Open Group Technical Standard, 2006,
11346	Extended API Set Part 1.
11347	The SIGPOLL and SIGPROF signals and text relating to the XSI STREAMS option are marked
11348	obsolescent.
11349	The SA_RESETHAND, SA_RESTART, SA_SIGINFO, SA_NOCLDWAIT, and SA_NODEFER
11350	constants are moved from the XSI option to the Base.
11351	Functionality relating to the Realtime Signals Extension option is moved to the Base.
11352	This reference page is clarified with respect to macros and symbolic constants, and declarations
11353	for the pthread_attr_t , pthread_t , and uid_t types and the timespec structure are added.
11354	SIGRTMIN and SIGRTMAX are required to be positive integer expressions.
11355	The APPLICATION USAGE section is updated to describe the <i>si_pid</i> and <i>si_uid</i> members of
11356	siginfo_t .
11357	POSIX.1-2008, Technical Corrigendum 1, XBD/TC1-2008/0062 [208], XBD/TC1-2008/0063 [80], +
11358	and XBD/TC1-2008/0064 [157] are applied.

11359 NAME

11360 spawn.h — spawn (ADVANCED REALTIME)

11361 SYNOPSIS

11362 SPN `#include <spawn.h>`

11363 DESCRIPTION

11364 The <spawn.h> header shall define the **posix_spawnattr_t** and **posix_spawn_file_actions_t**
 11365 types used in performing spawn operations.

11366 The <spawn.h> header shall define the **mode_t** and **pid_t** types as described in <sys/types.h>.

11367 The <spawn.h> header shall define the **sigset_t** type as described in <signal.h>.

11368 The tag **sched_param** shall be declared as naming an incomplete structure type, the contents of
 11369 which are described in the <sched.h> header.

11370 The <spawn.h> header shall define the following symbolic constants for use as the flags that
 11371 may be set in a **posix_spawnattr_t** object using the *posix_spawnattr_setflags()* function:

11372 POSIX_SPAWN_RESETIDS

11373 POSIX_SPAWN_SETPGROUP

11374 PS POSIX_SPAWN_SETSCHEDPARAM

11375 POSIX_SPAWN_SETSCHEDULER

11376 POSIX_SPAWN_SETSIGDEF

11377 POSIX_SPAWN_SETSIGMASK

11378 The following shall be declared as functions and may also be defined as macros. Function
 11379 prototypes shall be provided.

```

11380 int    posix_spawn(pid_t *restrict, const char *restrict,
11381                  const posix_spawn_file_actions_t *,
11382                  const posix_spawnattr_t *restrict, char *const [restrict],
11383                  char *const [restrict]);
11384 int    posix_spawn_file_actions_addclose(posix_spawn_file_actions_t *,
11385                  int);
11386 int    posix_spawn_file_actions_adddup2(posix_spawn_file_actions_t *,
11387                  int, int);
11388 int    posix_spawn_file_actions_addopen(posix_spawn_file_actions_t *restrict,
11389                  int, const char *restrict, int, mode_t);
11390 int    posix_spawn_file_actions_destroy(posix_spawn_file_actions_t *);
11391 int    posix_spawn_file_actions_init(posix_spawn_file_actions_t *);
11392 int    posix_spawnattr_destroy(posix_spawnattr_t *);
11393 int    posix_spawnattr_getflags(const posix_spawnattr_t *restrict,
11394                  short *restrict);
11395 int    posix_spawnattr_getpgroup(const posix_spawnattr_t *restrict,
11396                  pid_t *restrict);
11397 PS int    posix_spawnattr_getschedparam(const posix_spawnattr_t *restrict,
11398                  struct sched_param *restrict);
11399 int    posix_spawnattr_getschedpolicy(const posix_spawnattr_t *restrict,
11400                  int *restrict);
11401 int    posix_spawnattr_getsigdefault(const posix_spawnattr_t *restrict,
11402                  sigset_t *restrict);
11403 int    posix_spawnattr_getsigmask(const posix_spawnattr_t *restrict,
11404                  sigset_t *restrict);
11405 int    posix_spawnattr_init(posix_spawnattr_t *);

```

```

11406     int    posix_spawnattr_setflags(posix_spawnattr_t *, short);
11407     int    posix_spawnattr_setpgroup(posix_spawnattr_t *, pid_t);
11408 PS     int    posix_spawnattr_setschedparam(posix_spawnattr_t *restrict,
11409         const struct sched_param *restrict);
11410     int    posix_spawnattr_setschedpolicy(posix_spawnattr_t *, int);
11411     int    posix_spawnattr_setsigdefault(posix_spawnattr_t *restrict,
11412         const sigset_t *restrict);
11413     int    posix_spawnattr_setsigmask(posix_spawnattr_t *restrict,
11414         const sigset_t *restrict);
11415     int    posix_spawn(pid_t *restrict, const char *restrict,
11416         const posix_spawn_file_actions_t *,
11417         const posix_spawnattr_t *restrict,
11418         char *const [restrict], char *const [restrict]);

```

11419 Inclusion of the <spawn.h> header may make visible symbols defined in the <sched.h> and
 11420 <signal.h> headers.

11421 APPLICATION USAGE

11422 None.

11423 RATIONALE

11424 None.

11425 FUTURE DIRECTIONS

11426 None.

11427 SEE ALSO

11428 [<sched.h>](#), [<semaphore.h>](#), [<signal.h>](#), [<sys/types.h>](#)

11429 XSH [posix_spawn\(\)](#), [posix_spawn_file_actions_addclose\(\)](#), [posix_spawn_file_actions_adddup2\(\)](#),
 11430 [posix_spawn_file_actions_destroy\(\)](#), [posix_spawnattr_destroy\(\)](#), [posix_spawnattr_getflags\(\)](#),
 11431 [posix_spawnattr_getpgroup\(\)](#), [posix_spawnattr_getschedparam\(\)](#), [posix_spawnattr_getschedpolicy\(\)](#),
 11432 [posix_spawnattr_getsigdefault\(\)](#), [posix_spawnattr_getsigmask\(\)](#)

11433 CHANGE HISTORY

11434 First released in Issue 6. Included for alignment with IEEE Std 1003.1d-1999.

11435 The **restrict** keyword is added to the prototypes for [posix_spawn\(\)](#),
 11436 [posix_spawn_file_actions_addopen\(\)](#), [posix_spawnattr_getsigdefault\(\)](#), [posix_spawnattr_getflags\(\)](#),
 11437 [posix_spawnattr_getpgroup\(\)](#), [posix_spawnattr_getschedparam\(\)](#), [posix_spawnattr_getschedpolicy\(\)](#),
 11438 [posix_spawnattr_getsigmask\(\)](#), [posix_spawnattr_setsigdefault\(\)](#), [posix_spawnattr_setschedparam\(\)](#),
 11439 [posix_spawnattr_setsigmask\(\)](#), and [posix_spawnnp\(\)](#).

11440 Issue 7

11441 This reference page is clarified with respect to macros and symbolic constants, and declarations
 11442 for the **mode_t**, **pid_t**, and **sigset_t** types are added.

11443 NAME

11444 stdarg.h — handle variable argument list

11445 SYNOPSIS

```
11446       #include <stdarg.h>

11447       void va_start(va_list ap, argN);
11448       void va_copy(va_list dest, va_list src);
11449       type va_arg(va_list ap, type);
11450       void va_end(va_list ap);
```

11451 DESCRIPTION

11452 CX The functionality described on this reference page is aligned with the ISO C standard. Any
 11453 conflict between the requirements described here and the ISO C standard is unintentional. This
 11454 volume of POSIX.1-2008 defers to the ISO C standard.

11455 The <stdarg.h> header shall contain a set of macros which allows portable functions that accept
 11456 variable argument lists to be written. Functions that have variable argument lists (such as
 11457 *printf()*) but do not use these macros are inherently non-portable, as different systems use
 11458 different argument-passing conventions.

11459 The <stdarg.h> header shall define the **va_list** type for variables used to traverse the list.

11460 The *va_start()* macro is invoked to initialize *ap* to the beginning of the list before any calls to
 11461 *va_arg()*.

11462 The *va_copy()* macro initializes *dest* as a copy of *src*, as if the *va_start()* macro had been applied
 11463 to *dest* followed by the same sequence of uses of the *va_arg()* macro as had previously been used
 11464 to reach the present state of *src*. Neither the *va_copy()* nor *va_start()* macro shall be invoked to
 11465 reinitialize *dest* without an intervening invocation of the *va_end()* macro for the same *dest*.

11466 The object *ap* may be passed as an argument to another function; if that function invokes the
 11467 *va_arg()* macro with parameter *ap*, the value of *ap* in the calling function is unspecified and shall
 11468 be passed to the *va_end()* macro prior to any further reference to *ap*. The parameter *argN* is the
 11469 identifier of the rightmost parameter in the variable parameter list in the function definition (the
 11470 one just before the *...*). If the parameter *argN* is declared with the **register** storage class, with a
 11471 function type or array type, or with a type that is not compatible with the type that results after
 11472 application of the default argument promotions, the behavior is undefined.

11473 The *va_arg()* macro shall return the next argument in the list pointed to by *ap*. Each invocation
 11474 of *va_arg()* modifies *ap* so that the values of successive arguments are returned in turn. The *type*
 11475 parameter shall be a type name specified such that the type of a pointer to an object that has the
 11476 specified type can be obtained simply by postfixing a '*' to type. If there is no actual next
 11477 argument, or if *type* is not compatible with the type of the actual next argument (as promoted
 11478 according to the default argument promotions), the behavior is undefined, except for the
 11479 following cases:

- 11480 • One type is a signed integer type, the other type is the corresponding unsigned integer
 11481 type, and the value is representable in both types.
- 11482 • One type is a pointer to **void** and the other is a pointer to a character type.
- 11483 XSI • Both types are pointers.

11484 Different types can be mixed, but it is up to the routine to know what type of argument is
 11485 expected.

11486 The *va_end()* macro is used to clean up; it invalidates *ap* for use (unless *va_start()* or *va_copy()* is
 11487 invoked again).

Each invocation of the *va_start()* and *va_copy()* macros shall be matched by a corresponding invocation of the *va_end()* macro in the same function.

Multiple traversals, each bracketed by *va_start()* ... *va_end()*, are possible.

EXAMPLES

This example is a possible implementation of *execl()*:

```
#include <stdarg.h>

#define MAXARGS 31

/*
 * execl is called by
 * execl(file, arg1, arg2, ..., (char *) (0));
 */
int execl(const char *file, const char *args, ...)
{
    va_list ap;
    char *array[MAXARGS + 1];
    int argno = 0;

    va_start(ap, args);
    while (args != 0 && argno < MAXARGS)
    {
        array[argno++] = args;
        args = va_arg(ap, const char *);
    }
    array[argno] = (char *) 0;
    va_end(ap);
    return execv(file, array);
}
```

APPLICATION USAGE

It is up to the calling routine to communicate to the called routine how many arguments there are, since it is not always possible for the called routine to determine this in any other way. For example, *execl()* is passed a null pointer to signal the end of the list. The *printf()* function can tell how many arguments are there by the *format* argument.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

XSH *exec*, *fprintf()*

CHANGE HISTORY

First released in Issue 4. Derived from the ANSI C standard.

Issue 6

This reference page is updated to align with the ISO/IEC 9899:1999 standard.

11529 **NAME**

11530 stdbool.h — boolean type and values

11531 **SYNOPSIS**

11532 #include <stdbool.h>

11533 **DESCRIPTION**

11534 CX The functionality described on this reference page is aligned with the ISO C standard. Any
11535 conflict between the requirements described here and the ISO C standard is unintentional. This
11536 volume of POSIX.1-2008 defers to the ISO C standard.

11537 The <stdbool.h> header shall define the following macros:

11538 bool Expands to **_Bool**.

11539 true Expands to the integer constant 1.

11540 false Expands to the integer constant 0.

11541 __bool_true_false_are_defined

11542 Expands to the integer constant 1.

11543 An application may undefine and then possibly redefine the macros bool, true, and false.

11544 **APPLICATION USAGE**

11545 None.

11546 **RATIONALE**

11547 None.

11548 **FUTURE DIRECTIONS**

11549 The ability to undefine and redefine the macros bool, true, and false is an obsolescent feature
11550 and may be removed in a future version.

11551 **SEE ALSO**

11552 None.

11553 **CHANGE HISTORY**

11554 First released in Issue 6. Included for alignment with the ISO/IEC 9899:1999 standard.

11555 **NAME**

11556 **stddef.h** — standard type definitions

11557 **SYNOPSIS**

11558 #include <stddef.h>

11559 **DESCRIPTION**

11560 CX The functionality described on this reference page is aligned with the ISO C standard. Any
11561 conflict between the requirements described here and the ISO C standard is unintentional. This
11562 volume of POSIX.1-2008 defers to the ISO C standard.

11563 The <stddef.h> header shall define the following macros:

11564 CX **NULL** Null pointer constant. The macro shall expand to an integer constant expression
11565 with the value 0 cast to type **void ***.

11566 offsetof(*type*, *member-designator*)

11567 Integer constant expression of type **size_t**, the value of which is the offset in bytes
11568 to the structure member (*member-designator*), from the beginning of its structure
11569 (*type*).

11570 The <stddef.h> header shall define the following types:

11571 **ptrdiff_t** Signed integer type of the result of subtracting two pointers.

11572 **wchar_t** Integer type whose range of values can represent distinct codes for all members of
11573 the largest extended character set specified among the supported locales; the null
11574 character shall have the code value zero. Each member of the basic character set
11575 shall have a code value equal to its value when used as the lone character in an
11576 integer character constant if an implementation does not define
11577 __STDC_MB_MIGHT_NEQ_WC__.

11578 **size_t** Unsigned integer type of the result of the *sizeof* operator.

11579 The implementation shall support one or more programming environments in which the widths
11580 of **ptrdiff_t**, **size_t**, and **wchar_t** are no greater than the width of type **long**. The names of these
11581 programming environments can be obtained using the *confstr()* function or the *getconf* utility.

11582 **APPLICATION USAGE**

11583 None.

11584 **RATIONALE**

11585 The ISO C standard does not require the **NULL** macro to include the cast to type **void *** and
11586 specifies that the **NULL** macro be implementation-defined. POSIX.1-2008 requires the cast and
11587 therefore need not be implementation-defined.

11588 **FUTURE DIRECTIONS**

11589 None.

11590 **SEE ALSO**

11591 [<sys/types.h>](#), [<wchar.h>](#)

11592 XSH *confstr()*

11593 XCU *getconf*

11594 **CHANGE HISTORY**

11595 First released in Issue 4. Derived from the ANSI C standard.

11596 Issue 7

11597 This reference page is clarified with respect to macros and symbolic constants.

11598 SD5-XBD-ERN-53 is applied, updating the definition of **wchar_t** to align with
11599 ISO/IEC 9899:1999 standard, Technical Corrigendum 3.

11600 **NAME**

11601 stdint.h — integer types

11602 **SYNOPSIS**

11603 #include <stdint.h>

11604 **DESCRIPTION**

11605 CX Some of the functionality described on this reference page extends the ISO C standard.
11606 Applications shall define the appropriate feature test macro (see XSH [Section 2.2](#), on page 472) to
11607 enable the visibility of these symbols in this header.

11608 The <stdint.h> header shall declare sets of integer types having specified widths, and shall
11609 define corresponding sets of macros. It shall also define macros that specify limits of integer
11610 types corresponding to types defined in other standard headers.

11611 **Note:** The “width” of an integer type is the number of bits used to store its value in a pure binary
11612 system; the actual type may use more bits than that (for example, a 28-bit type could be stored
11613 in 32 bits of actual storage). An N -bit signed type has values in the range -2^{N-1} or $1-2^{N-1}$ to
11614 $2^{N-1}-1$, while an N -bit unsigned type has values in the range 0 to 2^N-1 .

11615 Types are defined in the following categories:

- 11616 • Integer types having certain exact widths
- 11617 • Integer types having at least certain specified widths
- 11618 • Fastest integer types having at least certain specified widths
- 11619 • Integer types wide enough to hold pointers to objects
- 11620 • Integer types having greatest width

11621 (Some of these types may denote the same type.)

11622 Corresponding macros specify limits of the declared types and construct suitable constants.

11623 For each type described herein that the implementation provides, the <stdint.h> header shall
11624 declare that **typedef** name and define the associated macros. Conversely, for each type described
11625 herein that the implementation does not provide, the <stdint.h> header shall not declare that
11626 **typedef** name, nor shall it define the associated macros. An implementation shall provide those
11627 types described as required, but need not provide any of the others (described as optional).

11628 **Integer Types**

11629 When **typedef** names differing only in the absence or presence of the initial u are defined, they
11630 shall denote corresponding signed and unsigned types as described in the ISO/IEC 9899:1999
11631 standard, Section 6.2.5; an implementation providing one of these corresponding types shall also
11632 provide the other.

11633 In the following descriptions, the symbol N represents an unsigned decimal integer with no
11634 leading zeros (for example, 8 or 24, but not 04 or 048).

- 11635 • Exact-width integer types

11636 The **typedef** name **intN_t** designates a signed integer type with width N , no padding bits,
11637 and a two’s-complement representation. Thus, **int8_t** denotes a signed integer type with a
11638 width of exactly 8 bits.

11639 The **typedef** name **uintN_t** designates an unsigned integer type with width N . Thus,
11640 **uint24_t** denotes an unsigned integer type with a width of exactly 24 bits.

11641 CX The following types are required:

11642 **int8_t**
 11643 **int16_t**
 11644 **int32_t**
 11645 **uint8_t**
 11646 **uint16_t**
 11647 **uint32_t**

11648 If an implementation provides integer types with width 64 that meet these requirements,
 11649 then the following types are required:

11650 **int64_t**
 11651 **uint64_t**

11652 CX In particular, this will be the case if any of the following are true:

- 11653 — The implementation supports the `_POSIX_V7_ILP32_OFFBIG` programming
 11654 environment and the application is being built in the `_POSIX_V7_ILP32_OFFBIG`
 11655 programming environment (see the Shell and Utilities volume of POSIX.1-2008, *c99*,
 11656 Programming Environments).
- 11657 — The implementation supports the `_POSIX_V7_LP64_OFF64` programming
 11658 environment and the application is being built in the `_POSIX_V7_LP64_OFF64`
 11659 programming environment.
- 11660 — The implementation supports the `_POSIX_V7_LPBIG_OFFBIG` programming
 11661 environment and the application is being built in the `_POSIX_V7_LPBIG_OFFBIG`
 11662 programming environment.

11663 All other types of this form are optional.

11664 • Minimum-width integer types

11665 The **typedef** name **int_leastN_t** designates a signed integer type with a width of at least *N*,
 11666 such that no signed integer type with lesser size has at least the specified width. Thus,
 11667 **int_least32_t** denotes a signed integer type with a width of at least 32 bits.

11668 The **typedef** name **uint_leastN_t** designates an unsigned integer type with a width of at
 11669 least *N*, such that no unsigned integer type with lesser size has at least the specified width.
 11670 Thus, **uint_least16_t** denotes an unsigned integer type with a width of at least 16 bits.

11671 The following types are required:

11672 **int_least8_t**
 11673 **int_least16_t**
 11674 **int_least32_t**
 11675 **int_least64_t**
 11676 **uint_least8_t**
 11677 **uint_least16_t**
 11678 **uint_least32_t**
 11679 **uint_least64_t**

11680 All other types of this form are optional.

11681 • Fastest minimum-width integer types

11682 Each of the following types designates an integer type that is usually fastest to operate
11683 with among all integer types that have at least the specified width.

11684 The designated type is not guaranteed to be fastest for all purposes; if the implementation
11685 has no clear grounds for choosing one type over another, it will simply pick some integer
11686 type satisfying the signedness and width requirements.

11687 The **typedef** name **int_fastN_t** designates the fastest signed integer type with a width of at
11688 least *N*. The **typedef** name **uint_fastN_t** designates the fastest unsigned integer type with
11689 a width of at least *N*.

11690 The following types are required:

11691 **int_fast8_t**
11692 **int_fast16_t**
11693 **int_fast32_t**
11694 **int_fast64_t**
11695 **uint_fast8_t**
11696 **uint_fast16_t**
11697 **uint_fast32_t**
11698 **uint_fast64_t**

11699 All other types of this form are optional.

11700 • Integer types capable of holding object pointers

11701 The following type designates a signed integer type with the property that any valid
11702 pointer to **void** can be converted to this type, then converted back to a pointer to **void**, and
11703 the result will compare equal to the original pointer:

11704 **intptr_t**

11705 The following type designates an unsigned integer type with the property that any valid
11706 pointer to **void** can be converted to this type, then converted back to a pointer to **void**, and
11707 the result will compare equal to the original pointer:

11708 **uintptr_t**

11709 XSI On XSI-conformant systems, the **intptr_t** and **uintptr_t** types are required; otherwise, they
11710 are optional.

11711 • Greatest-width integer types

11712 The following type designates a signed integer type capable of representing any value of
11713 any signed integer type:

11714 **intmax_t**

11715 The following type designates an unsigned integer type capable of representing any value
11716 of any unsigned integer type:

11717 **uintmax_t**

11718 These types are required.

11719 **Note:** Applications can test for optional types by using the corresponding limit macro from [Limits of](#)
 11720 [Specified-Width Integer Types](#).

11721 **Limits of Specified-Width Integer Types**

11722 The following macros specify the minimum and maximum limits of the types declared in the
 11723 **<stdint.h>** header. Each macro name corresponds to a similar type name in [Integer Types](#) (on
 11724 page 347).

11725 Each instance of any defined macro shall be replaced by a constant expression suitable for use in
 11726 **#if** preprocessing directives, and this expression shall have the same type as would an
 11727 expression that is an object of the corresponding type converted according to the integer
 11728 promotions. Its implementation-defined value shall be equal to or greater in magnitude
 11729 (absolute value) than the corresponding value given below, with the same sign, except where
 11730 stated to be exactly the given value.

- 11731 • Limits of exact-width integer types
 - 11732 — Minimum values of exact-width signed integer types:
 - 11733 {INTN_MIN} Exactly $-(2^{N-1})$
 - 11734 — Maximum values of exact-width signed integer types:
 - 11735 {INTN_MAX} Exactly $2^{N-1} - 1$
 - 11736 — Maximum values of exact-width unsigned integer types:
 - 11737 {UINTN_MAX} Exactly $2^N - 1$
- 11738 • Limits of minimum-width integer types
 - 11739 — Minimum values of minimum-width signed integer types:
 - 11740 {INT_LEASTN_MIN} $-(2^{N-1} - 1)$
 - 11741 — Maximum values of minimum-width signed integer types:
 - 11742 {INT_LEASTN_MAX} $2^{N-1} - 1$
 - 11743 — Maximum values of minimum-width unsigned integer types:
 - 11744 {UINT_LEASTN_MAX} $2^N - 1$
- 11745 • Limits of fastest minimum-width integer types
 - 11746 — Minimum values of fastest minimum-width signed integer types:
 - 11747 {INT_FASTN_MIN} $-(2^{N-1} - 1)$
 - 11748 — Maximum values of fastest minimum-width signed integer types:
 - 11749 {INT_FASTN_MAX} $2^{N-1} - 1$
 - 11750 — Maximum values of fastest minimum-width unsigned integer types:
 - 11751 {UINT_FASTN_MAX} $2^N - 1$
- 11752 • Limits of integer types capable of holding object pointers
 - 11753 — Minimum value of pointer-holding signed integer type:
 - 11754 {INTPTR_MIN} $-(2^{15} - 1)$

- 11755 — Maximum value of pointer-holding signed integer type:
- 11756 {INTPTR_MAX} $2^{15} - 1$
- 11757 — Maximum value of pointer-holding unsigned integer type:
- 11758 {UINTPTR_MAX} $2^{16} - 1$
- 11759 • Limits of greatest-width integer types
- 11760 — Minimum value of greatest-width signed integer type:
- 11761 {INTMAX_MIN} $-(2^{63} - 1)$
- 11762 — Maximum value of greatest-width signed integer type:
- 11763 {INTMAX_MAX} $2^{63} - 1$
- 11764 — Maximum value of greatest-width unsigned integer type:
- 11765 {UINTMAX_MAX} $2^{64} - 1$

11766 Limits of Other Integer Types

11767 The following macros specify the minimum and maximum limits of integer types corresponding
11768 to types defined in other standard headers.

11769 Each instance of these macros shall be replaced by a constant expression suitable for use in **#if**
11770 preprocessing directives, and this expression shall have the same type as would an expression
11771 that is an object of the corresponding type converted according to the integer promotions. Its
11772 implementation-defined value shall be equal to or greater in magnitude (absolute value) than
11773 the corresponding value given below, with the same sign.

- 11774 • Limits of **ptrdiff_t**:
- 11775 {PTRDIFF_MIN} −65 535
- 11776 {PTRDIFF_MAX} +65 535
- 11777 • Limits of **sig_atomic_t**:
- 11778 {SIG_ATOMIC_MIN} See below.
- 11779 {SIG_ATOMIC_MAX} See below.
- 11780 • Limit of **size_t**:
- 11781 {SIZE_MAX} 65 535
- 11782 • Limits of **wchar_t**:
- 11783 {WCHAR_MIN} See below.
- 11784 {WCHAR_MAX} See below.
- 11785 • Limits of **wint_t**:
- 11786 {WINT_MIN} See below.
- 11787 {WINT_MAX} See below.

11788 If **sig_atomic_t** (see the <signal.h> header) is defined as a signed integer type, the value of
11789 {SIG_ATOMIC_MIN} shall be no greater than −127 and the value of {SIG_ATOMIC_MAX} shall
11790 be no less than 127; otherwise, **sig_atomic_t** shall be defined as an unsigned integer type, and
11791 the value of {SIG_ATOMIC_MIN} shall be 0 and the value of {SIG_ATOMIC_MAX} shall be no

11792 less than 255.

11793 If **wchar_t** (see the <stddef.h> header) is defined as a signed integer type, the value of
 11794 {WCHAR_MIN} shall be no greater than -127 and the value of {WCHAR_MAX} shall be no less
 11795 than 127; otherwise, **wchar_t** shall be defined as an unsigned integer type, and the value of
 11796 {WCHAR_MIN} shall be 0 and the value of {WCHAR_MAX} shall be no less than 255.

11797 If **wint_t** (see the <wchar.h> header) is defined as a signed integer type, the value of
 11798 {WINT_MIN} shall be no greater than -32767 and the value of {WINT_MAX} shall be no less
 11799 than 32767; otherwise, **wint_t** shall be defined as an unsigned integer type, and the value of
 11800 {WINT_MIN} shall be 0 and the value of {WINT_MAX} shall be no less than 65535.

11801 **Macros for Integer Constant Expressions**

11802 The following macros expand to integer constant expressions suitable for initializing objects that
 11803 have integer types corresponding to types defined in the <stdint.h> header. Each macro name
 11804 corresponds to a similar type name listed under *Minimum-width integer types* and *Greatest-width*
 11805 *integer types*.

11806 Each invocation of one of these macros shall expand to an integer constant expression suitable
 11807 for use in **#if** preprocessing directives. The type of the expression shall have the same type as
 11808 would an expression that is an object of the corresponding type converted according to the
 11809 integer promotions. The value of the expression shall be that of the argument.

11810 The argument in any instance of these macros shall be an unsuffixed integer constant with a
 11811 value that does not exceed the limits for the corresponding type.

- 11812 • Macros for minimum-width integer constant expressions

11813 The macro **INTN_C(value)** shall expand to an integer constant expression corresponding to
 11814 the type **int_leastN_t**. The macro **UINTN_C(value)** shall expand to an integer constant
 11815 expression corresponding to the type **uint_leastN_t**. For example, if **uint_least64_t** is a
 11816 name for the type **unsigned long long**, then **UINT64_C(0x123)** might expand to the integer
 11817 constant **0x123ULL**.

- 11818 • Macros for greatest-width integer constant expressions

11819 The following macro expands to an integer constant expression having the value specified
 11820 by its argument and the type **intmax_t**:

11821 **INTMAX_C(value)**

11822 The following macro expands to an integer constant expression having the value specified
 11823 by its argument and the type **uintmax_t**:

11824 **UINTMAX_C(value)**

11825 **APPLICATION USAGE**

11826 None.

11827 **RATIONALE**

11828 The <stdint.h> header is a subset of the <inttypes.h> header more suitable for use in
 11829 freestanding environments, which might not support the formatted I/O functions. In some
 11830 environments, if the formatted conversion support is not wanted, using this header instead of
 11831 the <inttypes.h> header avoids defining such a large number of macros.

11832 As a consequence of adding **int8_t**, the following are true:

11833 • A byte is exactly 8 bits.

11834 • {CHAR_BIT} has the value 8, {SCHAR_MAX} has the value 127, {SCHAR_MIN} has the

11835 value -128, and {UCHAR_MAX} has the value 255.

11836 (The POSIX standard explicitly requires 8-bit char and two's-complement arithmetic.)

11837 **FUTURE DIRECTIONS**

11838 **typedef** names beginning with **int** or **uint** and ending with **_t** may be added to the types defined

11839 in the **<stdint.h>** header. Macro names beginning with **INT** or **UINT** and ending with **_MAX**,

11840 **_MIN**, or **_C** may be added to the macros defined in the **<stdint.h>** header.

11841 **SEE ALSO**

11842 [**<inttypes.h>**](#), [**<signal.h>**](#), [**<stddef.h>**](#), [**<wchar.h>**](#)

11843 XSH [Section 2.2](#) (on page 472)

11844 **CHANGE HISTORY**

11845 First released in Issue 6. Included for alignment with the ISO/IEC 9899:1999 standard.

11846 ISO/IEC 9899:1999 standard, Technical Corrigendum 1 is applied.

11847 **Issue 7**

11848 ISO/IEC 9899:1999 standard, Technical Corrigendum 3 #40 is applied.

11849 SD5-XBD-ERN-67 is applied.

11850 **NAME**

11851 stdio.h — standard buffered input/output

11852 **SYNOPSIS**

11853 #include <stdio.h>

11854 **DESCRIPTION**

11855 CX Some of the functionality described on this reference page extends the ISO C standard.
 11856 Applications shall define the appropriate feature test macro (see XSH [Section 2.2](#), on page 472) to
 11857 enable the visibility of these symbols in this header.

11858 The <stdio.h> header shall define the following data types through **typedef**:

11859	FILE	A structure containing information about a file.
11860	fpos_t	A non-array type containing all information needed to specify uniquely every position within a file.
11861		
11862	off_t	As described in <sys/types.h>.
11863	size_t	As described in <stddef.h>.
11864	CX ssize_t	As described in <sys/types.h>.
11865	CX va_list	As described in <stdarg.h>.

11866 The <stdio.h> header shall define the following macros which shall expand to integer constant
 11867 expressions:

11868	CX BUFSIZ	Size of <stdio.h> buffers. This shall expand to a positive value.
11869	CX L_ctermid	Maximum size of character array to hold <i>ctermid()</i> output.
11870	OB L_tmpnam	Maximum size of character array to hold <i>tmpnam()</i> output.

11871 The <stdio.h> header shall define the following macros which shall expand to integer constant
 11872 expressions with distinct values:

11873	_IOFBF	Input/output fully buffered.
11874	_IOLBF	Input/output line buffered.
11875	_IONBF	Input/output unbuffered.

11876 The <stdio.h> header shall define the following macros which shall expand to integer constant
 11877 expressions with distinct values:

11878	SEEK_CUR	Seek relative to current position.
11879	SEEK_END	Seek relative to end-of-file.
11880	SEEK_SET	Seek relative to start-of-file.

11881 The <stdio.h> header shall define the following macros which shall expand to integer constant
 11882 expressions denoting implementation limits:

11883	{FILENAME_MAX}	Maximum size in bytes of the longest pathname that the implementation guarantees can be opened.
11884		
11885	{FOPEN_MAX}	Number of streams which the implementation guarantees can be open simultaneously. The value is at least eight.
11886		
11887	OB {TMP_MAX}	Minimum number of unique filenames generated by <i>tmpnam()</i> . Maximum number of times an application can call <i>tmpnam()</i> reliably. The value of {TMP_MAX} is at least 25.
11888		
11889		

11890	OB XSI	On XSI-conformant systems, the value of {TMP_MAX} is at least 10 000.
11891		The <stdio.h> header shall define the following macro which shall expand to an integer constant
11892		expression with type int and a negative value:
11893		EOF End-of-file return value.
11894		The <stdio.h> header shall define NULL as described in <stddef.h>.
11895		The <stdio.h> header shall define the following macro which shall expand to a string constant:
11896	OB XSI	P_tmpdir Default directory prefix for <i>tempnam()</i> .
11897		The <stdio.h> header shall define the following macros which shall expand to expressions of
11898		type “pointer to FILE ” that point to the FILE objects associated, respectively, with the standard
11899		error, input, and output streams:
11900		<i>stderr</i> Standard error output stream.
11901		<i>stdin</i> Standard input stream.
11902		<i>stdout</i> Standard output stream.
11903		The following shall be declared as functions and may also be defined as macros. Function
11904		prototypes shall be provided.
11905		void clearerr(FILE *);
11906	CX	char *ctermid(char *);
11907		int dprintf(int, const char *restrict, ...)
11908		int fclose(FILE *);
11909	CX	FILE *fdopen(int, const char *);
11910		int feof(FILE *);
11911		int ferror(FILE *);
11912		int fflush(FILE *);
11913		int fgetc(FILE *);
11914		int fgetpos(FILE *restrict, fpos_t *restrict);
11915		char *fgets(char *restrict, int, FILE *restrict);
11916	CX	int fileno(FILE *);
11917		void flockfile(FILE *);
11918		FILE *fmemopen(void *restrict, size_t, const char *restrict);
11919		FILE *fopen(const char *restrict, const char *restrict);
11920		int fprintf(FILE *restrict, const char *restrict, ...);
11921		int fputc(int, FILE *);
11922		int fputs(const char *restrict, FILE *restrict);
11923		size_t fread(void *restrict, size_t, size_t, FILE *restrict);
11924		FILE *freopen(const char *restrict, const char *restrict,
11925		FILE *restrict);
11926		int fscanf(FILE *restrict, const char *restrict, ...);
11927		int fseek(FILE *, long, int);
11928	CX	int fseeko(FILE *, off_t, int);
11929		int fsetpos(FILE *, const fpos_t *);
11930		long ftell(FILE *);
11931	CX	off_t ftello(FILE *);
11932		int ftrylockfile(FILE *);
11933		void funlockfile(FILE *);
11934		size_t fwrite(const void *restrict, size_t, size_t, FILE *restrict);
11935		int getc(FILE *);

```

11936      int      getchar(void);
11937 CX      int     getc_unlocked(FILE *);
11938      int      getchar_unlocked(void);
11939      ssize_t   getdelim(char **restrict, size_t *restrict, int,
11940                        FILE *restrict);
11941      ssize_t   getline(char **restrict, size_t *restrict, FILE *restrict);
11942 OB      char     *gets(char *);
11943 CX      FILE     *open_memstream(char **, size_t *);
11944      int      pclose(FILE *);
11945      void      perror(const char *);
11946 CX      FILE     *popen(const char *, const char *);
11947      int      printf(const char *restrict, ...);
11948      int      putc(int, FILE *);
11949      int      putchar(int);
11950 CX      int      putc_unlocked(int, FILE *);
11951      int      putchar_unlocked(int);
11952      int      puts(const char *);
11953      int      remove(const char *);
11954      int      rename(const char *, const char *);
11955 CX      int      renameat(int, const char *, int, const char *);
11956      void      rewind(FILE *);
11957      int      scanf(const char *restrict, ...);
11958      void      setbuf(FILE *restrict, char *restrict);
11959      int      setvbuf(FILE *restrict, char *restrict, int, size_t);
11960      int      snprintf(char *restrict, size_t, const char *restrict, ...);
11961      int      sprintf(char *restrict, const char *restrict, ...);
11962      int      sscanf(const char *restrict, const char *restrict, ...);
11963 OB XSI     char     *tempnam(const char *, const char *);
11964      FILE     *tmpfile(void);
11965 OB      char     *tmpnam(char *);
11966      int      ungetc(int, FILE *);
11967 CX      int      vdprintf(int, const char *restrict, va_list);
11968      int      vfprintf(FILE *restrict, const char *restrict, va_list);
11969      int      vfscanf(FILE *restrict, const char *restrict, va_list);
11970      int      vprintf(const char *restrict, va_list);
11971      int      vscanf(const char *restrict, va_list);
11972      int      vsnprintf(char *restrict, size_t, const char *restrict,
11973                        va_list);
11974      int      vsprintf(char *restrict, const char *restrict, va_list);
11975      int      vsscanf(const char *restrict, const char *restrict, va_list);
11976 CX      Inclusion of the <stdio.h> header may also make visible all symbols from <stddef.h>.

```

11977 APPLICATION USAGE

11978 Since standard I/O streams may use an underlying file descriptor to access the file associated
11979 with a stream, application developers need to be aware that {FOPEN_MAX} streams may not be
11980 available if file descriptors are being used to access files that are not associated with streams.

11981 RATIONALE

11982 There is a conflict between the ISO C standard and the POSIX definition of the {TMP_MAX}
11983 macro that is addressed by ISO/IEC 9899:1999 standard, Defect Report 336. The POSIX standard
11984 is in alignment with the public record of the response to the Defect Report. This change has not
11985 yet been published as part of the ISO C standard.

11986 **FUTURE DIRECTIONS**

11987 None.

11988 **SEE ALSO**

11989 [<stdarg.h>](#), [<stddef.h>](#), [<sys/types.h>](#)

11990 XSH Section 2.2 (on page 472), [clearerr\(\)](#), [ctermid\(\)](#), [fclose\(\)](#), [fdopen\(\)](#), [feof\(\)](#), [ferror\(\)](#), [fflush\(\)](#),
11991 [fgetc\(\)](#), [fgetpos\(\)](#), [fgets\(\)](#), [fileno\(\)](#), [flockfile\(\)](#), [fmemopen\(\)](#), [fopen\(\)](#), [fprintf\(\)](#), [fputc\(\)](#), [fputs\(\)](#), [fread\(\)](#),
11992 [freopen\(\)](#), [fscanf\(\)](#), [fseek\(\)](#), [fsetpos\(\)](#), [ftell\(\)](#), [fwrite\(\)](#), [getc\(\)](#), [getchar\(\)](#), [getc_unlocked\(\)](#), [getdelim\(\)](#),
11993 [getopt\(\)](#), [gets\(\)](#), [open_memstream\(\)](#), [pclose\(\)](#), [perror\(\)](#), [popen\(\)](#), [putc\(\)](#), [putchar\(\)](#), [puts\(\)](#), [remove\(\)](#),
11994 [rename\(\)](#), [rewind\(\)](#), [setbuf\(\)](#), [setvbuf\(\)](#), [stdin](#), [system\(\)](#), [tempnam\(\)](#), [tmpfile\(\)](#), [tmpnam\(\)](#), [ungetc\(\)](#),
11995 [vfprintf\(\)](#), [vfscanf\(\)](#)

11996 **CHANGE HISTORY**

11997 First released in Issue 1. Derived from Issue 1 of the SVID.

11998 **Issue 5**

11999 The DESCRIPTION is updated for alignment with the POSIX Threads Extension.

12000 Large File System extensions are added.

12001 The constant `L_cuserid` and the external variables `optarg`, `opterr`, `optind`, and `optopt` are marked as
12002 extensions and LEGACY.

12003 The `cuserid()` and `getopt()` functions are marked LEGACY.

12004 **Issue 6**

12005 The constant `L_cuserid` and the external variables `optarg`, `opterr`, `optind`, and `optopt` are removed
12006 as they were previously marked LEGACY.

12007 The `cuserid()`, `getopt()`, and `getw()` functions are removed as they were previously marked
12008 LEGACY.

12009 Several functions are marked as part of the Thread-Safe Functions option.

12010 This reference page is updated to align with the ISO/IEC 9899:1999 standard. Note that the
12011 description of the `fpos_t` type is now explicitly updated to exclude array types.

12012 Extensions beyond the ISO C standard are marked.

12013 **Issue 7**

12014 Austin Group Interpretation 1003.1-2001 #172 is applied, adding rationale about a conflict for the
12015 definition of `{TMP_MAX}` with the ISO C standard.

12016 SD5-XBD-ERN-99 is applied, adding APPLICATION USAGE.

12017 The `dprintf()`, `fmemopen()`, `getdelim()`, `getline()`, `open_memstream()`, and `vdprintf()` functions are
12018 added from The Open Group Technical Standard, 2006, Extended API Set Part 1.

12019 The `renameat()` function is added from The Open Group Technical Standard, 2006, Extended API
12020 Set Part 2.

12021 The `gets()`, `tmpnam()`, and `tempnam()` functions and the `L_tmpnam` macro are marked
12022 obsolescent.

12023 This reference page is clarified with respect to macros and symbolic constants, and a declaration
12024 for the `off_t` type is added.

12025 POSIX.1-2008, Technical Corrigendum 1, XBD/TC1-2008/0065 [291,427] is applied.

+

12026 **NAME**

12027 stdlib.h — standard library definitions

12028 **SYNOPSIS**

12029 #include <stdlib.h>

12030 **DESCRIPTION**

12031 CX Some of the functionality described on this reference page extends the ISO C standard.
 12032 Applications shall define the appropriate feature test macro (see XSH [Section 2.2](#), on page 472) to
 12033 enable the visibility of these symbols in this header.

12034 The <stdlib.h> header shall define the following macros which shall expand to integer constant
 12035 expressions:

12036 EXIT_FAILURE Unsuccessful termination for *exit()*; evaluates to a non-zero value.

12037 EXIT_SUCCESS Successful termination for *exit()*; evaluates to 0.

12038 {RAND_MAX} Maximum value returned by *rand()*; at least 32 767.

12039 The <stdlib.h> header shall define the following macro which shall expand to a positive integer
 12040 expression with type **size_t**:

12041 {MB_CUR_MAX} Maximum number of bytes in a character specified by the current locale
 12042 (category *LC_CTYPE*).

12043 The <stdlib.h> header shall define NULL as described in <stddef.h>.

12044 The <stdlib.h> header shall define the following data types through **typedef**:

12045 **div_t** Structure type returned by the *div()* function.

12046 **ldiv_t** Structure type returned by the *ldiv()* function.

12047 **lldiv_t** Structure type returned by the *lldiv()* function.

12048 **size_t** As described in <stddef.h>.

12049 **wchar_t** As described in <stddef.h>.

12050 CX In addition, the <stdlib.h> header shall define the following symbolic constants and macros as
 12051 described in <sys/wait.h>:

12052 WEXITSTATUS
 12053 WIFEXITED
 12054 WIFSIGNALED
 12055 WIFSTOPPED
 12056 WNOHANG
 12057 WSTOPSIG
 12058 WTERMSIG
 12059 WUNTRACED

12060 The following shall be declared as functions and may also be defined as macros. Function
 12061 prototypes shall be provided.

12062 void _Exit(int);
 12063 XSI long a64l(const char *);
 12064 void abort(void);
 12065 int abs(int);
 12066 int atexit(void (*)(void));
 12067 double atof(const char *);

12068		int	atoi(const char *);
12069		long	atol(const char *);
12070		long long	atoll(const char *);
12071		void	*bsearch(const void *, const void *, size_t, size_t, int (*)(const void *, const void *));
12072			
12073		void	*calloc(size_t, size_t);
12074		div_t	div(int, int);
12075	XSI	double	drand48(void);
12076		double	erand48(unsigned short [3]);
12077		void	exit(int);
12078		void	free(void *);
12079		char	*getenv(const char *);
12080	CX	int	getsubopt(char **, char *const *, char **);
12081	XSI	int	grantpt(int);
12082		char	*initstate(unsigned, char *, size_t);
12083		long	jrand48(unsigned short [3]);
12084		char	*l64a(long);
12085		long	labs(long);
12086	XSI	void	lcong48(unsigned short [7]);
12087		ldiv_t	ldiv(long, long);
12088		long long	llabs(long long);
12089		lldiv_t	lldiv(long long, long long);
12090	XSI	long	lrand48(void);
12091		void	*malloc(size_t);
12092		int	mblen(const char *, size_t);
12093		size_t	mbstowcs(wchar_t *restrict, const char *restrict, size_t);
12094		int	mbtowc(wchar_t *restrict, const char *restrict, size_t);
12095	CX	char	*mkdtemp(char *);
12096		int	mkstemp(char *);
12097	XSI	long	mrnd48(void);
12098		long	nrnd48(unsigned short [3]);
12099	ADV	int	posix_memalign(void **, size_t, size_t);
12100	XSI	int	posix_openpt(int);
12101		char	*ptsname(int);
12102		int	putenv(char *);
12103		void	qsort(void *, size_t, size_t, int (*)(const void *, const void *));
12104			
12105		int	rand(void);
12106	OB CX	int	rand_r(unsigned *);
12107	XSI	long	random(void);
12108		void	*realloc(void *, size_t);
12109	XSI	char	*realpath(const char *restrict, char *restrict);
12110		unsigned short	*seed48(unsigned short [3]);
12111	CX	int	setenv(const char *, const char *, int);
12112	XSI	void	setkey(const char *);
12113		char	*setstate(char *);
12114		void	srand(unsigned);
12115	XSI	void	srand48(long);
12116		void	srandom(unsigned);
12117		double	strtod(const char *restrict, char **restrict);
12118		float	strtof(const char *restrict, char **restrict);
12119		long	strtol(const char *restrict, char **restrict, int);

```

12120     long double    strtold(const char *restrict, char **restrict);
12121     long long      strtoll(const char *restrict, char **restrict, int);
12122     unsigned long   strtoul(const char *restrict, char **restrict, int);
12123     unsigned long long
12124     strtoull(const char *restrict, char **restrict, int);
12125     int             system(const char *);
12126 XSI    int         unlockpt(int);
12127 CX     int         unsetenv(const char *);
12128     size_t          wcstombs(char *restrict, const wchar_t *restrict, size_t);
12129     int             wctomb(char *, wchar_t);

12130 CX     Inclusion of the <stdlib.h> header may also make visible all symbols from <stddef.h>,
12131         <limits.h>, <math.h>, and <sys/wait.h>.

```

12132 APPLICATION USAGE

12133 None.

12134 RATIONALE

12135 None.

12136 FUTURE DIRECTIONS

12137 None.

12138 SEE ALSO

12139 [<limits.h>](#), [<math.h>](#), [<stddef.h>](#), [<sys/types.h>](#), [<sys/wait.h>](#)

12140 XSH Section 2.2 (on page 472), [_Exit\(\)](#), [a64l\(\)](#), [abort\(\)](#), [abs\(\)](#), [atexit\(\)](#), [atof\(\)](#), [atoi\(\)](#), [atol\(\)](#),
12141 [bsearch\(\)](#), [calloc\(\)](#), [div\(\)](#), [drand48\(\)](#), [exit\(\)](#), [free\(\)](#), [getenv\(\)](#), [getsubopt\(\)](#), [grantpt\(\)](#), [initstate\(\)](#), [labs\(\)](#),
12142 [ldiv\(\)](#), [malloc\(\)](#), [mblen\(\)](#), [mbstowcs\(\)](#), [mbtowc\(\)](#), [mkdtemp\(\)](#), [posix_memalign\(\)](#), [posix_openpt\(\)](#),
12143 [ptsname\(\)](#), [putenv\(\)](#), [qsort\(\)](#), [rand\(\)](#), [realloc\(\)](#), [realpath\(\)](#), [setenv\(\)](#), [setkey\(\)](#), [strtod\(\)](#), [strtol\(\)](#),
12144 [strtoul\(\)](#), [system\(\)](#), [unlockpt\(\)](#), [unsetenv\(\)](#), [wcstombs\(\)](#), [wctomb\(\)](#)

12145 CHANGE HISTORY

12146 First released in Issue 3.

12147 Issue 5

12148 The DESCRIPTION is updated for alignment with the POSIX Threads Extension.

12149 The [ttyslot\(\)](#) and [valloc\(\)](#) functions are marked LEGACY.

12150 The type of the third argument to [initstate\(\)](#) is changed from **int** to **size_t**. The type of the return
12151 value from [setstate\(\)](#) is changed from **char** to **char ***, and the type of the first argument is
12152 changed from **char *** to **const char ***.

12153 Issue 6

12154 The Open Group Corrigendum U021/1 is applied, correcting the prototype for [realpath\(\)](#) to be
12155 consistent with the reference page.

12156 The Open Group Corrigendum U028/13 is applied, correcting the prototype for [putenv\(\)](#) to be
12157 consistent with the reference page.

12158 The [rand_r\(\)](#) function is marked as part of the Thread-Safe Functions option.

12159 Function prototypes for [setenv\(\)](#) and [unsetenv\(\)](#) are added.

12160 The [posix_memalign\(\)](#) function is added for alignment with IEEE Std 1003.1d-1999.

12161 This reference page is updated to align with the ISO/IEC 9899:1999 standard.

12162 The [ecvt\(\)](#), [fcvt\(\)](#), [gcvt\(\)](#), and [mktemp\(\)](#) functions are marked LEGACY.

- 12163 The *ttyslot()* and *valloc()* functions are removed as they were previously marked LEGACY.
- 12164 Extensions beyond the ISO C standard are marked.
- 12165 **Issue 7**
- 12166 SD5-XBD-ERN-79 and SD5-XBD-ERN-105 are applied.
- 12167 The LEGACY functions are removed.
- 12168 The *mkdtemp()* function is added from The Open Group Technical Standard, 2006, Extended API
- 12169 Set Part 1.
- 12170 The *rand_r()* function is marked obsolescent.
- 12171 This reference page is clarified with respect to macros and symbolic constants.
- 12172 The type of the first argument to *setstate()* is changed from **const char *** to **char ***.
- 12173 POSIX.1-2008, Technical Corrigendum 1, XBD/TC1-2008/0066 [197] is applied. +

12174 NAME

12175 string.h — string operations

12176 SYNOPSIS

12177 #include <string.h>

12178 DESCRIPTION

12179 CX Some of the functionality described on this reference page extends the ISO C standard.
 12180 Applications shall define the appropriate feature test macro (see XSH [Section 2.2](#), on page 472) to
 12181 enable the visibility of these symbols in this header.

12182 The <string.h> header shall define NULL and **size_t** as described in <stddef.h>.

12183 CX The <string.h> header shall define the **locale_t** type as described in <locale.h>.

12184 The following shall be declared as functions and may also be defined as macros. Function
 12185 prototypes shall be provided for use with ISO C standard compilers.

```

12186 XSI void *memcpy(void *restrict, const void *restrict, int, size_t);
12187 void *memchr(const void *, int, size_t);
12188 int memcmp(const void *, const void *, size_t);
12189 void *memcpy(void *restrict, const void *restrict, size_t);
12190 void *memmove(void *, const void *, size_t);
12191 void *memset(void *, int, size_t);
12192 CX char *strcpy(char *restrict, const char *restrict);
12193 char *strncpy(char *restrict, const char *restrict, size_t);
12194 char *strcat(char *restrict, const char *restrict);
12195 char *strchr(const char *, int);
12196 int strcmp(const char *, const char *);
12197 int strcoll(const char *, const char *);
12198 CX int strcoll_l(const char *, const char *, locale_t);
12199 char *strcpy(char *restrict, const char *restrict);
12200 size_t strcspn(const char *, const char *);
12201 CX char *strdup(const char *);
12202 char *strerror(int);
12203 CX char *strerror_l(int, locale_t);
12204 int strerror_r(int, char *, size_t);
12205 size_t strlen(const char *);
12206 char *strncat(char *restrict, const char *restrict, size_t);
12207 int strncmp(const char *, const char *, size_t);
12208 char *strncpy(char *restrict, const char *restrict, size_t);
12209 CX char *strndup(const char *, size_t);
12210 size_t strnlen(const char *, size_t);
12211 char *strpbrk(const char *, const char *);
12212 char *strrchr(const char *, int);
12213 CX char *strsignal(int);
12214 size_t strspn(const char *, const char *);
12215 char *strstr(const char *, const char *);
12216 char *strtok(char *restrict, const char *restrict);
12217 CX char *strtok_r(char *restrict, const char *restrict, char **restrict);
12218 size_t strxfrm(char *restrict, const char *restrict, size_t);
12219 CX size_t strxfrm_l(char *restrict, const char *restrict,
12220 size_t, locale_t);

```


12221	CX	Inclusion of the <string.h> header may also make visible all symbols from <stddef.h>.
12222		APPLICATION USAGE
12223		None.
12224		RATIONALE
12225		None.
12226		FUTURE DIRECTIONS
12227		None.
12228		SEE ALSO
12229		<locale.h>, <stddef.h>, <sys/types.h>
12230		XSH Section 2.2 (on page 472), <i>memccpy()</i> , <i>memchr()</i> , <i>memcmp()</i> , <i>memcpy()</i> , <i>memmove()</i> ,
12231		<i>memset()</i> , <i>strcat()</i> , <i>strchr()</i> , <i>strcmp()</i> , <i>strcoll()</i> , <i>strcpy()</i> , <i>strcspn()</i> , <i>strdup()</i> , <i>strerror()</i> , <i>strlen()</i> ,
12232		<i>strncat()</i> , <i>strncmp()</i> , <i>strncpy()</i> , <i>strpbrk()</i> , <i>strrchr()</i> , <i>strsignal()</i> , <i>strspn()</i> , <i>strstr()</i> , <i>strtok()</i> , <i>strxfrm()</i>
12233		CHANGE HISTORY
12234		First released in Issue 1. Derived from Issue 1 of the SVID.
12235	Issue 5	
12236		The DESCRIPTION is updated for alignment with the POSIX Threads Extension.
12237	Issue 6	
12238		The <i>strtok_r()</i> function is marked as part of the Thread-Safe Functions option.
12239		This reference page is updated to align with the ISO/IEC 9899:1999 standard.
12240		The <i>strerror_r()</i> function is added in response to IEEE PASC Interpretation 1003.1c #39.
12241	Issue 7	
12242		SD5-XBD-ERN-15 is applied, correcting the prototype for the <i>strerror_r()</i> function.
12243		The <i>stpncpy()</i> , <i>stpncpy()</i> , <i>strndup()</i> , <i>strnlen()</i> , and <i>strsignal()</i> functions are added from The Open
12244		Group Technical Standard, 2006, Extended API Set Part 1.
12245		The <i>strcoll_l()</i> , <i>strerror_l()</i> , and <i>strxfrm_l()</i> functions are added from The Open Group Technical
12246		Standard, 2006, Extended API Set Part 4.
12247		This reference page is clarified with respect to macros and symbolic constants, and a declaration
12248		for the locale_t type is added.

12249 **NAME**

12250 strings.h — string operations

12251 **SYNOPSIS**

12252 #include <strings.h>

12253 **DESCRIPTION**

12254 The following shall be declared as functions and may also be defined as macros. Function
 12255 prototypes shall be provided for use with ISO C standard compilers.

```
12256 XSI  int    ffs(int);
12257      int    strcasecmp(const char *, const char *);
12258      int    strcasecmp_l(const char *, const char *, locale_t);
12259      int    strncasecmp(const char *, const char *, size_t);
12260      int    strncasecmp_l(const char *, const char *, size_t, locale_t);
```

12261 The <strings.h> header shall define the **locale_t** type as described in <locale.h>.12262 The <strings.h> header shall define the **size_t** type as described in <sys/types.h>.12263 **APPLICATION USAGE**

12264 None.

12265 **RATIONALE**

12266 None.

12267 **FUTURE DIRECTIONS**

12268 None.

12269 **SEE ALSO**

12270 <locale.h>, <sys/types.h>

12271 XSH *ffs()*, *strcasecmp()*12272 **CHANGE HISTORY**

12273 First released in Issue 4, Version 2.

12274 **Issue 6**

12275 The Open Group Corrigendum U021/2 is applied, correcting the prototype for *index()* to be
 12276 consistent with the reference page.

12277 The *bcmp()*, *bcopy()*, *bzero()*, *index()*, and *rindex()* functions are marked LEGACY.12278 **Issue 7**12279 SD5-XBD-ERN-56 is applied, adding a reference to <sys/types.h> for the **size_t** type.

12280 The LEGACY functions are removed.

12281 The <strings.h> header is moved from the XSI option to the Base.

12282 The *strcasecmp_l()* and *strncasecmp_l()* functions are added from The Open Group Technical
 12283 Standard, 2006, Extended API Set Part 4.

12284 A declaration for the **locale_t** type is added.

12285 **NAME**12286 stropts.h — STREAMS interface (**STREAMS**)12287 **SYNOPSIS**

12288 OB XSR #include <stropts.h>

12289 **DESCRIPTION**12290 The <stropts.h> header shall define the **bandinfo** structure, which shall include at least the
12291 following members:

12292	int	bi_flag	Flushing type.
12293	unsigned char	bi_pri	Priority band.

12294 The <stropts.h> header shall define the **strpeek** structure, which shall include at least the
12295 following members:

12296	struct strbuf	ctlbuf	The control portion of the message.
12297	struct strbuf	databuf	The data portion of the message.
12298	t_uscalar_t	flags	RS_HIPRI or 0.

12299 The <stropts.h> header shall define the **strbuf** structure, which shall include at least the
12300 following members:

12301	char	*buf	Pointer to buffer.
12302	int	len	Length of data.
12303	int	maxlen	Maximum buffer length.

12304 The <stropts.h> header shall define the **strfdinsert** structure, which shall include at least the
12305 following members:

12306	struct strbuf	ctlbuf	The control portion of the message.
12307	struct strbuf	databuf	The data portion of the message.
12308	int	fildes	File descriptor of the other STREAM.
12309	t_uscalar_t	flags	RS_HIPRI or 0.
12310	int	offset	Relative location of the stored value.

12311 The <stropts.h> header shall define the **striocctl** structure, which shall include at least the
12312 following members:

12313	int	ic_cmd	ioctl() command.
12314	char	*ic_dp	Pointer to buffer.
12315	int	ic_len	Length of data.
12316	int	ic_timeout	Timeout for response.

12317 The <stropts.h> header shall define the **strrecvfd** structure, which shall include at least the
12318 following members:

12319	int	fd	Received file descriptor.
12320	gid_t	gid	GID of sender.
12321	uid_t	uid	UID of sender.

12322 The <stropts.h> header shall define the **uid_t** and **gid_t** types through **typedef**, as described in
12323 <sys/types.h>.12324 The <stropts.h> header shall define the **t_scalar_t** and **t_uscalar_t** types, respectively, as signed
12325 and unsigned opaque types of equal length of at least 32 bits.12326 The <stropts.h> header shall define the **str_list** structure, which shall include at least the
12327 following members:

```

12328     struct str_mlist  *sl_modlist  STREAMS module names.
12329     int                sl_nmods    Number of STREAMS module names.

12330     The <stropts.h> header shall define the str_mlist structure, which shall include at least the
12331     following member:

12332     char  l_name[FMNAMESZ+1]  A STREAMS module name.

12333     The <stropts.h> header shall define at least the following symbolic constants for use as the
12334     request argument to ioctl():

12335     I_ATMARK      Is the top message “marked”?
12336     I_CANPUT      Is a band writable?
12337     I_CKBAND      See if any messages exist in a band.
12338     I_FDINSERT    Send implementation-defined information about another STREAM.
12339     I_FIND        Look for a STREAMS module.
12340     I_FLUSH       Flush a STREAM.
12341     I_FLUSHBAND   Flush one band of a STREAM.
12342     I_GETBAND     Get the band of the top message on a STREAM.
12343     I_GETCLTIME   Get close time delay.
12344     I_GETSIG      Retrieve current notification signals.
12345     I_GRDOPT      Get the read mode.
12346     I_GWROPT      Get the write mode.
12347     I_LINK        Connect two STREAMS.
12348     I_LIST        Get all the module names on a STREAM.
12349     I_LOOK        Get the top module name.
12350     I_NREAD       Size the top message.
12351     I_PEEK        Peek at the top message on a STREAM.
12352     I_PLINK       Persistently connect two STREAMS.
12353     I_POP         Pop a STREAMS module.
12354     I_PUNLINK     Dismantle a persistent STREAMS link.
12355     I_PUSH        Push a STREAMS module.
12356     I_RECVFD      Get a file descriptor sent via I_SENDFD.
12357     I_SENDFD      Pass a file descriptor through a STREAMS pipe.
12358     I_SETCLTIME   Set close time delay.
12359     I_SETSIG      Ask for notification signals.
12360     I_SRDOPT      Set the read mode.
12361     I_STR         Send a STREAMS ioctl().

```

12362	I_SWROPT	Set the write mode.
12363	I_UNLINK	Disconnect two STREAMs.
12364	The <stropts.h> header shall define at least the following symbolic constant for use with	
12365	I_LOOK:	
12366	FMNAMESZ	The minimum size in bytes of the buffer referred to by the <i>arg</i> argument.
12367	The <stropts.h> header shall define at least the following symbolic constants for use with	
12368	I_FLUSH:	
12369	FLUSHR	Flush read queues.
12370	FLUSHRW	Flush read and write queues.
12371	FLUSHW	Flush write queues.
12372	The <stropts.h> header shall define at least the following symbolic constants for use with	
12373	I_SETSIG:	
12374	S_BANDURG	When used in conjunction with S_RDBAND, SIGURG is generated instead of SIGPOLL when a priority message reaches the front of the STREAM head read queue.
12375		
12376		
12377	S_ERROR	Notification of an error condition reaches the STREAM head.
12378	S_HANGUP	Notification of a hangup reaches the STREAM head.
12379	S_HIPRI	A high-priority message is present on a STREAM head read queue.
12380	S_INPUT	A message, other than a high-priority message, has arrived at the head of a STREAM head read queue.
12381		
12382	S_MSG	A STREAMS signal message that contains the SIGPOLL signal reaches the front of the STREAM head read queue.
12383		
12384	S_OUTPUT	The write queue for normal data (priority band 0) just below the STREAM head is no longer full. This notifies the process that there is room on the queue for sending (or writing) normal data downstream.
12385		
12386		
12387	S_RDBAND	A message with a non-zero priority band has arrived at the head of a STREAM head read queue.
12388		
12389	S_RDNORM	A normal (priority band set to 0) message has arrived at the head of a STREAM head read queue.
12390		
12391	S_WRBAND	The write queue for a non-zero priority band just below the STREAM head is no longer full.
12392		
12393	S_WRNORM	Equivalent to S_OUTPUT.
12394	The <stropts.h> header shall define at least the following symbolic constant for use with	
12395	I_PEEK:	
12396	RS_HIPRI	Only look for high-priority messages.
12397	The <stropts.h> header shall define at least the following symbolic constants for use with	
12398	I_SRDOPT:	
12399	RMSGD	Message-discard mode.

12400	RMSGN	Message-non-discard mode.
12401	RNORM	Byte-STREAM mode, the default.
12402	RPROTDAT	Deliver the control part of a message as data when a process issues a <i>read()</i> .
12403	RPROTDIS	Discard the control part of a message, delivering any data part, when a
12404		process issues a <i>read()</i> .
12405	RPROTNORM	Fail <i>read()</i> with [EBADMSG] if a message containing a control part is at the
12406		front of the STREAM head read queue.
12407	The <stropts.h> header shall define at least the following symbolic constant for use with	
12408	L_SWOPT:	
12409	SNDZERO	Send a zero-length message downstream when a <i>write()</i> of 0 bytes occurs.
12410	The <stropts.h> header shall define at least the following symbolic constants for use with	
12411	L_ATMARK:	
12412	ANYMARK	Check if the message is marked.
12413	LASTMARK	Check if the message is the last one marked on the queue.
12414	The <stropts.h> header shall define at least the following symbolic constant for use with	
12415	L_UNLINK:	
12416	MUXID_ALL	Unlink all STREAMs linked to the STREAM associated with <i>files</i> .
12417	The <stropts.h> header shall define the following symbolic constants for <i>getmsg()</i> , <i>getpmsg()</i> ,	
12418	<i>putmsg()</i> , and <i>putpmsg()</i> :	
12419	MORECTL	More control information is left in message.
12420	MOREDATA	More data is left in message.
12421	MSG_ANY	Receive any message.
12422	MSG_BAND	Receive message from specified band.
12423	MSG_HIPRI	Send/receive high-priority message.
12424	The <stropts.h> header may make visible all of the symbols from <unistd.h>.	
12425	The following shall be declared as functions and may also be defined as macros. Function	
12426	prototypes shall be provided.	
12427	<code>int fattach(int, const char *);</code>	
12428	<code>int fdetach(const char *);</code>	
12429	<code>int getmsg(int, struct strbuf *restrict, struct strbuf *restrict,</code>	
12430	<code> int *restrict);</code>	
12431	<code>int getpmsg(int, struct strbuf *restrict, struct strbuf *restrict,</code>	
12432	<code> int *restrict, int *restrict);</code>	
12433	<code>int ioctl(int, int, ...);</code>	
12434	<code>int isastream(int);</code>	
12435	<code>int putmsg(int, const struct strbuf *, const struct strbuf *, int);</code>	
12436	<code>int putpmsg(int, const struct strbuf *, const struct strbuf *, int,</code>	
12437	<code> int);</code>	

12438 **APPLICATION USAGE**

12439 None.

12440 **RATIONALE**

12441 None.

12442 **FUTURE DIRECTIONS**

12443 None.

12444 **SEE ALSO**12445 [<sys/types.h>](#), [<unistd.h>](#)12446 XSH [close\(\)](#), [fattach\(\)](#), [fcntl\(\)](#), [fdetach\(\)](#), [getmsg\(\)](#), [ioctl\(\)](#), [isastream\(\)](#), [open\(\)](#), [pipe\(\)](#), [read\(\)](#), [poll\(\)](#),12447 [putmsg\(\)](#), [signal\(\)](#), [write\(\)](#)12448 **CHANGE HISTORY**

12449 First released in Issue 4, Version 2.

12450 **Issue 5**12451 The *flags* members of the **strpeek** and **strfdinsert** structures are changed from **type long** to
12452 **t_uscalar_t**.12453 **Issue 6**

12454 This header is marked as part of the XSI STREAMS Option Group.

12455 The **restrict** keyword is added to the prototypes for [getmsg\(\)](#) and [getpmsg\(\)](#).12456 **Issue 7**12457 SD5-XBD-ERN-87 is applied, correcting an error in the **strrecvfd** structure.12458 The **<stropts.h>** header is marked obsolescent.

12459 This reference page is clarified with respect to macros and symbolic constants.

12460 **NAME**

12461 sys/ipc.h — XSI interprocess communication access structure

12462 **SYNOPSIS**12463 XSI `#include <sys/ipc.h>`12464 **DESCRIPTION**

12465 The **<sys/ipc.h>** header is used by three mechanisms for XSI interprocess communication (IPC):
 12466 messages, semaphores, and shared memory. All use a common structure type, **ipc_perm**, to pass
 12467 information used in determining permission to perform an IPC operation.

12468 The **<sys/ipc.h>** header shall define the **ipc_perm** structure, which shall include the following
 12469 members:

12470	uid_t	uid	Owner's user ID.
12471	gid_t	gid	Owner's group ID.
12472	uid_t	cuid	Creator's user ID.
12473	gid_t	cgid	Creator's group ID.
12474	mode_t	mode	Read/write permission.

12475 The **<sys/ipc.h>** header shall define the **uid_t**, **gid_t**, **mode_t**, and **key_t** types as described in
 12476 **<sys/types.h>**.

12477 The **<sys/ipc.h>** header shall define the following symbolic constants.

12478 Mode bits:

12479	IPC_CREAT	Create entry if key does not exist.
12480	IPC_EXCL	Fail if key exists.
12481	IPC_NOWAIT	Error if request must wait.

12482 Keys:

12483	IPC_PRIVATE	Private key.

12484 Control commands:

12485	IPC_RMID	Remove identifier.
12486	IPC_SET	Set options.
12487	IPC_STAT	Get options.

12488 The following shall be declared as a function and may also be defined as a macro. A function
 12489 prototype shall be provided.

12490 `key_t ftok(const char *, int);`

12491 **APPLICATION USAGE**

12492 None.

12493 **RATIONALE**

12494 None.

12495 **FUTURE DIRECTIONS**

12496 None.

12497 **SEE ALSO**

12498 <sys/types.h>

12499 XSH *flock()*

12500 **CHANGE HISTORY**

12501 First released in Issue 2. Derived from System V Release 2.0.

12502 **Issue 7**

12503 This reference page is clarified with respect to macros and symbolic constants.

12504 **NAME**

12505 sys/mman.h — memory management declarations

12506 **SYNOPSIS**

12507 #include <sys/mman.h>

12508 **DESCRIPTION**12509 The <sys/mman.h> header shall define the following symbolic constants for use as protection
12510 options:

12511 PROT_EXEC Page can be executed.

12512 PROT_NONE Page cannot be accessed.

12513 PROT_READ Page can be read.

12514 PROT_WRITE Page can be written.

12515 The <sys/mman.h> header shall define the following symbolic constants for use as flag options:

12516 MAP_FIXED Interpret *addr* exactly.

12517 MAP_PRIVATE Changes are private.

12518 MAP_SHARED Share changes.

12519 XSI|SIO The <sys/mman.h> header shall define the following symbolic constants for the *msync()*
12520 function:

12521 MS_ASYNC Perform asynchronous writes.

12522 MS_INVALIDATE Invalidate mappings.

12523 MS_SYNC Perform synchronous writes.

12524 ML The <sys/mman.h> header shall define the following symbolic constants for the *mlockall()*
12525 function:

12526 MCL_CURRENT Lock currently mapped pages.

12527 MCL_FUTURE Lock pages that become mapped.

12528 The <sys/mman.h> header shall define the symbolic constant MAP_FAILED which shall have
12529 type **void *** and shall be used to indicate a failure from the *mmap()* function .12530 ADV If the Advisory Information option is supported, the <sys/mman.h> header shall define
12531 symbolic constants for the *advice* argument to the *posix_madvise()* function as follows:

12532 POSIX_MADV_DONTNEED

12533 The application expects that it will not access the specified range in the near future.

12534 POSIX_MADV_NORMAL

12535 The application has no advice to give on its behavior with respect to the specified range. It
12536 is the default characteristic if no advice is given for a range of memory.

12537 POSIX_MADV_RANDOM

12538 The application expects to access the specified range in a random order.

12539 POSIX_MADV_SEQUENTIAL

12540 The application expects to access the specified range sequentially from lower addresses to
12541 higher addresses.

12542		POSIX_MADV_WILLNEED
12543		The application expects to access the specified range in the near future.
12544	TYM	The <sys/mman.h> header shall define the following symbolic constants for use as flags for the <i>posix_typed_mem_open()</i> function:
12545		
12546		POSIX_TYPED_MEM_ALLOCATE
12547		Allocate on <i>mmap()</i> .
12548		POSIX_TYPED_MEM_ALLOCATE_CONTIG
12549		Allocate contiguously on <i>mmap()</i> .
12550		POSIX_TYPED_MEM_MAP_ALLOCATABLE
12551		Map on <i>mmap()</i> , without affecting allocatability.
12552		The <sys/mman.h> header shall define the mode_t , off_t , and size_t types as described in
12553		<sys/types.h>.
12554	TYM	The <sys/mman.h> header shall define the posix_typed_mem_info structure, which shall
12555		include at least the following member:
12556		size_t posix_tmi_length Maximum length which may be allocated
12557		from a typed memory object.
12558		The following shall be declared as functions and may also be defined as macros. Function
12559		prototypes shall be provided.
12560	MLR	int mlock(const void *, size_t);
12561	ML	int mlockall(int);
12562		void *mmap(void *, size_t, int, int, int, off_t);
12563		int mprotect(void *, size_t, int);
12564	XSI SIO	int msync(void *, size_t, int);
12565	MLR	int munlock(const void *, size_t);
12566	ML	int munlockall(void);
12567		int munmap(void *, size_t);
12568	ADV	int posix_madvise(void *, size_t, int);
12569	TYM	int posix_mem_offset(const void *restrict, size_t, off_t *restrict,
12570		size_t *restrict, int *restrict);
12571		int posix_typed_mem_get_info(int, struct posix_typed_mem_info *);
12572		int posix_typed_mem_open(const char *, int, int);
12573	SHM	int shm_open(const char *, int, mode_t);
12574		int shm_unlink(const char *);

12575 **APPLICATION USAGE**

12576 None.

12577 **RATIONALE**

12578 None.

12579 **FUTURE DIRECTIONS**

12580 None.

12581 **SEE ALSO**12582 [<sys/types.h>](#)

12583 XSH [mlock\(\)](#), [mlockall\(\)](#), [mmap\(\)](#), [mprotect\(\)](#), [msync\(\)](#), [munmap\(\)](#), [posix_madvise\(\)](#),
 12584 [posix_mem_offset\(\)](#), [posix_typed_mem_get_info\(\)](#), [posix_typed_mem_open\(\)](#), [shm_open\(\)](#),
 12585 [shm_unlink\(\)](#)

12586 **CHANGE HISTORY**

12587 First released in Issue 4, Version 2.

12588 **Issue 5**

12589 Updated for alignment with the POSIX Realtime Extension.

12590 **Issue 6**

12591 The [<sys/mman.h>](#) header is marked as dependent on support for either the Memory Mapped
 12592 Files, Process Memory Locking, or Shared Memory Objects options.

12593 The following changes are made for alignment with IEEE Std 1003.1j-2000:

- 12594 • The TYM margin code is added to the list of margin codes for the [<sys/mman.h>](#) header
 12595 line, as well as for other lines.
- 12596 • The POSIX_TYPED_MEM_ALLOCATE, POSIX_TYPED_MEM_ALLOCATE_CONTIG,
 12597 and POSIX_TYPED_MEM_MAP_ALLOCATABLE flags are added.
- 12598 • The [posix_tmi_length](#) structure is added.
- 12599 • The [posix_mem_offset\(\)](#), [posix_typed_mem_get_info\(\)](#), and [posix_typed_mem_open\(\)](#) functions
 12600 are added.

12601 The **restrict** keyword is added to the prototype for [posix_mem_offset\(\)](#).12602 IEEE PASC Interpretation 1003.1 #102 is applied, adding the prototype for [posix_madvise\(\)](#).

12603 IEEE Std 1003.1-2001/Cor 1-2002, item XBD/TC1/D6/16 is applied, correcting margin code and
 12604 shading errors for the [mlock\(\)](#) and [munlock\(\)](#) functions.

12605 IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/34 is applied, changing the margin code
 12606 for the [mmap\(\)](#) function from MF|SHM to MC3 (notation for MF|SHM|TYM).

12607 IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/36 is applied, changing the margin code
 12608 for the [munmap\(\)](#) function from MF|SHM to MC3 (notation for MF|SHM|TYM).

12609 **Issue 7**

12610 SD5-XBD-ERN-5 is applied, rewriting the DESCRIPTION.

12611 Functionality relating to the Memory Protection and Memory Mapped Files options is moved to
 12612 the Base.

12613 This reference page is clarified with respect to macros and symbolic constants.

12614 **NAME**

12615 sys/msg.h — XSI message queue structures

12616 **SYNOPSIS**

12617 XSI `#include <sys/msg.h>`

12618 **DESCRIPTION**

12619 The <sys/msg.h> header shall define the following data types through **typedef**:

12620 **msgqnum_t** Used for the number of messages in the message queue.

12621 **msglen_t** Used for the number of bytes allowed in a message queue.

12622 These types shall be unsigned integer types that are able to store values at least as large as a type
12623 **unsigned short**.

12624 The <sys/msg.h> header shall define the following symbolic constant as a message operation
12625 flag:

12626 MSG_NOERROR No error if big message.

12627 The <sys/msg.h> header shall define the **msqid_ds** structure, which shall include the following
12628 members:

12629	struct ipc_perm	msg_perm	Operation permission structure.
12630	msgqnum_t	msg_qnum	Number of messages currently on queue.
12631	msglen_t	msg_qbytes	Maximum number of bytes allowed on queue.
12632	pid_t	msg_lspid	Process ID of last <i>msgsnd()</i> .
12633	pid_t	msg_lrpid	Process ID of last <i>msgrcv()</i> .
12634	time_t	msg_stime	Time of last <i>msgsnd()</i> .
12635	time_t	msg_rtime	Time of last <i>msgrcv()</i> .
12636	time_t	msg_ctime	Time of last change.

12637 The <sys/msg.h> header shall define the **pid_t**, **size_t**, **ssize_t**, and **time_t** types as described in
12638 <sys/types.h>.

12639 The following shall be declared as functions and may also be defined as macros. Function
12640 prototypes shall be provided.

```
12641 int      msgctl(int, int, struct msqid_ds *);
12642 int      msgget(key_t, int);
12643 ssize_t  msgrcv(int, void *, size_t, long, int);
12644 int      msgsnd(int, const void *, size_t, int);
```

12645 In addition, the <sys/msg.h> header shall include the <sys/ipc.h> header.

12646 **APPLICATION USAGE**

12647 None.

12648 **RATIONALE**

12649 None.

12650 **FUTURE DIRECTIONS**

12651 None.

12652 **SEE ALSO**

12653 <sys/ipc.h>, <sys/types.h>

12654 XSH *msgctl()*, *msgget()*, *msgrcv()*, *msgsnd()*

12655 **CHANGE HISTORY**

12656 First released in Issue 2. Derived from System V Release 2.0.

12657 **Issue 7**

12658 Austin Group Interpretation 1003.1-2001 #179 is applied.

12659 This reference page is clarified with respect to macros and symbolic constants.

12660 **NAME**

12661 sys/resource.h — definitions for XSI resource operations

12662 **SYNOPSIS**

12663 XSI `#include <sys/resource.h>`

12664 **DESCRIPTION**

12665 The <sys/resource.h> header shall define the following symbolic constants as possible values of
12666 the *which* argument of *getpriority()* and *setpriority()*:

12667 PRIO_PROCESS Identifies the *who* argument as a process ID.

12668 PRIO_PGRP Identifies the *who* argument as a process group ID.

12669 PRIO_USER Identifies the *who* argument as a user ID.

12670 The <sys/resource.h> header shall define the following type through **typedef**:

12671 **rlim_t** Unsigned integer type used for limit values.

12672 The <sys/resource.h> header shall define the following symbolic constants, which shall have
12673 values suitable for use in **#if** preprocessing directives:

12674 RLIM_INFINITY A value of **rlim_t** indicating no limit.

12675 RLIM_SAVED_MAX A value of type **rlim_t** indicating an unrepresentable saved hard
12676 limit.

12677 RLIM_SAVED_CUR A value of type **rlim_t** indicating an unrepresentable saved soft limit.

12678 On implementations where all resource limits are representable in an object of type **rlim_t**,
12679 RLIM_SAVED_MAX and RLIM_SAVED_CUR need not be distinct from RLIM_INFINITY.

12680 The <sys/resource.h> header shall define the following symbolic constants as possible values of
12681 the *who* parameter of *getrusage()*:

12682 RUSAGE_SELF Returns information about the current process.

12683 RUSAGE_CHILDREN Returns information about children of the current process.

12684 The <sys/resource.h> header shall define the **rlimit** structure, which shall include at least the
12685 following members:

12686 `rlim_t rlim_cur` The current (soft) limit.

12687 `rlim_t rlim_max` The hard limit.

12688 The <sys/resource.h> header shall define the **rusage** structure, which shall include at least the
12689 following members:

12690 `struct timeval ru_utime` User time used.

12691 `struct timeval ru_stime` System time used.

12692 The <sys/resource.h> header shall define the **timeval** structure as described in <sys/time.h>.

12693 The <sys/resource.h> header shall define the following symbolic constants as possible values for
12694 the *resource* argument of *getrlimit()* and *setrlimit()*:

12695 RLIMIT_CORE Limit on size of **core** file.

12696 RLIMIT_CPU Limit on CPU time per process.

12697	RLIMIT_DATA	Limit on data segment size.
12698	RLIMIT_FSIZE	Limit on file size.
12699	RLIMIT_NOFILE	Limit on number of open files.
12700	RLIMIT_STACK	Limit on stack size.
12701	RLIMIT_AS	Limit on address space size.
12702	The following shall be declared as functions and may also be defined as macros. Function	
12703	prototypes shall be provided.	
12704	<code>int getpriority(int, id_t);</code>	
12705	<code>int getrlimit(int, struct rlimit *);</code>	
12706	<code>int getrusage(int, struct rusage *);</code>	
12707	<code>int setpriority(int, id_t, int);</code>	
12708	<code>int setrlimit(int, const struct rlimit *);</code>	
12709	The <sys/resource.h> header shall define the id_t type through typedef , as described in	
12710	<sys/types.h>.	
12711	Inclusion of the <sys/resource.h> header may also make visible all symbols from <sys/time.h>.	
12712	APPLICATION USAGE	
12713	None.	
12714	RATIONALE	
12715	None.	
12716	FUTURE DIRECTIONS	
12717	None.	
12718	SEE ALSO	
12719	<sys/time.h> , <sys/types.h>	
12720	XSH getpriority() , getrlimit() , getrusage()	
12721	CHANGE HISTORY	
12722	First released in Issue 4, Version 2.	
12723	Issue 5	
12724	Large File System extensions are added.	
12725	Issue 7	
12726	This reference page is clarified with respect to macros and symbolic constants.	

12727 **NAME**

12728 sys/select.h — select types

12729 **SYNOPSIS**

12730 #include <sys/select.h>

12731 **DESCRIPTION**

12732 The <sys/select.h> header shall define the **timeval** structure, which shall include at least the
12733 following members:

12734	time_t	tv_sec	Seconds.
12735	suseconds_t	tv_usec	Microseconds.

12736 The <sys/select.h> header shall define the **time_t** and **suseconds_t** types as described in
12737 <sys/types.h>.

12738 The <sys/select.h> header shall define the **sigset_t** type as described in <signal.h>.

12739 The <sys/select.h> header shall define the **timespec** structure as described in <time.h>.

12740 The <sys/select.h> header shall define the **fd_set** type as a structure.

12741 The <sys/select.h> header shall define the following symbolic constant, which shall have a value
12742 suitable for use in #if preprocessing directives:

12743 FD_SETSIZE Maximum number of file descriptors in an **fd_set** structure.

12744 The following shall be declared as functions, defined as macros, or both. If functions are
12745 declared, function prototypes shall be provided.

```
12746 void FD_CLR(int, fd_set *);
12747 int FD_ISSET(int, fd_set *);
12748 void FD_SET(int, fd_set *);
12749 void FD_ZERO(fd_set *);
```

12750 If implemented as macros, these may evaluate their arguments more than once, so applications
12751 should ensure that the arguments they supply are never expressions with side-effects.

12752 The following shall be declared as functions and may also be defined as macros. Function
12753 prototypes shall be provided.

```
12754 int pselect(int, fd_set *restrict, fd_set *restrict, fd_set *restrict,
12755             const struct timespec *restrict, const sigset_t *restrict);
12756 int select(int, fd_set *restrict, fd_set *restrict, fd_set *restrict,
12757            struct timeval *restrict);
```

12758 Inclusion of the <sys/select.h> header may make visible all symbols from the headers
12759 <signal.h> and <time.h>.

12760 **APPLICATION USAGE**

12761 None.

12762 **RATIONALE**

12763 None.

12764 **FUTURE DIRECTIONS**

12765 None.

12766 **SEE ALSO**12767 [<signal.h>](#), [<sys/time.h>](#), [<sys/types.h>](#), [<time.h>](#)12768 XSH [pselect\(\)](#)12769 **CHANGE HISTORY**

12770 First released in Issue 6. Derived from IEEE Std 1003.1g-2000.

12771 The requirement for the **fd_set** structure to have a member *fds_bits* has been removed as per The
12772 Open Group Base Resolution bwg2001-005.12773 **Issue 7**

12774 SD5-XBD-ERN-6 is applied, reordering the DESCRIPTION.

12775 This reference page is clarified with respect to macros and symbolic constants.

12776 **NAME**

12777 sys/sem.h — XSI semaphore facility

12778 **SYNOPSIS**

12779 XSI `#include <sys/sem.h>`

12780 **DESCRIPTION**

12781 The **<sys/sem.h>** header shall define the following symbolic constant for use as a semaphore
12782 operation flag:

12783 SEM_UNDO Set up adjust on exit entry.

12784 The **<sys/sem.h>** header shall define the following symbolic constants for use as commands for
12785 the *semctl()* function:

12786 GETNCNT Get *semmcnt*.

12787 GETPID Get *sempid*.

12788 GETVAL Get *semval*.

12789 GETALL Get all cases of *semval*.

12790 GETZCNT Get *semzcnt*.

12791 SETVAL Set *semval*.

12792 SETALL Set all cases of *semval*.

12793 The **<sys/sem.h>** header shall define the **semid_ds** structure, which shall include the following
12794 members:

12795 struct ipc_perm sem_perm Operation permission structure.

12796 unsigned short sem_nsems Number of semaphores in set.

12797 time_t sem_otime Last *semop()* time.

12798 time_t sem_ctime Last time changed by *semctl()*.

12799 The **<sys/sem.h>** header shall define the **pid_t**, **size_t**, and **time_t** types as described in
12800 **<sys/types.h>**.

12801 A semaphore shall be represented by an anonymous structure, which shall include the following
12802 members:

12803 unsigned short semval Semaphore value.

12804 pid_t sempid Process ID of last operation.

12805 unsigned short semncnt Number of processes waiting for *semval*
12806 to become greater than current value.

12807 unsigned short semzcnt Number of processes waiting for *semval*
12808 to become 0.

12809 The **<sys/sem.h>** header shall define the **sembuf** structure, which shall include the following
12810 members:

12811 unsigned short sem_num Semaphore number.

12812 short sem_op Semaphore operation.

12813 short sem_flg Operation flags.

12814 The following shall be declared as functions and may also be defined as macros. Function
12815 prototypes shall be provided.

12816 int semctl(int, int, int, ...);

```
12817         int    semget(key_t, int, int);
12818         int    semop(int, struct sembuf *, size_t);
```

12819 In addition, the **<sys/sem.h>** header shall include the **<sys/ipc.h>** header.

12820 **APPLICATION USAGE**

12821 None.

12822 **RATIONALE**

12823 None.

12824 **FUTURE DIRECTIONS**

12825 None.

12826 **SEE ALSO**

12827 **<sys/ipc.h>**, **<sys/types.h>**

12828 XSH *semctl()*, *semget()*, *semop()*

12829 **CHANGE HISTORY**

12830 First released in Issue 2. Derived from System V Release 2.0.

12831 **Issue 7**

12832 Austin Group Interpretation 1003.1-2001 #179 is applied.

12833 This reference page is clarified with respect to macros and symbolic constants.

12834 **NAME**

12835 sys/shm.h — XSI shared memory facility

12836 **SYNOPSIS**

12837 XSI `#include <sys/shm.h>`

12838 **DESCRIPTION**

12839 The <sys/shm.h> header shall define the following symbolic constants:

12840 SHM_RDONLY Attach read-only (else read-write).

12841 SHM_RND Round attach address to SHMLBA.

12842 SHMLBA Segment low boundary address multiple.

12843 The <sys/shm.h> header shall define the following data types through **typedef**:

12844 **shmatt_t** Unsigned integer used for the number of current attaches that must be able to
12845 store values at least as large as a type **unsigned short**.

12846 The <sys/shm.h> header shall define the **shmid_ds** structure, which shall include the following
12847 members:

12848	struct ipc_perm	shm_perm	Operation permission structure.
12849	size_t	shm_segsz	Size of segment in bytes.
12850	pid_t	shm_lpid	Process ID of last shared memory operation.
12851	pid_t	shm_cpid	Process ID of creator.
12852	shmatt_t	shm_nattch	Number of current attaches.
12853	time_t	shm_atime	Time of last <i>shmat</i> ().
12854	time_t	shm_dtime	Time of last <i>shmdt</i> ().
12855	time_t	shm_ctime	Time of last change by <i>shmctl</i> ().

12856 The <sys/shm.h> header shall define the **pid_t**, **size_t**, and **time_t** types as described in
12857 <sys/types.h>.

12858 The following shall be declared as functions and may also be defined as macros. Function
12859 prototypes shall be provided.

```
12860 void *shmat(int, const void *, int);
12861 int shmctl(int, int, struct shmid_ds *);
12862 int shmdt(const void *);
12863 int shmget(key_t, size_t, int);
```

12864 In addition, the <sys/shm.h> header shall include the <sys/ipc.h> header.

12865 **APPLICATION USAGE**

12866 None.

12867 **RATIONALE**

12868 None.

12869 **FUTURE DIRECTIONS**

12870 None.

12871 **SEE ALSO**

12872 [<sys/ipc.h>](#), [<sys/types.h>](#)

12873 XSH *shmat*(), *shmctl*(), *shmdt*(), *shmget*()

CHANGE HISTORY

12874
12875 First released in Issue 2. Derived from System V Release 2.0.

Issue 5

12876
12877 The type of *shm_segsz* is changed from **int** to **size_t**.

Issue 7

12878
12879 Austin Group Interpretation 1003.1-2001 #179 is applied.

12880 This reference page is clarified with respect to macros and symbolic constants.

12881 **NAME**

12882 sys/socket.h — main sockets header

12883 **SYNOPSIS**

12884 #include <sys/socket.h>

12885 **DESCRIPTION**

12886 The <sys/socket.h> header shall define the **socklen_t** type, which is an integer type of width of
12887 at least 32 bits; see APPLICATION USAGE.

12888 The <sys/socket.h> header shall define the **sa_family_t** unsigned integer type.

12889 The <sys/socket.h> header shall define the **sockaddr** structure, which shall include at least the
12890 following members:

12891 sa_family_t sa_family Address family.
12892 char sa_data[] Socket address (variable-length data).

12893 The **sockaddr** structure is used to define a socket address which is used in the *bind()*, *connect()*,
12894 *getpeername()*, *getsockname()*, *recvfrom()*, and *sendto()* functions.

12895 The <sys/socket.h> header shall define the **sockaddr_storage** structure, which shall be:

- 12896 • Large enough to accommodate all supported protocol-specific address structures
- 12897 • Aligned at an appropriate boundary so that pointers to it can be cast as pointers to
12898 protocol-specific address structures and used to access the fields of those structures
12899 without alignment problems

12900 The **sockaddr_storage** structure shall include at least the following members:

12901 sa_family_t ss_family

12902 When a pointer to a **sockaddr_storage** structure is cast as a pointer to a **sockaddr** structure, the
12903 *ss_family* field of the **sockaddr_storage** structure shall map onto the *sa_family* field of the
12904 **sockaddr** structure. When a pointer to a **sockaddr_storage** structure is cast as a pointer to a
12905 protocol-specific address structure, the *ss_family* field shall map onto a field of that structure that
12906 is of type **sa_family_t** and that identifies the protocol's address family.

12907 The <sys/socket.h> header shall define the **msghdr** structure, which shall include at least the
12908 following members:

12909 void *msg_name Optional address.
12910 socklen_t msg_namelen Size of address.
12911 struct iovec *msg_iov Scatter/gather array.
12912 int msg_iovlen Members in *msg_iov*.
12913 void *msg_control Ancillary data; see below.
12914 socklen_t msg_controllen Ancillary data buffer *len*.
12915 int msg_flags Flags on received message.

12916 The **msghdr** structure is used to minimize the number of directly supplied parameters to the
12917 *recvmsg()* and *sendmsg()* functions. This structure is used as a *value-result* parameter in the
12918 *recvmsg()* function and *value* only for the *sendmsg()* function.

12919 The <sys/socket.h> header shall define the **iovec** structure as described in <sys/uio.h>.

12920 The <sys/socket.h> header shall define the **cmsg_hdr** structure, which shall include at least the
12921 following members:

12922 socklen_t cmsg_len Data byte count, including the **cmsg_hdr**.
12923 int cmsg_level Originating protocol.

12924 int cmsg_type Protocol-specific type.

12925 The **cmsg_hdr** structure is used for storage of ancillary data object information.

12926 Ancillary data consists of a sequence of pairs, each consisting of a **cmsg_hdr** structure followed
12927 by a data array. The data array contains the ancillary data message, and the **cmsg_hdr** structure
12928 contains descriptive information that allows an application to correctly parse the data.

12929 The values for *cmsg_level* shall be legal values for the *level* argument to the *getsockopt()* and
12930 *setsockopt()* functions. The system documentation shall specify the *cmsg_type* definitions for the
12931 supported protocols.

12932 Ancillary data is also possible at the socket level. The **<sys/socket.h>** header shall define the
12933 following symbolic constant for use as the *cmsg_type* value when *cmsg_level* is SOL_SOCKET:

12934 SCM_RIGHTS Indicates that the data array contains the access rights to be sent or
12935 received.

12936 The **<sys/socket.h>** header shall define the following macros to gain access to the data arrays in
12937 the ancillary data associated with a message header:

12938 CMMSG_DATA(*cmsg*)
12939 If the argument is a pointer to a **cmsg_hdr** structure, this macro shall return an unsigned
12940 character pointer to the data array associated with the **cmsg_hdr** structure.

12941 CMMSG_NXTHDR(*mhdr, cmsg*)
12942 If the first argument is a pointer to a **msghdr** structure and the second argument is a pointer
12943 to a **cmsg_hdr** structure in the ancillary data pointed to by the *msg_control* field of that
12944 **msghdr** structure, this macro shall return a pointer to the next **cmsg_hdr** structure, or a null
12945 pointer if this structure is the last **cmsg_hdr** in the ancillary data.

12946 CMMSG_FIRSTHDR(*mhdr*)
12947 If the argument is a pointer to a **msghdr** structure, this macro shall return a pointer to the
12948 first **cmsg_hdr** structure in the ancillary data associated with this **msghdr** structure, or a null
12949 pointer if there is no ancillary data associated with the **msghdr** structure.

12950 The **<sys/socket.h>** header shall define the **linger** structure, which shall include at least the
12951 following members:

12952 int l_onoff Indicates whether linger option is enabled.
12953 int l_linger Linger time, in seconds.

12954 The **<sys/socket.h>** header shall define the following symbolic constants with distinct values:

12955 SOCK_DGRAM Datagram socket.

12956 RS SOCK_RAW Raw Protocol Interface.

12957 SOCK_SEQPACKET Sequenced-packet socket.

12958 SOCK_STREAM Byte-stream socket.

12959 The **<sys/socket.h>** header shall define the following symbolic constant for use as the *level*
12960 argument of *setsockopt()* and *getsockopt()*.

12961 SOL_SOCKET Options to be accessed at socket level, not protocol level.

12962 The **<sys/socket.h>** header shall define the following symbolic constants with distinct values for
12963 use as the *option_name* argument in *getsockopt()* or *setsockopt()* calls (see XSH [Section 2.10.16](#), on
12964 page 527):

12965	SO_ACCEPTCONN	Socket is accepting connections.
12966	SO_BROADCAST	Transmission of broadcast messages is supported.
12967	SO_DEBUG	Debugging information is being recorded.
12968	SO_DONTROUTE	Bypass normal routing.
12969	SO_ERROR	Socket error status.
12970	SO_KEEPALIVE	Connections are kept alive with periodic messages.
12971	SO_LINGER	Socket lingers on close.
12972	SO_OOBINLINE	Out-of-band data is transmitted in line.
12973	SO_RCVBUF	Receive buffer size.
12974	SO_RCVLOWAT	Receive “low water mark”.
12975	SO_RCVTIMEO	Receive timeout.
12976	SO_REUSEADDR	Reuse of local addresses is supported.
12977	SO_SNDBUF	Send buffer size.
12978	SO_SNDLOWAT	Send “low water mark”.
12979	SO_SNDTIMEO	Send timeout.
12980	SO_TYPE	Socket type.
12981	The <sys/socket.h> header shall define the following symbolic constant for use as the maximum	
12982	<i>backlog</i> queue length which may be specified by the <i>backlog</i> field of the <i>listen()</i> function:	
12983	SOMAXCONN	The maximum <i>backlog</i> queue length.
12984	The <sys/socket.h> header shall define the following symbolic constants with distinct values for	
12985	use as the valid values for the <i>msg_flags</i> field in the msghdr structure, or the <i>flags</i> parameter in	
12986	<i>recv()</i> , <i>recvfrom()</i> , <i>recvmsg()</i> , <i>send()</i> , <i>sendmsg()</i> , or <i>sendto()</i> calls:	
12987	MSG_CTRUNC	Control data truncated.
12988	MSG_DONTROUTE	Send without using routing tables.
12989	MSG_EOR	Terminates a record (if supported by the protocol).
12990	MSG_OOB	Out-of-band data.
12991	MSG_NOSIGNAL	No SIGPIPE generated when an attempt to send is made on a stream-
12992		oriented socket that is no longer connected.
12993	MSG_PEEK	Leave received data in queue.
12994	MSG_TRUNC	Normal data truncated.
12995	MSG_WAITALL	Attempt to fill the read buffer.
12996	The <sys/socket.h> header shall define the following symbolic constants with distinct values:	
12997	AF_INET	Internet domain sockets for use with IPv4 addresses.
12998	IP6 AF_INET6	Internet domain sockets for use with IPv6 addresses.
12999	AF_UNIX	UNIX domain sockets.

13000 AF_UNSPEC Unspecified.

13001 The <sys/socket.h> header shall define the following symbolic constants with distinct values:

13002 SHUT_RD Disables further receive operations.

13003 SHUT_RDWR Disables further send and receive operations.

13004 SHUT_WR Disables further send operations.

13005 The <sys/socket.h> header shall define the **size_t** and **ssize_t** types as described in

13006 <sys/types.h>.

13007 The following shall be declared as functions and may also be defined as macros. Function

13008 prototypes shall be provided.

```

13009 int      accept(int, struct sockaddr *restrict, socklen_t *restrict);
13010 int      bind(int, const struct sockaddr *, socklen_t);
13011 int      connect(int, const struct sockaddr *, socklen_t);
13012 int      getpeername(int, struct sockaddr *restrict, socklen_t *restrict);
13013 int      getsockname(int, struct sockaddr *restrict, socklen_t *restrict);
13014 int      getsockopt(int, int, int, void *restrict, socklen_t *restrict);
13015 int      listen(int, int);
13016 ssize_t  recv(int, void *, size_t, int);
13017 ssize_t  recvfrom(int, void *restrict, size_t, int,
13018                 struct sockaddr *restrict, socklen_t *restrict);
13019 ssize_t  recvmsg(int, struct msghdr *, int);
13020 ssize_t  send(int, const void *, size_t, int);
13021 ssize_t  sendmsg(int, const struct msghdr *, int);
13022 ssize_t  sendto(int, const void *, size_t, int, const struct sockaddr *,
13023                socklen_t);
13024 int      setsockopt(int, int, int, const void *, socklen_t);
13025 int      shutdown(int, int);
13026 int      socketatmark(int);
13027 int      socket(int, int, int);
13028 int      socketpair(int, int, int, int [2]);

```

13029 Inclusion of <sys/socket.h> may also make visible all symbols from <sys/uio.h>.

13030 APPLICATION USAGE

13031 To forestall portability problems, it is recommended that applications not use values larger than

13032 $2^{31} - 1$ for the **socklen_t** type.

13033 The **sockaddr_storage** structure solves the problem of declaring storage for automatic variables

13034 which is both large enough and aligned enough for storing the socket address data structure of

13035 any family. For example, code with a file descriptor and without the context of the address

13036 family can pass a pointer to a variable of this type, where a pointer to a socket address structure

13037 is expected in calls such as *getpeername()*, and determine the address family by accessing the

13038 received content after the call.

13039 The example below illustrates a data structure which aligns on a 64-bit boundary. An

13040 implementation-defined field *_ss_align* following *_ss_pad1* is used to force a 64-bit alignment

13041 which covers proper alignment good enough for needs of at least **sockaddr_in6** (IPv6) and

13042 **sockaddr_in** (IPv4) address data structures. The size of padding field *_ss_pad1* depends on the

13043 chosen alignment boundary. The size of padding field *_ss_pad2* depends on the value of overall

13044 size chosen for the total size of the structure. This size and alignment are represented in the

13045 above example by implementation-defined (not required) constants *_SS_MAXSIZE* (chosen

value 128) and `_SS_ALIGNMENT` (with chosen value 8). Constants `_SS_PAD1SIZE` (derived value 6) and `_SS_PAD2SIZE` (derived value 112) are also for illustration and not required. The implementation-defined definitions and structure field names above start with an <underscore> to denote implementation private name space. Portable code is not expected to access or reference those fields or constants.

```

/*
 * Desired design of maximum size and alignment.
 */
#define _SS_MAXSIZE 128
/* Implementation-defined maximum size. */
#define _SS_ALIGNSIZE (sizeof(int64_t))
/* Implementation-defined desired alignment. */

/*
 * Definitions used for sockaddr_storage structure paddings design.
 */
#define _SS_PAD1SIZE (_SS_ALIGNSIZE - sizeof(sa_family_t))
#define _SS_PAD2SIZE (_SS_MAXSIZE - (sizeof(sa_family_t)+ \
                                   _SS_PAD1SIZE + _SS_ALIGNSIZE))
struct sockaddr_storage {
    sa_family_t  ss_family; /* Address family. */
/*
 * Following fields are implementation-defined.
 */
    char _ss_pad1[_SS_PAD1SIZE];
    /* 6-byte pad; this is to make implementation-defined
       pad up to alignment field that follows explicit in
       the data structure. */
    int64_t _ss_align; /* Field to force desired structure
                       storage alignment. */
    char _ss_pad2[_SS_PAD2SIZE];
    /* 112-byte pad to achieve desired size,
       _SS_MAXSIZE value minus size of ss_family
       __ss_pad1, __ss_align fields is 112. */
};

```

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

<sys/types.h>, <sys/uio.h>

XSH *accept()*, *bind()*, *connect()*, *getpeername()*, *getsockname()*, *getsockopt()*, *listen()*, *recv()*, *recvfrom()*, *recvmsg()*, *send()*, *sendmsg()*, *sendto()*, *setsockopt()*, *shutdown()*, *socketatmark()*, *socket()*, *socketpair()*

CHANGE HISTORY

First released in Issue 6. Derived from the XNS, Issue 5.2 specification.

The **restrict** keyword is added to the prototypes for *accept()*, *getpeername()*, *getsockname()*, *getsockopt()*, and *recvfrom()*.

Issue 7

SD5-XBD-ERN-56 is applied, adding a reference to **<sys/types.h>** for the **ssize_t** type.

SD5-XBD-ERN-62 is applied.

The MSG_NOSIGNAL symbolic constant is added from The Open Group Technical Standard, 2006, Extended API Set Part 2.

This reference page is clarified with respect to macros and symbolic constants, and a declaration for the **size_t** type is added.

POSIX.1-2008, Technical Corrigendum 1, XBD/TC1-2008/0067 [355] is applied.

+

13101 **NAME**

13102 sys/stat.h — data returned by the `stat()` function

13103 **SYNOPSIS**

13104 `#include <sys/stat.h>`

13105 **DESCRIPTION**

13106 The <sys/stat.h> header shall define the structure of the data returned by the `fstat()`, `lstat()`, and
13107 `stat()` functions.

13108 The <sys/stat.h> header shall define the **stat** structure, which shall include at least the following
13109 members:

13110	<code>dev_t st_dev</code>	Device ID of device containing file.
13111	<code>ino_t st_ino</code>	File serial number.
13112	<code>mode_t st_mode</code>	Mode of file (see below).
13113	<code>nlink_t st_nlink</code>	Number of hard links to the file.
13114	<code>uid_t st_uid</code>	User ID of file.
13115	<code>gid_t st_gid</code>	Group ID of file.
13116	XSI <code>dev_t st_rdev</code>	Device ID (if file is character or block special).
13117	<code>off_t st_size</code>	For regular files, the file size in bytes.
13118		For symbolic links, the length in bytes of the
13119		pathname contained in the symbolic link.
13120	SHM	For a shared memory object, the length in bytes.
13121	TYM	For a typed memory object, the length in bytes.
13122		For other file types, the use of this field is
13123		unspecified.
13124	<code>struct timespec st_atim</code>	Last data access timestamp.
13125	<code>struct timespec st_mtim</code>	Last data modification timestamp.
13126	<code>struct timespec st_ctim</code>	Last file status change timestamp.
13127	XSI <code>blksize_t st_blksize</code>	A file system-specific preferred I/O block size
13128		for this object. In some file system types, this
13129		may vary from file to file.
13130	<code>blkcnt_t st_blocks</code>	Number of blocks allocated for this object.

13131 The `st_ino` and `st_dev` fields taken together uniquely identify the file within the system.

13132 XSI The <sys/stat.h> header shall define the **blkcnt_t**, **blksize_t**, **dev_t**, **ino_t**, **mode_t**, **nlink_t**,
13133 **uid_t**, **gid_t**, **off_t**, and **time_t** types as described in <sys/types.h>.

13134 The <sys/stat.h> header shall define the **timespec** structure as described in <time.h>. Times
13135 shall be given in seconds since the Epoch.

13136 Which structure members have meaningful values depends on the type of file. For further
13137 information, see the descriptions of `fstat()`, `lstat()`, and `stat()` in the System Interfaces volume of
13138 POSIX.1-2008.

13139 For compatibility with earlier versions of this standard, the `st_atime` macro shall be defined with
13140 the value `st_atim.tv_sec`. Similarly, `st_ctime` and `st_mtime` shall be defined as macros with the
13141 values `st_ctim.tv_sec` and `st_mtim.tv_sec`, respectively.

The <sys/stat.h> header shall define the following symbolic constants for the file types encoded in type **mode_t**. The values shall be suitable for use in **#if** preprocessing directives:

S_IFMT	Type of file.
S_IFBLK	Block special.
S_IFCHR	Character special.
S_IFIFO	FIFO special.
S_IFREG	Regular.
S_IFDIR	Directory.
S_IFLNK	Symbolic link.
S_IFSOCK	Socket.

The <sys/stat.h> header shall define the following symbolic constants for the file mode bits encoded in type **mode_t**, with the indicated numeric values. These macros shall expand to an expression which has a type that allows them to be used, either singly or OR'ed together, as the third argument to *open()* without the need for a **mode_t** cast. The values shall be suitable for use in **#if** preprocessing directives.

Name	Numeric Value	Description
S_IRWXU	0700	Read, write, execute/search by owner.
S_IRUSR	0400	Read permission, owner.
S_IWUSR	0200	Write permission, owner.
S_IXUSR	0100	Execute/search permission, owner.
S_IRWXG	070	Read, write, execute/search by group.
S_IRGRP	040	Read permission, group.
S_IWGRP	020	Write permission, group.
S_IXGRP	010	Execute/search permission, group.
S_IRWXO	07	Read, write, execute/search by others.
S_IROTH	04	Read permission, others.
S_IWOTH	02	Write permission, others.
S_IXOTH	01	Execute/search permission, others.
S_ISUID	04000	Set-user-ID on execution.
S_ISGID	02000	Set-group-ID on execution.
S_ISVTX	01000	On directories, restricted deletion flag.

The following macros shall be provided to test whether a file is of the specified type. The value *m* supplied to the macros is the value of *st_mode* from a **stat** structure. The macro shall evaluate to a non-zero value if the test is true; 0 if the test is false.

S_ISBLK(<i>m</i>)	Test for a block special file.
S_ISCHR(<i>m</i>)	Test for a character special file.
S_ISDIR(<i>m</i>)	Test for a directory.
S_ISFIFO(<i>m</i>)	Test for a pipe or FIFO special file.
S_ISREG(<i>m</i>)	Test for a regular file.

13181		<code>S_ISLNK(<i>m</i>)</code>	Test for a symbolic link.
13182		<code>S_ISSOCK(<i>m</i>)</code>	Test for a socket.
13183		The implementation may implement message queues, semaphores, or shared memory objects as distinct file types. The following macros shall be provided to test whether a file is of the specified type. The value of the <i>buf</i> argument supplied to the macros is a pointer to a stat structure. The macro shall evaluate to a non-zero value if the specified object is implemented as a distinct file type and the specified file type is contained in the stat structure referenced by <i>buf</i> . Otherwise, the macro shall evaluate to zero.	
13184			
13185			
13186			
13187			
13188			
13189		<code>S_TYPEISMQ(<i>buf</i>)</code>	Test for a message queue.
13190		<code>S_TYPEISSEM(<i>buf</i>)</code>	Test for a semaphore.
13191		<code>S_TYPEISSHM(<i>buf</i>)</code>	Test for a shared memory object.
13192	TYM	The implementation may implement typed memory objects as distinct file types, and the following macro shall test whether a file is of the specified type. The value of the <i>buf</i> argument supplied to the macros is a pointer to a stat structure. The macro shall evaluate to a non-zero value if the specified object is implemented as a distinct file type and the specified file type is contained in the stat structure referenced by <i>buf</i> . Otherwise, the macro shall evaluate to zero.	
13193			
13194			
13195			
13196			
13197		<code>S_TYPEISTMO(<i>buf</i>)</code>	Test macro for a typed memory object.
13198		The <sys/stat.h> header shall define the following symbolic constants as distinct integer values outside of the range [0,999 999 999], for use with the <i>futimens()</i> and <i>utimensat()</i> functions:	
13199			
13200		<code>UTIME_NOW</code>	
13201		<code>UTIME_OMIT</code>	
13202		The following shall be declared as functions and may also be defined as macros. Function prototypes shall be provided.	
13203			
13204		<code>int</code>	<code>chmod(const char *, mode_t);</code>
13205		<code>int</code>	<code>fchmod(int, mode_t);</code>
13206		<code>int</code>	<code>fchmodat(int, const char *, mode_t, int);</code>
13207		<code>int</code>	<code>fstat(int, struct stat *);</code>
13208		<code>int</code>	<code>fstatat(int, const char *restrict, struct stat *restrict, int);</code>
13209		<code>int</code>	<code>futimens(int, const struct timespec [2]);</code>
13210		<code>int</code>	<code>lstat(const char *restrict, struct stat *restrict);</code>
13211		<code>int</code>	<code>mkdir(const char *, mode_t);</code>
13212		<code>int</code>	<code>mkdirat(int, const char *, mode_t);</code>
13213		<code>int</code>	<code>mkfifo(const char *, mode_t);</code>
13214		<code>int</code>	<code>mkfifoat(int, const char *, mode_t);</code>
13215	XSI	<code>int</code>	<code>mknod(const char *, mode_t, dev_t);</code>
13216		<code>int</code>	<code>mknodat(int, const char *, mode_t, dev_t);</code>
13217		<code>int</code>	<code>stat(const char *restrict, struct stat *restrict);</code>
13218		<code>mode_t</code>	<code>umask(mode_t);</code>
13219		<code>int</code>	<code>utimensat(int, const char *, const struct timespec [2], int);</code>

13220 APPLICATION USAGE

13221 Use of the macros is recommended for determining the type of a file.

13222 RATIONALE

13223 A conforming C-language application must include **<sys/stat.h>** for functions that have
 13224 arguments or return values of type **mode_t**, so that symbolic values for that type can be used.
 13225 An alternative would be to require that these constants are also defined by including
 13226 **<sys/types.h>**.

13227 The **S_ISUID** and **S_ISGID** bits may be cleared on any write, not just on *open()*, as some
 13228 historical implementations do.

13229 System calls that update the time entry fields in the **stat** structure must be documented by the
 13230 implementors. POSIX-conforming systems should not update the time entry fields for functions
 13231 listed in the System Interfaces volume of POSIX.1-2008 unless the standard requires that they do,
 13232 except in the case of documented extensions to the standard.

13233 Upon assignment, file timestamps are immediately converted to the resolution of the file system
 13234 by truncation (i.e., the recorded time can be older than the actual time). For example, if the file
 13235 system resolution is 1 microsecond, then a conforming *stat()* must always return an
 13236 *st_mtim.tv_nsec* that is a multiple of 1000. Some older implementations returned higher-
 13237 resolution timestamps while the *inode* information was cached, and then spontaneously
 13238 truncated the *tv_nsec* fields when they were stored to and retrieved from disk, but this behavior
 13239 does not conform.

13240 Note that *st_dev* must be unique within a Local Area Network (LAN) in a “system” made up of
 13241 multiple computers’ file systems connected by a LAN.

13242 Networked implementations of a POSIX-conforming system must guarantee that all files visible
 13243 within the file tree (including parts of the tree that may be remotely mounted from other
 13244 machines on the network) on each individual processor are uniquely identified by the
 13245 combination of the *st_ino* and *st_dev* fields.

13246 The unit for the *st_blocks* member of the **stat** structure is not defined within POSIX.1-2008. In
 13247 some implementations it is 512 bytes. It may differ on a file system basis. There is no correlation
 13248 between values of the *st_blocks* and *st_blksize*, and the *f_bsize* (from **<sys/statvfs.h>**) structure
 13249 members.

13250 Traditionally, some implementations defined the multiplier for *st_blocks* in **<sys/param.h>** as the
 13251 symbol **DEV_BSIZE**.

13252 Some earlier versions of this standard did not specify values for the file mode bit macros. The
 13253 expectation was that some implementors might choose to use a different encoding for these bits
 13254 than the traditional one, and that new applications would use symbolic file modes instead of
 13255 numeric. This version of the standard specifies the traditional encoding, in recognition that
 13256 nearly 20 years after the first publication of this standard numeric file modes are still in
 13257 widespread use by application developers, and that all conforming implementations still use the
 13258 traditional encoding.

13259 FUTURE DIRECTIONS

13260 No new **S_IFMT** symbolic names for the file type values of **mode_t** will be defined by
 13261 POSIX.1-2008; if new file types are required, they will only be testable through *S_ISxx()* or
 13262 *S_TYPEISxxx()* macros instead.

SEE ALSO

<sys/statvfs.h>, <sys/types.h>, <time.h>

XSH *chmod()*, *fchmod()*, *fstat()*, *fstatat()*, *futimens()*, *mkdir()*, *mkfifo()*, *mknod()*, *umask()*

CHANGE HISTORY

First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 5

The DESCRIPTION is updated for alignment with the POSIX Realtime Extension.

The type of *st_blksize* is changed from **long** to **blksize_t**; the type of *st_blocks* is changed from **long** to **blkcnt_t**.

Issue 6

The S_TYPEISMQ(), S_TYPEISSEM(), and S_TYPEISSHM() macros are unconditionally mandated.

The Open Group Corrigendum U035/4 is applied. In the DESCRIPTION, the types **blksize_t** and **blkcnt_t** have been described.

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The **dev_t**, **ino_t**, **mode_t**, **nlink_t**, **uid_t**, **gid_t**, **off_t**, and **time_t** types are mandated.

S_IFSOCK and S_ISSOCK are added for sockets.

The description of **stat** structure members is changed to reflect contents when file type is a symbolic link.

The test macro S_TYPEISTMO is added for alignment with IEEE Std 1003.1j-2000.

The **restrict** keyword is added to the prototypes for *lstat()* and *stat()*.

The *lstat()* function is made mandatory.

IEEE Std 1003.1-2001/Cor 1-2002, item XBD/TC1/D6/17 is applied, adding text regarding the *st_blocks* member of the **stat** structure to the RATIONALE.

IEEE Std 1003.1-2001/Cor 2-2004, item XBD/TC2/D6/25 is applied, adding to the DESCRIPTION that the **timespec** structure may be defined as described in the <time.h> header.

Issue 7

SD5-XSH-ERN-161 is applied, updating the DESCRIPTION to clarify that the descriptions of the interfaces should be consulted in order to determine which structure members have meaningful values.

The *fchmodat()*, *fstatat()*, *mkdirat()*, *mkfifoat()*, *mknodat()*, and *utimensat()* functions are added from The Open Group Technical Standard, 2006, Extended API Set Part 2.

The *futimens()* function is added.

This reference page is clarified with respect to macros and symbolic constants.

Changes are made related to support for finegrained timestamps and the **UTIME_NOW** and **UTIME_OMIT** symbolic constants are added.

POSIX.1-2008, Technical Corrigendum 1, XBD/TC1-2008/0068 [207] is applied.

+

13301 **NAME**

13302 sys/statvfs.h — VFS File System information structure

13303 **SYNOPSIS**

13304 #include <sys/statvfs.h>

13305 **DESCRIPTION**13306 The <sys/statvfs.h> header shall define the **statvfs** structure, which shall include at least the
13307 following members:

13308	unsigned long	f_bsize	File system block size.
13309	unsigned long	f_frsize	Fundamental file system block size.
13310	fsblkcnt_t	f_blocks	Total number of blocks on file system in units of <i>f_frsize</i> .
13311	fsblkcnt_t	f_bfree	Total number of free blocks.
13312	fsblkcnt_t	f_bavail	Number of free blocks available to non-privileged process.
13313			
13314	fsfilcnt_t	f_files	Total number of file serial numbers.
13315	fsfilcnt_t	f_ffree	Total number of free file serial numbers.
13316	fsfilcnt_t	f_favail	Number of file serial numbers available to non-privileged process.
13317			
13318	unsigned long	f_fsid	File system ID.
13319	unsigned long	f_flag	Bit mask of <i>f_flag</i> values.
13320	unsigned long	f_namemax	Maximum filename length.

13321 The <sys/statvfs.h> header shall define the **fsblkcnt_t** and **fsfilcnt_t** types as described in
13322 <sys/types.h>.13323 The <sys/statvfs.h> header shall define the following symbolic constants for the *f_flag* member:

13324 ST_RDONLY Read-only file system.

13325 ST_NOSUID Does not support the semantics of the ST_ISUID and ST_ISGID file mode bits.

13326 The following shall be declared as functions and may also be defined as macros. Function
13327 prototypes shall be provided.

13328 int fstatvfs(int, struct statvfs *);

13329 int statvfs(const char *restrict, struct statvfs *restrict);

13330 **APPLICATION USAGE**

13331 None.

13332 **RATIONALE**

13333 None.

13334 **FUTURE DIRECTIONS**

13335 None.

13336 **SEE ALSO**

13337 <sys/types.h>

13338 XSH *fstatvfs()*13339 **CHANGE HISTORY**

13340 First released in Issue 4, Version 2.

13341 **Issue 5**13342 The type of *f_blocks*, *f_bfree*, and *f_bavail* is changed from **unsigned long** to **fsblkcnt_t**; the type of
13343 *f_files*, *f_ffree*, and *f_favail* is changed from **unsigned long** to **fsfilcnt_t**.

13344 **Issue 6**

13345 The Open Group Corrigendum U035/5 is applied. In the DESCRIPTION, the types **fsblkcnt_t**
13346 and **fsfilcnt_t** have been described.

13347 The **restrict** keyword is added to the prototype for *statvfs()*.

13348 IEEE Std 1003.1-2001/Cor 1-2002, item XBD/TC1/D6/18 is applied, changing the description of
13349 ST_NOSUID from “Does not support *setuid()*/*setgid()* semantics” to “Does not support the
13350 semantics of the ST_ISUID and ST_ISGID file mode bits”.

13351 **Issue 7**

13352 The <sys/statvfs.h> header is moved from the XSI option to the Base.

13353 This reference page is clarified with respect to macros and symbolic constants.

13354 **NAME**

13355 sys/time.h — time types

13356 **SYNOPSIS**13357 XSI `#include <sys/time.h>`13358 **DESCRIPTION**

13359 The **<sys/time.h>** header shall define the **timeval** structure, which shall include at least the
 13360 following members:

13361	time_t	tv_sec	Seconds.
13362	suseconds_t	tv_usec	Microseconds.

13363 OB The **<sys/time.h>** header shall define the **itimerval** structure, which shall include at least the
 13364 following members:

13365	struct timeval	it_interval	Timer interval.
13366	struct timeval	it_value	Current value.

13367 The **<sys/time.h>** header shall define the **time_t** and **suseconds_t** types as described in
 13368 **<sys/types.h>**.

13369 The **<sys/time.h>** header shall define the **fd_set** type as described in **<sys/select.h>**.

13370 OB The **<sys/time.h>** header shall define the following symbolic constants for the *which* argument of
 13371 *getitimer()* and *setitimer()*:

13372	ITIMER_REAL	Decrements in real time.
13373	ITIMER_VIRTUAL	Decrements in process virtual time.
13374	ITIMER_PROF	Decrements both in process virtual time and when the system is running 13375 on behalf of the process.

13376 The **<sys/time.h>** header shall define the following as described in **<sys/select.h>**:

```
13377 FD_CLR()
13378 FD_ISSET()
13379 FD_SET()
13380 FD_ZERO()
13381 FD_SETSIZE
```

13382 The following shall be declared as functions and may also be defined as macros. Function
 13383 prototypes shall be provided.

```
13384 OB int getitimer(int, struct itimerval *);
13385 int gettimeofday(struct timeval *restrict, void *restrict);
13386 int setitimer(int, const struct itimerval *restrict,
13387             struct itimerval *restrict);
13388 int select(int, fd_set *restrict, fd_set *restrict, fd_set *restrict,
13389           struct timeval *restrict);
13390 int utimes(const char *, const struct timeval [2]);
```

13391 Inclusion of the **<sys/time.h>** header may make visible all symbols from the **<sys/select.h>**
 13392 header.

13393 **APPLICATION USAGE**

13394 None.

13395 **RATIONALE**

13396 None.

13397 **FUTURE DIRECTIONS**

13398 None.

13399 **SEE ALSO**

13400 [<sys/select.h>](#), [<sys/types.h>](#)

13401 XSH [futimens\(\)](#), [getitimer\(\)](#), [gettimeofday\(\)](#), [pselect\(\)](#)

13402 **CHANGE HISTORY**

13403 First released in Issue 4, Version 2.

13404 **Issue 5**

13405 The type of *tv_usec* is changed from **long** to **suseconds_t**.

13406 **Issue 6**

13407 The **restrict** keyword is added to the prototypes for *gettimeofday()*, *select()*, and *setitimer()*.

13408 The note is added that inclusion of this header may also make symbols visible from
13409 **<sys/select.h>**.

13410 The *utimes()* function is marked LEGACY.

13411 **Issue 7**

13412 This reference page is clarified with respect to macros and symbolic constants.

13413 **NAME**

13414 sys/times.h — file access and modification times structure

13415 **SYNOPSIS**

13416 #include <sys/times.h>

13417 **DESCRIPTION**13418 The <sys/times.h> header shall define the **tms** structure, which is returned by *times()* and shall
13419 include at least the following members:

13420 clock_t tms_utime User CPU time.

13421 clock_t tms_stime System CPU time.

13422 clock_t tms_cutime User CPU time of terminated child processes.

13423 clock_t tms_cstime System CPU time of terminated child processes.

13424 The <sys/times.h> header shall define the **clock_t** type as described in <sys/types.h>.13425 The following shall be declared as a function and may also be defined as a macro. A function
13426 prototype shall be provided.

13427 clock_t times(struct tms *);

13428 **APPLICATION USAGE**

13429 None.

13430 **RATIONALE**

13431 None.

13432 **FUTURE DIRECTIONS**

13433 None.

13434 **SEE ALSO**

13435 <sys/types.h>

13436 XSH *times()*13437 **CHANGE HISTORY**

13438 First released in Issue 1. Derived from Issue 1 of the SVID.

13439 **NAME**

13440 sys/types.h — data types

13441 **SYNOPSIS**

13442 #include <sys/types.h>

13443 **DESCRIPTION**

13444 The <sys/types.h> header shall define at least the following types:

13445	blkcnt_t	Used for file block counts.
13446	blksize_t	Used for block sizes.
13447	clock_t	Used for system times in clock ticks or CLOCKS_PER_SEC; see <time.h>.
13448		
13449	clockid_t	Used for clock ID type in the clock and timer functions.
13450	dev_t	Used for device IDs.
13451	fsblkcnt_t	Used for file system block counts.
13452	fsfilcnt_t	Used for file system file counts.
13453	gid_t	Used for group IDs.
13454	id_t	Used as a general identifier; can be used to contain at least a pid_t ,
13455		uid_t , or gid_t .
13456	ino_t	Used for file serial numbers.
13457	XSI key_t	Used for XSI interprocess communication.
13458	mode_t	Used for some file attributes.
13459	nlink_t	Used for link counts.
13460	off_t	Used for file sizes.
13461	pid_t	Used for process IDs and process group IDs.
13462	pthread_attr_t	Used to identify a thread attribute object.
13463	pthread_barrier_t	Used to identify a barrier.
13464	pthread_barrierattr_t	Used to define a barrier attributes object.
13465	pthread_cond_t	Used for condition variables.
13466	pthread_condattr_t	Used to identify a condition attribute object.
13467	pthread_key_t	Used for thread-specific data keys.
13468	pthread_mutex_t	Used for mutexes.
13469	pthread_mutexattr_t	Used to identify a mutex attribute object.
13470	pthread_once_t	Used for dynamic package initialization.
13471	pthread_rwlock_t	Used for read-write locks.
13472	pthread_rwlockattr_t	Used for read-write lock attributes.
13473	pthread_spinlock_t	Used to identify a spin lock.

13474		pthread_t	Used to identify a thread.	
13475		size_t	Used for sizes of objects.	
13476		ssize_t	Used for a count of bytes or an error indication.	
13477	XSI	suseconds_t	Used for time in microseconds.	
13478		time_t	Used for time in seconds.	
13479		timer_t	Used for timer ID returned by <i>timer_create()</i> .	
13480	OB TRC	trace_attr_t	Used to identify a trace stream attributes object	
13481	OB TRC	trace_event_id_t	Used to identify a trace event type.	
13482	OB TEF	trace_event_set_t	Used to identify a trace event type set.	
13483	OB TRC	trace_id_t	Used to identify a trace stream.	
13484		uid_t	Used for user IDs.	
13485		All of the types shall be defined as arithmetic types of an appropriate length, with the following exceptions:		
13486				
13487		pthread_attr_t		
13488		pthread_barrier_t		
13489		pthread_barrierattr_t		
13490		pthread_cond_t		
13491		pthread_condattr_t		
13492		pthread_key_t		
13493		pthread_mutex_t		
13494		pthread_mutexattr_t		
13495		pthread_once_t		
13496		pthread_rwlock_t		
13497		pthread_rwlockattr_t		
13498		pthread_spinlock_t		
13499		pthread_t		
13500	OB TRC	trace_attr_t		
13501		trace_event_id_t		
13502	OB TEF	trace_event_set_t		
13503	OB TRC	trace_id_t		
13504		Additionally:		
13505		• mode_t shall be an integer type.		
13506		• dev_t shall be an integer type.		+
13507		• nlink_t , uid_t , gid_t , and id_t shall be integer types.		
13508		• blkcnt_t and off_t shall be signed integer types.		
13509		• fsblkcnt_t , fsfilcnt_t , and ino_t shall be defined as unsigned integer types.		
13510		• size_t shall be an unsigned integer type.		
13511		• blksize_t , pid_t , and ssize_t shall be signed integer types.		

13512 CX • **clock_t** shall be an integer or real-floating type. **time_t** shall be an integer type. |

13513 The type **ssize_t** shall be capable of storing values at least in the range $[-1, \{\text{SSIZE\_MAX}\}]$.

13514 XSI The type **suseconds_t** shall be a signed integer type capable of storing values at least in the

13515 range $[-1, 1\,000\,000]$.

13516 The implementation shall support one or more programming environments in which the widths

13517 of **blksize_t**, **pid_t**, **size_t**, **ssize_t**, and **suseconds_t** are no greater than the width of type **long**.

13518 The names of these programming environments can be obtained using the *confstr()* function or

13519 the *getconf* utility.

13520 There are no defined comparison or assignment operators for the following types:

13521 **pthread_attr_t**

13522 **pthread_barrier_t**

13523 **pthread_barrierattr_t**

13524 **pthread_cond_t**

13525 **pthread_condattr_t**

13526 **pthread_mutex_t**

13527 **pthread_mutexattr_t**

13528 **pthread_rwlock_t**

13529 **pthread_rwlockattr_t**

13530 **pthread_spinlock_t**

13531 OB TRC **trace_attr_t**

13532 APPLICATION USAGE

13533 None.

13534 RATIONALE

13535 None.

13536 FUTURE DIRECTIONS

13537 None.

13538 SEE ALSO

13539 <time.h>

13540 XSH *confstr()*

13541 XCU *getconf*

13542 CHANGE HISTORY

13543 First released in Issue 1. Derived from Issue 1 of the SVID.

13544 Issue 5

13545 The **clockid_t** and **timer_t** types are defined for alignment with the POSIX Realtime Extension.

13546 The types **blkcnt_t**, **blksize_t**, **fsblkcnt_t**, **fsfilcnt_t**, and **suseconds_t** are added.

13547 Large File System extensions are added.

13548 Updated for alignment with the POSIX Threads Extension.

13549 Issue 6

13550 The **pthread_barrier_t**, **pthread_barrierattr_t**, and **pthread_spinlock_t** types are added for

13551 alignment with IEEE Std 1003.1j-2000.

13552 The margin code is changed from XSI to THR for the **pthread_rwlock_t** and

13553 **pthread_rwlockattr_t** types as Read-Write Locks have been absorbed into the POSIX Threads

13554 option. The threads types are marked THR.

13555 IEEE Std 1003.1-2001/Cor 2-2004, item XBD/TC2/D6/26 is applied, adding **pthread_t** to the list
13556 of types that are not required to be arithmetic types, thus allowing **pthread_t** to be defined as a
13557 structure.

13558 **Issue 7**

13559 Austin Group Interpretation 1003.1-2001 #033 is applied, requiring **key_t** to be an arithmetic
13560 type.

13561 The Trace option types are marked obsolescent.

13562 The **clock_t** and **id_t** types are moved from the XSI option to the Base.

13563 The **pthread_barrier_t** and **pthread_barrierattr_t** types are moved from the Barriers option to
13564 the Base.

13565 The **pthread_spinlock_t** type is moved from the Spin Locks option to the Base.

13566 Functionality relating to the Timers and Threads options is moved to the Base.

13567 POSIX.1-2008, Technical Corrigendum 1, XBD/TC1-2008/0069 [210], XBD/TC1-2008/0070 [28], +
13568 XBD/TC1-2008/0071 [376], XBD/TC1-2008/0072 [210], and XBD/TC1-2008/0073 [327] are +
13569 applied.

13570 NAME

13571 sys/uio.h — definitions for vector I/O operations

13572 SYNOPSIS

13573 XSI `#include <sys/uio.h>`

13574 DESCRIPTION

13575 The <sys/uio.h> header shall define the **iovec** structure, which shall include at least the
13576 following members:

13577 `void *iov_base` Base address of a memory region for input or output.
13578 `size_t iov_len` The size of the memory pointed to by *iov_base*.

13579 The <sys/uio.h> header uses the **iovec** structure for scatter/gather I/O.

13580 The <sys/uio.h> header shall define the **ssize_t** and **size_t** types as described in <sys/types.h>.

13581 The following shall be declared as functions and may also be defined as macros. Function
13582 prototypes shall be provided.

13583 `ssize_t readv(int, const struct iovec *, int);`
13584 `ssize_t writev(int, const struct iovec *, int);`

13585 APPLICATION USAGE

13586 The implementation can put a limit on the number of scatter/gather elements which can be
13587 processed in one call. The symbol {IOV_MAX} defined in <limits.h> should always be used to
13588 learn about the limits instead of assuming a fixed value.

13589 RATIONALE

13590 Traditionally, the maximum number of scatter/gather elements the system can process in one
13591 call were described by the symbolic value {UIO_MAXIOV}. In IEEE Std 1003.1-2001 this value is
13592 replaced by the constant {IOV_MAX} which can be found in <limits.h>.

13593 FUTURE DIRECTIONS

13594 None.

13595 SEE ALSO

13596 <limits.h>, <sys/types.h>

13597 XSH *read()*, *readv()*, *write()*, *writev()*

13598 CHANGE HISTORY

13599 First released in Issue 4, Version 2.

13600 Issue 6

13601 Text referring to scatter/gather I/O is added to the DESCRIPTION.

13602 **NAME**

13603 sys/un.h — definitions for UNIX domain sockets

13604 **SYNOPSIS**

13605 #include <sys/un.h>

13606 **DESCRIPTION**

13607 The <sys/un.h> header shall define the **sockaddr_un** structure, which shall include at least the
 13608 following members:

13609 sa_family_t sun_family Address family.
 13610 char sun_path[] Socket pathname.

13611 The **sockaddr_un** structure is used to store addresses for UNIX domain sockets. Pointers to this
 13612 type shall be cast by applications to **struct sockaddr *** for use with socket functions.

13613 The <sys/un.h> header shall define the **sa_family_t** type as described in <sys/socket.h>.

13614 **APPLICATION USAGE**

13615 The size of *sun_path* has intentionally been left undefined. This is because different
 13616 implementations use different sizes. For example, 4.3 BSD uses a size of 108, and 4.4 BSD uses a
 13617 size of 104. Since most implementations originate from BSD versions, the size is typically in the
 13618 range 92 to 108.

13619 Applications should not assume a particular length for *sun_path* or assume that it can hold
 13620 {_POSIX_PATH_MAX} bytes (256).

13621 **RATIONALE**

13622 None.

13623 **FUTURE DIRECTIONS**

13624 None.

13625 **SEE ALSO**

13626 <sys/socket.h>

13627 XSH *bind()*, *socket()*, *socketpair()*13628 **CHANGE HISTORY**

13629 First released in Issue 6. Derived from the XNS, Issue 5.2 specification.

13630 **Issue 7**

13631 The value for {_POSIX_PATH_MAX} is updated to 256.

13632 POSIX.1-2008, Technical Corrigendum 1, XBD/TC1-2008/0074 [355] is applied.

+

13633 **NAME**

13634 sys/utsname.h — system name structure

13635 **SYNOPSIS**

13636 #include <sys/utsname.h>

13637 **DESCRIPTION**

13638 The <sys/utsname.h> header shall define the structure **utsname** which shall include at least the
13639 following members:

13640	char	sysname[]	Name of this implementation of the operating system.
13641	char	nodename[]	Name of this node within the communications
13642			network to which this node is attached, if any.
13643	char	release[]	Current release level of this implementation.
13644	char	version[]	Current version level of this release.
13645	char	machine[]	Name of the hardware type on which the system is running.

13646 The character arrays are of unspecified size, but the data stored in them shall be terminated by a
13647 null byte.

13648 The following shall be declared as a function and may also be defined as a macro:

13649 int uname(struct utsname *);

13650 **APPLICATION USAGE**

13651 None.

13652 **RATIONALE**

13653 None.

13654 **FUTURE DIRECTIONS**

13655 None.

13656 **SEE ALSO**

13657 XSH *uname()*

13658 **CHANGE HISTORY**

13659 First released in Issue 1. Derived from Issue 1 of the SVID.

13660 **Issue 6**

13661 IEEE Std 1003.1-2001/Cor 2-2004, item XBD/TC2/D6/27 is applied, changing the description of
13662 *nodename* within the **utsname** structure from “an implementation-defined communications
13663 network” to “the communications network to which this node is attached, if any”.

13664 **NAME**

13665 sys/wait.h — declarations for waiting

13666 **SYNOPSIS**

13667 #include <sys/wait.h>

13668 **DESCRIPTION**13669 The <sys/wait.h> header shall define the following symbolic constants for use with *waitpid()*:13670 XSI **WCONTINUED** Report status of continued child process.13671 **WNOHANG** Do not hang if no status is available; return immediately.13672 **WUNTRACED** Report status of stopped child process.

13673 The <sys/wait.h> header shall define the following macros for analysis of process status values:

13674 **WEXITSTATUS** Return exit status.13675 XSI **WIFCONTINUED** True if child has been continued.13676 **WIFEXITED** True if child exited normally.13677 **WIFSIGNALED** True if child exited due to uncaught signal.13678 **WIFSTOPPED** True if child is currently stopped.13679 **WSTOPSIG** Return signal number that caused process to stop.13680 **WTERMSIG** Return signal number that caused process to terminate.13681 The <sys/wait.h> header shall define the following symbolic constants as possible values for the
13682 *options* argument to *waitid()*:13683 **WEXITED** Wait for processes that have exited.13684 **WNOWAIT** Keep the process whose status is returned in *info* in a waitable state.13685 **WSTOPPED** Status is returned for any child that has stopped upon receipt of a signal.13686 XSI The **WCONTINUED** and **WNOHANG** constants, described above for *waitpid()*, can also be
13687 used with *waitid()*.13688 The type **idtype_t** shall be defined as an enumeration type whose possible values shall include
13689 at least the following:13690 **P_ALL**13691 **P_PGID**13692 **P_PID**13693 The <sys/wait.h> header shall define the **id_t** and **pid_t** types as described in <sys/types.h>.13694 The <sys/wait.h> header shall define the **siginfo_t** type as described in <signal.h>.

13695 Inclusion of the <sys/wait.h> header may also make visible all symbols from <signal.h>.

13696 The following shall be declared as functions and may also be defined as macros. Function
13697 prototypes shall be provided.13698 **pid_t** wait(int *);13699 **int** waitid(idtype_t, id_t, siginfo_t *, int);13700 **pid_t** waitpid(pid_t, int *, int);

13701 **APPLICATION USAGE**

13702 None.

13703 **RATIONALE**

13704 None.

13705 **FUTURE DIRECTIONS**

13706 None.

13707 **SEE ALSO**

13708 <signal.h>, <sys/types.h>

13709 XSH *wait()*, *waitid()*

13710 **CHANGE HISTORY**

13711 First released in Issue 3.

13712 Included for alignment with the POSIX.1-1988 standard.

13713 **Issue 6**

13714 The *wait3()* function is removed.

13715 **Issue 7**

13716 The *waitid()* function and symbolic constants for its *options* argument are moved to the Base.

13717 The description of the WNOHANG constant is clarified.

13718 **NAME**

13719 syslog.h — definitions for system error logging

13720 **SYNOPSIS**

13721 XSI #include <syslog.h>

13722 **DESCRIPTION**13723 The <syslog.h> header shall define the following symbolic constants, zero or more of which
13724 may be OR'ed together to form the *logopt* option of *openlog()*:

13725 LOG_PID Log the process ID with each message.

13726 LOG_CONS Log to the system console on error.

13727 LOG_NDELAY Connect to syslog daemon immediately.

13728 LOG_ODELAY Delay open until *syslog()* is called.

13729 LOG_NOWAIT Do not wait for child processes.

13730 The <syslog.h> header shall define the following symbolic constants for use as the *facility*
13731 argument to *openlog()*:

13732 LOG_KERN Reserved for message generated by the system.

13733 LOG_USER Message generated by a process.

13734 LOG_MAIL Reserved for message generated by mail system.

13735 LOG_NEWS Reserved for message generated by news system.

13736 LOG_UUCP Reserved for message generated by UUCP system.

13737 LOG_DAEMON Reserved for message generated by system daemon.

13738 LOG_AUTH Reserved for message generated by authorization daemon.

13739 LOG_CRON Reserved for message generated by clock daemon.

13740 LOG_LPR Reserved for message generated by printer system.

13741 LOG_LOCAL0 Reserved for local use.

13742 LOG_LOCAL1 Reserved for local use.

13743 LOG_LOCAL2 Reserved for local use.

13744 LOG_LOCAL3 Reserved for local use.

13745 LOG_LOCAL4 Reserved for local use.

13746 LOG_LOCAL5 Reserved for local use.

13747 LOG_LOCAL6 Reserved for local use.

13748 LOG_LOCAL7 Reserved for local use.

13749 The <syslog.h> header shall define the following macros for constructing the *maskpri* argument
13750 to *setlogmask()*. The following macros expand to an expression of type **int** when the argument
13751 *pri* is an expression of type **int**:13752 LOG_MASK(*pri*) A mask for priority *pri*.13753 The <syslog.h> header shall define the following symbolic constants for use as the *priority*
13754 argument of *syslog()*:

13755	LOG_EMERG	A panic condition was reported to all processes.
13756	LOG_ALERT	A condition that should be corrected immediately.
13757	LOG_CRIT	A critical condition.
13758	LOG_ERR	An error message.
13759	LOG_WARNING	A warning message.
13760	LOG_NOTICE	A condition requiring special handling.
13761	LOG_INFO	A general information message.
13762	LOG_DEBUG	A message useful for debugging programs.
13763	The following shall be declared as functions and may also be defined as macros. Function	
13764	prototypes shall be provided.	
13765	void closelog(void);	
13766	void openlog(const char *, int, int);	
13767	int setlogmask(int);	
13768	void syslog(int, const char *, ...);	
13769	APPLICATION USAGE	
13770	None.	
13771	RATIONALE	
13772	None.	
13773	FUTURE DIRECTIONS	
13774	None.	
13775	SEE ALSO	
13776	XSH <i>closelog()</i>	
13777	CHANGE HISTORY	
13778	First released in Issue 4, Version 2.	
13779	Issue 5	
13780	Moved from X/Open UNIX to BASE.	
13781	Issue 7	
13782	This reference page is clarified with respect to macros and symbolic constants.	

13783 **NAME**

13784 tar.h — extended tar definitions

13785 **SYNOPSIS**

13786 #include <tar.h>

13787 **DESCRIPTION**

13788 The <tar.h> header shall define the following symbolic constants with the indicated values.

13789 General definitions:

13790

13791

13792

13793

13794

Name	Value	Description
TMAGIC	"ustar"	ustar plus null byte.
TMAGLEN	6	Length of the above.
TVERSION	"00"	00 without a null byte.
TVERSLEN	2	Length of the above.

13795

Typeflag field definitions:

13796

13797

13798

13799

13800

13801

13802

13803

13804

13805

Name	Value	Description
REGTYPE	'0'	Regular file.
AREGTYPE	'\0'	Regular file.
LNKTYPE	'1'	Link.
SYMTYPE	'2'	Symbolic link.
CHRTYPE	'3'	Character special.
BLKTYPE	'4'	Block special.
DIRTYPE	'5'	Directory.
FIFOTYPE	'6'	FIFO special.
CONTTYPE	'7'	Reserved.

13806

Mode field bit definitions (octal):

13807

13808

13809

13810 XSI

13811

13812

13813

13814

13815

13816

13817

13818

13819

Name	Value	Description
TSUID	04000	Set UID on execution.
TSGID	02000	Set GID on execution.
TSVTX	01000	On directories, restricted deletion flag.
TUREAD	00400	Read by owner.
TUWRITE	00200	Write by owner special.
TUEXEC	00100	Execute/search by owner.
TGREAD	00040	Read by group.
TGWRITE	00020	Write by group.
TGEXEC	00010	Execute/search by group.
TOREAD	00004	Read by other.
TOWRITE	00002	Write by other.
TOEXEC	00001	Execute/search by other.

13820 **APPLICATION USAGE**

13821 None.

13822 **RATIONALE**

13823 None.

13824 **FUTURE DIRECTIONS**

13825 None.

13826 **SEE ALSO**

13827 XCU *pax*

13828 **CHANGE HISTORY**

13829 First released in Issue 3. Derived from the POSIX.1-1988 standard.

13830 **Issue 6**

13831 The SEE ALSO section is updated to refer to *pax*.

13832 **Issue 7**

13833 This reference page is clarified with respect to macros and symbolic constants.

13834 **NAME**

13835 termios.h — define values for termios

13836 **SYNOPSIS**

13837 #include <termios.h>

13838 **DESCRIPTION**

13839 The <termios.h> header shall contain the definitions used by the terminal I/O interfaces (see
 13840 [Chapter 11](#) (on page 199) for the structures and names defined).

13841 **The termios Structure**

13842 The <termios.h> header shall define the following data types through **typedef**:

13843 **cc_t** Used for terminal special characters.

13844 **speed_t** Used for terminal baud rates.

13845 **tcflag_t** Used for terminal modes.

13846 The above types shall be all unsigned integer types.

13847 The implementation shall support one or more programming environments in which the widths
 13848 of **cc_t**, **speed_t**, and **tcflag_t** are no greater than the width of type **long**. The names of these
 13849 programming environments can be obtained using the *confstr()* function or the *getconf* utility.

13850 The <termios.h> header shall define the **termios** structure, which shall include at least the
 13851 following members:

13852	tcflag_t	c_iflag	Input modes.
13853	tcflag_t	c_oflag	Output modes.
13854	tcflag_t	c_cflag	Control modes.
13855	tcflag_t	c_lflag	Local modes.
13856	cc_t	c_cc[NCCS]	Control characters.

13857 The <termios.h> header shall define the following symbolic constant:

13858 **NCCS** Size of the array **c_cc** for control characters.

13859 The <termios.h> header shall define the following symbolic constants for use as subscripts for
 13860 the array **c_cc**:

	Subscript Usage		Description
	Canonical Mode	Non-Canonical Mode	
13861	VEOF	VINTR	EOF character.
13862	VEOL		EOL character.
13863	VERASE		ERASE character.
13864	VINTR		INTR character.
13865	VKILL		KILL character.
13866		VMIN	MIN value.
13867	VQUIT	VQUIT	QUIT character.
13868	VSTART	VSTART	START character.
13869	VSTOP	VSTOP	STOP character.
13870	VSUSP	VSUSP	SUSP character.
13871		VTIME	TIME value.

13874 The subscript values shall be suitable for use in **#if** preprocessing directives and shall be distinct,
 13875 except that the VMIN and VTIME subscripts may have the same values as the VEOF and VEOL
 13876 subscripts, respectively.

13877 **Input Modes**

13878 The <termios.h> header shall define the following symbolic constants for use as flags in the
 13879 *c_iflag* field. The *c_iflag* field describes the basic terminal input control.

13880	BRKINT	Signal interrupt on break.
13881	ICRNL	Map CR to NL on input.
13882	IGNBRK	Ignore break condition.
13883	IGNCR	Ignore CR.
13884	IGNPAR	Ignore characters with parity errors.
13885	INLCR	Map NL to CR on input.
13886	INPCK	Enable input parity check.
13887	ISTRIP	Strip character.
13888	IXANY	Enable any character to restart output.
13889	IXOFF	Enable start/stop input control.
13890	IXON	Enable start/stop output control.
13891	PARMRK	Mark parity errors.

13892 **Output Modes**

13893 The <termios.h> header shall define the following symbolic constants for use as flags in the
 13894 *c_oflag* field. The *c_oflag* field specifies the system treatment of output.

13895	OPOST	Post-process output.
13896	XSI ONLCR	Map NL to CR-NL on output.
13897	XSI OCRNL	Map CR to NL on output.
13898	XSI ONOCR	No CR output at column 0.
13899	XSI ONLRET	NL performs CR function.
13900	XSI OFDEL	Fill is DEL.
13901	XSI OFILL	Use fill characters for delay.
13902	XSI NLDLY	Select newline delays:
13903		NL0 Newline type 0.
13904		NL1 Newline type 1.
13905	XSI CRDLY	Select carriage-return delays:
13906		CR0 Carriage-return delay type 0.
13907		CR1 Carriage-return delay type 1.
13908		CR2 Carriage-return delay type 2.
13909		CR3 Carriage-return delay type 3.

13910	XSI	TABDLY	Select horizontal-tab delays:
13911		TAB0	Horizontal-tab delay type 0.
13912		TAB1	Horizontal-tab delay type 1.
13913		TAB2	Horizontal-tab delay type 2.
13914		TAB3	Expand tabs to spaces.
13915	XSI	BSDLY	Select backspace delays:
13916		BS0	Backspace-delay type 0.
13917		BS1	Backspace-delay type 1.
13918	XSI	VTDLY	Select vertical-tab delays:
13919		VT0	Vertical-tab delay type 0.
13920		VT1	Vertical-tab delay type 1.
13921	XSI	FFDLY	Select form-feed delays:
13922		FF0	Form-feed delay type 0.
13923		FF1	Form-feed delay type 1.

13924 **Baud Rate Selection**

13925 The <termios.h> header shall define the following symbolic constants for use as values of
 13926 objects of type **speed_t**.

13927 The input and output baud rates are stored in the **termios** structure. These are the valid values
 13928 for objects of type **speed_t**. Not all baud rates need be supported by the underlying hardware.

13929	B0	Hang up
13930	B50	50 baud
13931	B75	75 baud
13932	B110	110 baud
13933	B134	134.5 baud
13934	B150	150 baud
13935	B200	200 baud
13936	B300	300 baud
13937	B600	600 baud
13938	B1200	1 200 baud
13939	B1800	1 800 baud
13940	B2400	2 400 baud

13941	B4800	4 800 baud
13942	B9600	9 600 baud
13943	B19200	19 200 baud
13944	B38400	38 400 baud

13945 **Control Modes**

13946 The <termios.h> header shall define the following symbolic constants for use as flags in the
 13947 *c_cflag* field. The *c_cflag* field describes the hardware control of the terminal; not all values
 13948 specified are required to be supported by the underlying hardware.

13949	CSIZE	Character size:
13950		CS5 5 bits
13951		CS6 6 bits
13952		CS7 7 bits
13953		CS8 8 bits
13954	CSTOPB	Send two stop bits, else one.
13955	CREAD	Enable receiver.
13956	PARENB	Parity enable.
13957	PARODD	Odd parity, else even.
13958	HUPCL	Hang up on last close.
13959	CLOCAL	Ignore modem status lines.

13960 The implementation shall support the functionality associated with the symbols CS7, CS8,
 13961 CSTOPB, PARODD, and PARENB.

13962 **Local Modes**

13963 The <termios.h> header shall define the following symbolic constants for use as flags in the
 13964 *c_lflag* field. The *c_lflag* field of the argument structure is used to control various terminal
 13965 functions.

13966	ECHO	Enable echo.
13967	ECHOE	Echo erase character as error-correcting backspace.
13968	ECHOK	Echo KILL.
13969	ECHONL	Echo NL.
13970	ICANON	Canonical input (erase and kill processing).
13971	IEXTEN	Enable extended input character processing.
13972	ISIG	Enable signals.
13973	NOFLSH	Disable flush after interrupt or quit.
13974	TOSTOP	Send SIGTTOU for background output.

Attribute Selection

The <termios.h> header shall define the following symbolic constants for use with *tcsetattr()*:

TCSANOW Change attributes immediately.
 TCSADRAIN Change attributes when output has drained.
 TCSAFLUSH Change attributes when output has drained; also flush pending input.

Line Control

The <termios.h> header shall define the following symbolic constants for use with *tcflush()*:

TCIFLUSH Flush pending input.
 TCIOFLUSH Flush both pending input and untransmitted output.
 TCOFLUSH Flush untransmitted output.

The <termios.h> header shall define the following symbolic constants for use with *tcflow()*:

TCIOFF Transmit a STOP character, intended to suspend input data.
 TCION Transmit a START character, intended to restart input data.
 TCOOFF Suspend output.
 TCOON Restart output.

The <termios.h> header shall define the **pid_t** type as described in <sys/types.h>.

The following shall be declared as functions and may also be defined as macros. Function prototypes shall be provided.

```
speed_t cfgetispeed(const struct termios *);
speed_t cfgetospeed(const struct termios *);
int      cfsetispeed(struct termios *, speed_t);
int      cfsetospeed(struct termios *, speed_t);
int      tcdrain(int);
int      tcflow(int, int);
int      tcflush(int, int);
int      tcgetattr(int, struct termios *);
pid_t    tcgetsid(int);
int      tcsendbreak(int, int);
int      tcsetattr(int, int, const struct termios *);
```

APPLICATION USAGE

The following names are reserved for XSI-conformant systems to use as an extension to the above; therefore strictly conforming applications shall not use them:

CBAUD	EXTB	VDSUSP
DEFECHO	FLUSHO	VLNEXT
ECHOCTL	LOBLK	VREPRINT
ECHOK	PENDIN	VSTATUS
ECHOPRT	SWTCH	VWERASE
EXTA	VDISCARD	

14013	RATIONALE
14014	None.
14015	FUTURE DIRECTIONS
14016	None.
14017	SEE ALSO
14018	<sys/types.h>
14019	XSH <i>cfgetispeed()</i> , <i>cfgetospeed()</i> , <i>cfsetispeed()</i> , <i>cfsetospeed()</i> , <i>confstr()</i> , <i>tcdrain()</i> , <i>tcflow()</i> , <i>tcflush()</i> ,
14020	<i>tcgetattr()</i> , <i>tcgetsid()</i> , <i>tcsendbreak()</i> , <i>tcsetattr()</i>
14021	XCU Chapter 11 (on page 199), <i>getconf</i>
14022	CHANGE HISTORY
14023	First released in Issue 3.
14024	Included for alignment with the ISO POSIX-1 standard.
14025	Issue 6
14026	The LEGACY symbols IUCLC, OLCUC, and XCASE are removed.
14027	FIPS 151-2 requirements for the symbols CS7, CS8, CSTOPB, PARODD, and PARENB are
14028	reaffirmed.
14029	IEEE Std 1003.1-2001/Cor 1-2002, item XBD/TC1/D6/19 is applied, changing ECHOK to
14030	ECHOKE in the APPLICATION USAGE section.
14031	Issue 7
14032	Austin Group Interpretation 1003.1-2001 #144 is applied, moving functionality relating to the
14033	IXANY symbol from the XSI option to the Base.
14034	SD5-XBD-ERN-35 is applied, adding the OFDEL output mode.
14035	This reference page is clarified with respect to macros and symbolic constants, and a declaration
14036	for the pid_t type is added.

14037 **NAME**14038 `tgmath.h` — type-generic macros14039 **SYNOPSIS**14040 `#include <tgmath.h>`14041 **DESCRIPTION**

14042 CX The functionality described on this reference page is aligned with the ISO C standard. Any
 14043 conflict between the requirements described here and the ISO C standard is unintentional. This
 14044 volume of POSIX.1-2008 defers to the ISO C standard.

14045 The **<tgmath.h>** header shall include the headers **<math.h>** and **<complex.h>** and shall define
 14046 several type-generic macros.

14047 Of the functions contained within the **<math.h>** and **<complex.h>** headers without an *f* (**float**)
 14048 or *l* (**long double**) suffix, several have one or more parameters whose corresponding real type is
 14049 XSI **double**. For each such function, except *modf()*, *j0()*, *j1()*, *jn()*, *y0()*, *y1()*, and *yn()*, there shall
 14050 be a corresponding type-generic macro. The parameters whose corresponding real type is
 14051 **double** in the function synopsis are generic parameters. Use of the macro invokes a function
 14052 whose corresponding real type and type domain are determined by the arguments for the
 14053 generic parameters.

14054 Use of the macro invokes a function whose generic parameters have the corresponding real type
 14055 determined as follows:

- 14056 • First, if any argument for generic parameters has type **long double**, the type determined is
 14057 **long double**.
- 14058 • Otherwise, if any argument for generic parameters has type **double** or is of integer type,
 14059 the type determined is **double**.
- 14060 • Otherwise, the type determined is **float**.

14061 For each unsuffixed function in the **<math.h>** header for which there is a function in the
 14062 **<complex.h>** header with the same name except for a *c* prefix, the corresponding type-generic
 14063 macro (for both functions) has the same name as the function in the **<math.h>** header. The
 14064 corresponding type-generic macro for *fabs()* and *cabs()* is *fabs()*.

<math.h> Function	<complex.h> Function	Type-Generic Macro
<i>acos()</i>	<i>cacos()</i>	<i>acos()</i>
<i>asin()</i>	<i>casin()</i>	<i>asin()</i>
<i>atan()</i>	<i>catan()</i>	<i>atan()</i>
<i>acosh()</i>	<i>cacosh()</i>	<i>acosh()</i>
<i>asinh()</i>	<i>casinh()</i>	<i>asinh()</i>
<i>atanh()</i>	<i>catanh()</i>	<i>atanh()</i>
<i>cos()</i>	<i>ccos()</i>	<i>cos()</i>
<i>sin()</i>	<i>csin()</i>	<i>sin()</i>
<i>tan()</i>	<i>ctan()</i>	<i>tan()</i>
<i>cosh()</i>	<i>ccosh()</i>	<i>cosh()</i>
<i>sinh()</i>	<i>csinh()</i>	<i>sinh()</i>
<i>tanh()</i>	<i>ctanh()</i>	<i>tanh()</i>
<i>exp()</i>	<i>cexp()</i>	<i>exp()</i>
<i>log()</i>	<i>clog()</i>	<i>log()</i>
<i>pow()</i>	<i>cpow()</i>	<i>pow()</i>
<i>sqrt()</i>	<i>csqrt()</i>	<i>sqrt()</i>
<i>fabs()</i>	<i>cabs()</i>	<i>fabs()</i>

If at least one argument for a generic parameter is complex, then use of the macro invokes a complex function; otherwise, use of the macro invokes a real function.

For each unsuffixed function in the **<math.h>** header without a *c*-prefixed counterpart in the **<complex.h>** header, except for *modf()*, *j0()*, *j1()*, *jn()*, *y0()*, *y1()*, and *yn()*, the corresponding type-generic macro has the same name as the function. These type-generic macros are:

<i>atan2()</i>	<i>fma()</i>	<i>llround()</i>	<i>remainder()</i>
<i>cbrt()</i>	<i>fmax()</i>	<i>log10()</i>	<i>remquo()</i>
<i>ceil()</i>	<i>fmin()</i>	<i>log1p()</i>	<i>rint()</i>
<i>copysign()</i>	<i>fmod()</i>	<i>log2()</i>	<i>round()</i>
<i>erf()</i>	<i>frexp()</i>	<i>logb()</i>	<i>scalbln()</i>
<i>erfc()</i>	<i>hypot()</i>	<i>lrint()</i>	<i>scalbn()</i>
<i>exp2()</i>	<i>ilogb()</i>	<i>lround()</i>	<i>tgamma()</i>
<i>expm1()</i>	<i>ldexp()</i>	<i>nearbyint()</i>	<i>trunc()</i>
<i>fdim()</i>	<i>lgamma()</i>	<i>nextafter()</i>	
<i>floor()</i>	<i>llrint()</i>	<i>nexttoward()</i>	

If all arguments for generic parameters are real, then use of the macro invokes a real function; otherwise, use of the macro results in undefined behavior.

For each unsuffixed function in the **<complex.h>** header that is not a *c*-prefixed counterpart to a function in the **<math.h>** header, the corresponding type-generic macro has the same name as the function. These type-generic macros are:

carg()
cimag()
conj()
cproj()
creal()

Use of the macro with any real or complex argument invokes a complex function.

APPLICATION USAGE

With the declarations:

```
#include <tgmath.h>
int n;
float f;
double d;
long double ld;
float complex fc;
double complex dc;
long double complex ldc;
```

functions invoked by use of type-generic macros are shown in the following table:

Macro	Use Invokes
<i>exp(n)</i>	<i>exp(n)</i> , the function
<i>acosh(f)</i>	<i>acoshf(f)</i>
<i>sin(d)</i>	<i>sin(d)</i> , the function
<i>atan(ld)</i>	<i>atanl(ld)</i>
<i>log(fc)</i>	<i>clogf(fc)</i>
<i>sqrt(dc)</i>	<i>csqrt(dc)</i>

Macro	Use Invokes
<i>pow(ldc,f)</i>	<i>cpowl(ldc,f)</i>
<i>remainder(n,n)</i>	<i>remainder(n,n)</i> , the function
<i>nextafter(d,f)</i>	<i>nextafter(d,f)</i> , the function
<i>nexttoward(f,ld)</i>	<i>nexttowardf(f,ld)</i>
<i>copysign(n,ld)</i>	<i>copysignl(n,ld)</i>
<i>ceil(fc)</i>	Undefined behavior
<i>rint(dc)</i>	Undefined behavior
<i>fmax(ldc,ld)</i>	Undefined behavior
<i>carg(n)</i>	<i>carg(n)</i> , the function
<i>cproj(f)</i>	<i>cprojf(f)</i>
<i>creal(d)</i>	<i>creal(d)</i> , the function
<i>cimag(ld)</i>	<i>cimagl(ld)</i>
<i>cabs(fc)</i>	<i>cabsf(fc)</i>
<i>carg(dc)</i>	<i>carg(dc)</i> , the function
<i>cproj(ldc)</i>	<i>cprojl(ldc)</i>

RATIONALE

Type-generic macros allow calling a function whose type is determined by the argument type, as is the case for C operators such as '+' and '*'. For example, with a type-generic *cos()* macro, the expression *cos((float)x)* will have type **float**. This feature enables writing more portably efficient code and alleviates need for awkward casting and suffixing in the process of porting or adjusting precision. Generic math functions are a widely appreciated feature of Fortran.

The only arguments that affect the type resolution are the arguments corresponding to the parameters that have type **double** in the synopsis. Hence the type of a type-generic call to *nexttoward()*, whose second parameter is **long double** in the synopsis, is determined solely by the type of the first argument.

The term “type-generic” was chosen over the proposed alternatives of intrinsic and overloading. The term is more specific than intrinsic, which already is widely used with a more general meaning, and reflects a closer match to Fortran’s generic functions than to C++ overloading.

The macros are placed in their own header in order not to silently break old programs that include the <math.h> header; for example, with:

```
printf ("%e", sin(x))
```

*modf(double, double *)* is excluded because no way was seen to make it safe without complicating the type resolution.

The implementation might, as an extension, endow appropriate ones of the macros that POSIX.1-2008 specifies only for real arguments with the ability to invoke the complex functions.

POSIX.1-2008 does not prescribe any particular implementation mechanism for generic macros. It could be implemented simply with built-in macros. The generic macro for *sqrt()*, for example, could be implemented with:

```
#undef sqrt
#define sqrt(x) __BUILTIN_GENERIC_sqrt(x)
```

Generic macros are designed for a useful level of consistency with C++ overloaded math functions.

The great majority of existing C programs are expected to be unaffected when the <tgmath.h> header is included instead of the <math.h> or <complex.h> headers. Generic macros are similar to the ISO/IEC 9899:1999 standard library masking macros, though the semantic types of return

14173 values differ.

14174 The ability to overload on integer as well as floating types would have been useful for some
 14175 functions; for example, *copysign()*. Overloading with different numbers of arguments would
 14176 have allowed reusing names; for example, *remainder()* for *remquo()*. However, these facilities
 14177 would have complicated the specification; and their natural consistent use, such as for a floating
 14178 *abs()* or a two-argument *atan()*, would have introduced further inconsistencies with the
 14179 ISO/IEC 9899:1999 standard for insufficient benefit.

14180 The ISO C standard in no way limits the implementation's options for efficiency, including
 14181 inlining library functions.

14182 **FUTURE DIRECTIONS**

14183 None.

14184 **SEE ALSO**

14185 [<math.h>](#), [<complex.h>](#)

14186 XSH *cabs()*, *fabs()*, *modf()*

14187 **CHANGE HISTORY**

14188 First released in Issue 6. Included for alignment with the ISO/IEC 9899:1999 standard.

14189 **Issue 7**

14190 Austin Group Interpretation 1003.1-2001 #184 is applied, clarifying the functions for which a
 14191 corresponding type-generic macro exists with the same name as the function.

14192 POSIX.1-2008, Technical Corrigendum 1, XBD/TC1-2008/0075 [357,427] is applied. +

14193 NAME

14194 time.h — time types

14195 SYNOPSIS

14196 #include <time.h>

14197 DESCRIPTION

14198 CX Some of the functionality described on this reference page extends the ISO C standard.
 14199 Applications shall define the appropriate feature test macro (see XSH [Section 2.2](#), on page 472) to
 14200 enable the visibility of these symbols in this header.

14201 The <time.h> header shall define the **clock_t**, **size_t**, **time_t**, types as described in
 14202 <sys/types.h>.

14203 CX The <time.h> header shall define the **clockid_t** and **timer_t** types as described in <sys/types.h>. -

14204 The <time.h> header shall define the **locale_t** type as described in <locale.h>. -

14205 CPT The <time.h> header shall define the **pid_t** type as described in <sys/types.h>.

14206 CX The tag **sigevent** shall be declared as naming an incomplete structure type, the contents of which
 14207 are described in the <signal.h> header.

14208 The <time.h> header shall declare the **tm** structure, which shall include at least the following
 14209 members:

14210	int	tm_sec	Seconds [0,60].
14211	int	tm_min	Minutes [0,59].
14212	int	tm_hour	Hour [0,23].
14213	int	tm_mday	Day of month [1,31].
14214	int	tm_mon	Month of year [0,11].
14215	int	tm_year	Years since 1900.
14216	int	tm_wday	Day of week [0,6] (Sunday =0).
14217	int	tm_yday	Day of year [0,365].
14218	int	tm_isdst	Daylight Savings flag.

14219 The value of *tm_isdst* shall be positive if Daylight Savings Time is in effect, 0 if Daylight Savings
 14220 Time is not in effect, and negative if the information is not available.

14221 CX The <time.h> header shall declare the **timespec** structure, which shall include at least the
 14222 following members:

14223	time_t	tv_sec	Seconds.
14224	long	tv_nsec	Nanoseconds.

14225 The <time.h> header shall also declare the **itimerspec** structure, which shall include at least the
 14226 following members:

14227	struct timespec	it_interval	Timer period.
14228	struct timespec	it_value	Timer expiration.

14229 The <time.h> header shall define the following macros:

14230 NULL As described in <stddef.h>.

14231 CLOCKS_PER_SEC A number used to convert the value returned by the *clock()* function into
 14232 XSI seconds. The value shall be an expression with type **clock_t**. The value of
 14233 CLOCKS_PER_SEC shall be 1 million on XSI-conformant systems.
 14234 However, it may be variable on other systems, and it should not be

14235		assumed that <code>CLOCKS_PER_SEC</code> is a compile-time constant.
14236	CX	The <time.h> header shall define the following symbolic constants. The values shall have a type
14237		that is assignment-compatible with <code>clockid_t</code> .
14238	MON	<code>CLOCK_MONOTONIC</code>
14239		The identifier for the system-wide monotonic clock, which is defined as a
14240		clock measuring real time, whose value cannot be set via <code>clock_settime()</code>
14241		and which cannot have negative clock jumps. The maximum possible
14242		clock jump shall be implementation-defined.
14243	CPT	<code>CLOCK_PROCESS_CPUTIME_ID</code>
14244		The identifier of the CPU-time clock associated with the process making a
14245		<code>clock()</code> or <code>timer*()</code> function call.
14246	CX	<code>CLOCK_REALTIME</code> The identifier of the system-wide clock measuring real time.
14247	TCT	<code>CLOCK_THREAD_CPUTIME_ID</code>
14248		The identifier of the CPU-time clock associated with the thread making a
14249		<code>clock()</code> or <code>timer*()</code> function call.
14250	CX	The <time.h> header shall define the following symbolic constant:
14251		<code>TIMER_ABSTIME</code> Flag indicating time is absolute. For functions taking timer objects, this
14252		refers to the clock associated with the timer.
14253	XSI	The <time.h> header shall provide a declaration or definition for <code>getdate_err</code> . The <code>getdate_err</code>
14254		symbol shall expand to an expression of type <code>int</code> . It is unspecified whether <code>getdate_err</code> is a macro
14255		or an identifier declared with external linkage, and whether or not it is a modifiable lvalue. If a
14256		macro definition is suppressed in order to access an actual object, or a program defines an
14257		identifier with the name <code>getdate_err</code> , the behavior is undefined.
14258		The following shall be declared as functions and may also be defined as macros. Function
14259		prototypes shall be provided.
14260	OB	<code>char *asctime(const struct tm *);</code>
14261	OB CX	<code>char *asctime_r(const struct tm *restrict, char *restrict);</code>
14262		<code>clock_t clock(void);</code>
14263	CPT	<code>int clock_getcpuclockid(pid_t, clockid_t *);</code>
14264	CX	<code>int clock_getres(clockid_t, struct timespec *);</code>
14265		<code>int clock_gettime(clockid_t, struct timespec *);</code>
14266		<code>int clock_nanosleep(clockid_t, int, const struct timespec *,</code>
14267		<code>struct timespec *);</code>
14268		<code>int clock_settime(clockid_t, const struct timespec *);</code>
14269	OB	<code>char *ctime(const time_t *);</code>
14270	OB CX	<code>char *ctime_r(const time_t *, char *);</code>
14271		<code>double difftime(time_t, time_t);</code>
14272	XSI	<code>struct tm *getdate(const char *);</code>
14273		<code>struct tm *gmtime(const time_t *);</code>
14274	CX	<code>struct tm *gmtime_r(const time_t *restrict, struct tm *restrict);</code>
14275		<code>struct tm *localtime(const time_t *);</code>
14276	CX	<code>struct tm *localtime_r(const time_t *restrict, struct tm *restrict);</code>
14277		<code>time_t mktime(struct tm *);</code>
14278	CX	<code>int nanosleep(const struct timespec *, struct timespec *);</code>
14279		<code>size_t strftime(char *restrict, size_t, const char *restrict,</code>
14280		<code>const struct tm *restrict);</code>

```

14281 CX      size_t      strftime_l(char *restrict, size_t, const char *restrict,
14282                const struct tm *restrict, locale_t);
14283 XSI      char        *strptime(const char *restrict, const char *restrict,
14284                struct tm *restrict);
14285          time_t      time(time_t *);
14286 CX      int          timer_create(clockid_t, struct sigevent *restrict,
14287                timer_t *restrict);
14288          int          timer_delete(timer_t);
14289          int          timer_getoverrun(timer_t);
14290          int          timer_gettime(timer_t, struct itimerspec *);
14291          int          timer_settime(timer_t, int, const struct itimerspec *restrict,
14292                struct itimerspec *restrict);
14293          void         tzset(void);

```

14294 The <time.h> header shall declare the following as variables:

```

14295 XSI      extern int     daylight;
14296          extern long    timezone;
14297 CX      extern char     *tzname[];

```

14298 CX Inclusion of the <time.h> header may make visible all symbols from the <signal.h> header.

14299 APPLICATION USAGE

14300 The range [0,60] for *tm_sec* allows for the occasional leap second.

14301 *tm_year* is a signed value; therefore, years before 1900 may be represented.

14302 To obtain the number of clock ticks per second returned by the *times()* function, applications
14303 should call *sysconf(_SC_CLK_TCK)*.

14304 RATIONALE

14305 The range [0,60] seconds allows for positive or negative leap seconds. The formal definition of
14306 UTC does not permit double leap seconds, so all mention of double leap seconds has been
14307 removed, and the range shortened from the former [0,61] seconds seen in earlier versions of this
14308 standard.

14309 FUTURE DIRECTIONS

14310 None.

14311 SEE ALSO

14312 [<locale.h>](#), [<signal.h>](#), [<stddef.h>](#), [<sys/types.h>](#)

14313 XSH Section 2.2 (on page 472), *asctime()*, *clock()*, *clock_getcpuclockid()*, *clock_getres()*,
14314 *clock_nanosleep()*, *ctime()*, *difftime()*, *getdate()*, *gmtime()*, *localtime()*, *mktime()*, *mq_receive()*,
14315 *mq_send()*, *nanosleep()*, *pthread_getcpuclockid()*, *pthread_mutex_timedlock()*,
14316 *pthread_rwlock_timedrdlock()*, *pthread_rwlock_timedwrlock()*, *sem_timedwait()*, *strftime()*, *strptime()*,
14317 *sysconf()*, *time()*, *timer_create()*, *timer_delete()*, *timer_getoverrun()*, *tzset()*, *utime()*

14318 CHANGE HISTORY

14319 First released in Issue 1. Derived from Issue 1 of the SVID.

14320 Issue 5

14321 The DESCRIPTION is updated for alignment with the POSIX Realtime Extension and the POSIX
14322 Threads Extension.

Issue 6

The Open Group Corrigendum U035/6 is applied. In the DESCRIPTION, the types **clockid_t** and **timer_t** have been described.

The following changes are made for alignment with the ISO POSIX-1: 1996 standard:

- The POSIX timer-related functions are marked as part of the Timers option.

The symbolic name CLK_TCK is removed. Application usage is added describing how its equivalent functionality can be obtained using *sysconf()*.

The *clock_getcpuclockid()* function and manifest constants CLOCK_PROCESS_CPUTIME_ID and CLOCK_THREAD_CPUTIME_ID are added for alignment with IEEE Std 1003.1d-1999.

The manifest constant CLOCK_MONOTONIC and the *clock_nanosleep()* function are added for alignment with IEEE Std 1003.1j-2000.

The following changes are made for alignment with the ISO/IEC 9899: 1999 standard:

- The range for seconds is changed from [0,61] to [0,60].
- The **restrict** keyword is added to the prototypes for *asctime_r()*, *gmtime_r()*, *localtime_r()*, *strftime()*, *strptime()*, *timer_create()*, and *timer_settime()*.

IEEE PASC Interpretation 1003.1 #84 is applied adding the statement that symbols from the <signal.h> header may be made visible when the <time.h> header is included.

Extensions beyond the ISO C standard are marked.

Issue 7

Austin Group Interpretation 1003.1-2001 #111 is applied.

SD5-XBD-ERN-74 is applied.

The *strptime_l()* function is added from The Open Group Technical Standard, 2006, Extended API Set Part 4.

Functionality relating to the Timers option is moved to the Base.

This reference page is clarified with respect to macros and symbolic constants, and declarations for the **locale_t** and **pid_t** types and the **sigevent** structure are added.

The description of the *getdate_err* value is expanded.

POSIX.1-2008, Technical Corrigendum 1, XBD/TC1-2008/0076 [212] and XBD/TC1-2008/0077 + [212] are applied.

14352 **NAME**

14353 trace.h — tracing

14354 **SYNOPSIS**

14355 OB TRC #include <trace.h>

14356 **DESCRIPTION**

14357 The <trace.h> header shall define the **posix_trace_event_info** structure, which shall include at
 14358 least the following members:

```

14359       trace_event_id_t   posix_event_id
14360       pid_t              posix_pid
14361       void               *posix_prog_address
14362       pthread_t          posix_thread_id
14363       struct timespec    posix_timestamp
14364       int                posix_truncation_status

```

14365 The <trace.h> header shall define the **posix_trace_status_info** structure, which shall include at
 14366 least the following members:

```

14367       int        posix_stream_full_status
14368       int        posix_stream_overrun_status
14369       int        posix_stream_status
14370 OB TRL   int        posix_log_full_status
14371       int        posix_log_overrun_status
14372       int        posix_stream_flush_error
14373       int        posix_stream_flush_status

```

14374 The <trace.h> header shall define the following symbolic constants:

```

14375       POSIX_TRACE_ALL_EVENTS
14376 OB TRL   POSIX_TRACE_APPEND
14377 OB TRI   POSIX_TRACE_CLOSE_FOR_CHILD
14378 OB TEF   POSIX_TRACE_FILTER
14379 OB TRL   POSIX_TRACE_FLUSH
14380       POSIX_TRACE_FLUSH_START
14381       POSIX_TRACE_FLUSH_STOP
14382       POSIX_TRACE_FLUSHING
14383       POSIX_TRACE_FULL
14384       POSIX_TRACE_LOOP
14385       POSIX_TRACE_NO_OVERRUN
14386 OB TRL   POSIX_TRACE_NOT_FLUSHING
14387       POSIX_TRACE_NOT_FULL
14388 OB TRI   POSIX_TRACE_INHERITED
14389       POSIX_TRACE_NOT_TRUNCATED
14390       POSIX_TRACE_OVERFLOW
14391       POSIX_TRACE_OVERRUN
14392       POSIX_TRACE_RESUME
14393       POSIX_TRACE_RUNNING
14394       POSIX_TRACE_START
14395       POSIX_TRACE_STOP
14396       POSIX_TRACE_SUSPENDED
14397       POSIX_TRACE_SYSTEM_EVENTS
14398       POSIX_TRACE_TRUNCATED_READ

```

```

14399     POSIX_TRACE_TRUNCATED_RECORD
14400     POSIX_TRACE_UNNAMED_USER_EVENT
14401     POSIX_TRACE_UNTIL_FULL
14402     POSIX_TRACE_WOPID_EVENTS

14403 OB TEF The <trace.h> header shall define the size_t, trace_attr_t, trace_event_id_t, trace_event_set_t,
14404 and trace_id_t types as described in <sys/types.h>.

14405 The following shall be declared as functions and may also be defined as macros. Function
14406 prototypes shall be provided.

14407     int  posix_trace_attr_destroy(trace_attr_t *);
14408     int  posix_trace_attr_getclockres(const trace_attr_t *,
14409         struct timespec *);
14410     int  posix_trace_attr_getcreatetime(const trace_attr_t *,
14411         struct timespec *);
14412     int  posix_trace_attr_getgenversion(const trace_attr_t *, char *);
14413 TRI     int  posix_trace_attr_getinherited(const trace_attr_t *restrict,
14414         int *restrict);
14415 TRL     int  posix_trace_attr_getlogfullpolicy(const trace_attr_t *restrict,
14416         int *restrict);
14417     int  posix_trace_attr_getlogsize(const trace_attr_t *restrict,
14418         size_t *restrict);
14419     int  posix_trace_attr_getmaxdatasize(const trace_attr_t *restrict,
14420         size_t *restrict);
14421     int  posix_trace_attr_getmaxsystemeventsize(const trace_attr_t *restrict,
14422         size_t *restrict);
14423     int  posix_trace_attr_getmaxusereventsize(const trace_attr_t *restrict,
14424         size_t, size_t *restrict);
14425     int  posix_trace_attr_getname(const trace_attr_t *, char *);
14426     int  posix_trace_attr_getstreamfullpolicy(const trace_attr_t *restrict,
14427         int *restrict);
14428     int  posix_trace_attr_getstreamsize(const trace_attr_t *restrict,
14429         size_t *restrict);
14430     int  posix_trace_attr_init(trace_attr_t *);
14431 TRI     int  posix_trace_attr_setinherited(trace_attr_t *, int);
14432 TRL     int  posix_trace_attr_setlogfullpolicy(trace_attr_t *, int);
14433     int  posix_trace_attr_setlogsize(trace_attr_t *, size_t);
14434     int  posix_trace_attr_setmaxdatasize(trace_attr_t *, size_t);
14435     int  posix_trace_attr_setname(trace_attr_t *, const char *);
14436     int  posix_trace_attr_setstreamfullpolicy(trace_attr_t *, int);
14437     int  posix_trace_attr_setstreamsize(trace_attr_t *, size_t);
14438     int  posix_trace_clear(trace_id_t);
14439 TRL     int  posix_trace_close(trace_id_t);
14440     int  posix_trace_create(pid_t, const trace_attr_t *restrict,
14441         trace_id_t *restrict);
14442 TRL     int  posix_trace_create_withlog(pid_t, const trace_attr_t *restrict,
14443         int, trace_id_t *restrict);
14444     void posix_trace_event(trace_event_id_t, const void *restrict, size_t);
14445     int  posix_trace_eventid_equal(trace_id_t, trace_event_id_t,
14446         trace_event_id_t);
14447     int  posix_trace_eventid_get_name(trace_id_t, trace_event_id_t, char *);
14448     int  posix_trace_eventid_open(const char *restrict,

```

```

14449         trace_event_id_t *restrict);
14450 TEF    int  posix_trace_eventset_add(trace_event_id_t, trace_event_set_t *);
14451       int  posix_trace_eventset_del(trace_event_id_t, trace_event_set_t *);
14452       int  posix_trace_eventset_empty(trace_event_set_t *);
14453       int  posix_trace_eventset_fill(trace_event_set_t *, int);
14454       int  posix_trace_eventset_ismember(trace_event_id_t,
14455         const trace_event_set_t *restrict, int *restrict);
14456       int  posix_trace_eventtypelist_getnext_id(trace_id_t,
14457         trace_event_id_t *restrict, int *restrict);
14458       int  posix_trace_eventtypelist_rewind(trace_id_t);
14459 TRL    int  posix_trace_flush(trace_id_t);
14460       int  posix_trace_get_attr(trace_id_t, trace_attr_t *);
14461 TEF    int  posix_trace_get_filter(trace_id_t, trace_event_set_t *);
14462       int  posix_trace_get_status(trace_id_t,
14463         struct posix_trace_status_info *);
14464       int  posix_trace_getnext_event(trace_id_t,
14465         struct posix_trace_event_info *restrict, void *restrict,
14466         size_t, size_t *restrict, int *restrict);
14467 TRL    int  posix_trace_open(int, trace_id_t *);
14468       int  posix_trace_rewind(trace_id_t);
14469 TEF    int  posix_trace_set_filter(trace_id_t, const trace_event_set_t *, int);
14470       int  posix_trace_shutdown(trace_id_t);
14471       int  posix_trace_start(trace_id_t);
14472       int  posix_trace_stop(trace_id_t);
14473       int  posix_trace_timedgetnext_event(trace_id_t,
14474         struct posix_trace_event_info *restrict, void *restrict,
14475         size_t, size_t *restrict, int *restrict,
14476         const struct timespec *restrict);
14477 TEF    int  posix_trace_trid_eventid_open(trace_id_t, const char *restrict,
14478         trace_event_id_t *restrict);
14479       int  posix_trace_trygetnext_event(trace_id_t,
14480         struct posix_trace_event_info *restrict, void *restrict, size_t,
14481         size_t *restrict, int *restrict);

```

APPLICATION USAGE

None.

RATIONALE

None.

FUTURE DIRECTIONS

The <trace.h> header may be removed in a future version.

SEE ALSO

[<sys/types.h>](#)

XSH Section 2.11 (on page 534), *posix_trace_attr_destroy()*, *posix_trace_attr_getclockres()*, *posix_trace_attr_getinherited()*, *posix_trace_attr_getlogsize()*, *posix_trace_clear()*, *posix_trace_close()*, *posix_trace_create()*, *posix_trace_event()*, *posix_trace_eventid_equal()*, *posix_trace_eventset_add()*, *posix_trace_eventtypelist_getnext_id()*, *posix_trace_get_attr()*, *posix_trace_get_filter()*, *posix_trace_getnext_event()*, *posix_trace_start()*

14495 **CHANGE HISTORY**

14496 First released in Issue 6. Derived from IEEE Std 1003.1q-2000.

14497 IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/40 is applied, adding the TRL margin
14498 code to the *posix_trace_flush()* function, for alignment with the System Interfaces volume of
14499 POSIX.1-2008.

14500 **Issue 7**

14501 SD5-XBD-ERN-56 is applied, adding a reference to <sys/types.h> for the **size_t** type.

14502 The <trace.h> header is marked obsolescent.

14503 This reference page is clarified with respect to macros and symbolic constants.

14504 NAME

14505 ulimit.h — ulimit commands

14506 SYNOPSIS

14507 OB XSI #include <ulimit.h>

14508 DESCRIPTION

14509 The **<ulimit.h>** header shall define the symbolic constants used by the *ulimit()* function.

14510 Symbolic constants:

14511 UL_GETFSIZE Get maximum file size.

14512 UL_SETFSIZE Set maximum file size.

14513 The following shall be declared as a function and may also be defined as a macro. A function
14514 prototype shall be provided.

14515 long ulimit(int, ...);

14516 APPLICATION USAGE

14517 See *ulimit()*.

14518 RATIONALE

14519 None.

14520 FUTURE DIRECTIONS

14521 None.

14522 SEE ALSO

14523 XSH *ulimit()*

14524 CHANGE HISTORY

14525 First released in Issue 3.

14526 Issue 7

14527 The **<ulimit.h>** header is marked obsolescent.

14528 **NAME**

14529 unistd.h — standard symbolic constants and types

14530 **SYNOPSIS**

14531 #include <unistd.h>

14532 **DESCRIPTION**

14533 The <unistd.h> header defines miscellaneous symbolic constants and types, and declares
14534 miscellaneous functions. The actual values of the constants are unspecified except as shown. The
14535 contents of this header are shown below.

14536 **Version Test Macros**

14537 The <unistd.h> header shall define the following symbolic constants. The values shall be
14538 suitable for use in #if preprocessing directives.

14539 _POSIX_VERSION

14540 Integer value indicating version of this standard (C-language binding) to which the
14541 implementation conforms. For implementations conforming to POSIX.1-2008, the value
14542 shall be 200809L.

14543 _POSIX2_VERSION

14544 Integer value indicating version of the Shell and Utilities volume of POSIX.1 to which the
14545 implementation conforms. For implementations conforming to POSIX.1-2008, the value
14546 shall be 200809L. For profile implementations that define _POSIX_SUBPROFILE (see
14547 Section 2.1.5.1) in <unistd.h>, POSIX2_VERSION may be left undefined or be defined with
14548 the value -1 to indicate that the Shell and Utilities volume of POSIX.1 is not supported. In
14549 this case, a call to sysconf(_SC_2_VERSION) shall return either 200809L or -1 indicating that
14550 the Shell and Utilities volume of POSIX.1 is or is not, respectively, supported at runtime.

14551 The <unistd.h> header shall define the following symbolic constant only if the implementation
14552 supports the XSI option; see Section 2.1.4 (on page 19). If defined, its value shall be suitable for
14553 use in #if preprocessing directives.

14554 XSI _XOPEN_VERSION

14555 Integer value indicating version of the X/Open Portability Guide to which the
14556 implementation conforms. The value shall be 700.

14557 **Constants for Options and Option Groups**

14558 The following symbolic constants, if defined in <unistd.h>, shall have a value of -1, 0, or
14559 greater, unless otherwise specified below. For profile implementations that define
14560 _POSIX_SUBPROFILE (see Section 2.1.5.1) in <unistd.h>, constants described below as always
14561 having a value greater than zero need not be defined and, if defined, may have a value of -1, 0,
14562 or greater. The values shall be suitable for use in #if preprocessing directives.

14563 If a symbolic constant is not defined or is defined with the value -1, the option is not supported
14564 for compilation. If it is defined with a value greater than zero, the option shall always be
14565 supported when the application is executed. If it is defined with the value zero, the option shall
14566 be supported for compilation and might or might not be supported at runtime. See Section 2.1.6
14567 (on page 26) for further information about the conformance requirements of these three
14568 categories of support.

14569 ADV _POSIX_ADVISORY_INFO

14570 The implementation supports the Advisory Information option. If this symbol is defined in
14571 <unistd.h>, it shall be defined to be -1, 0, or 200809L. The value of this symbol reported by
14572 sysconf() shall either be -1 or 200809L.

14573		_POSIX_ASYNCHRONOUS_IO
14574		The implementation supports asynchronous input and output. This symbol shall always be
14575		set to the value 200809L.
14576		_POSIX_BARRIERS
14577		The implementation supports barriers. This symbol shall always be set to the value
14578		200809L.
14579		_POSIX_CHOWN_RESTRICTED
14580		The use of <i>chown()</i> and <i>fchown()</i> is restricted to a process with appropriate privileges, and
14581		to changing the group ID of a file only to the effective group ID of the process or to one of
14582		its supplementary group IDs. This symbol shall be defined with a value other than -1.
14583		_POSIX_CLOCK_SELECTION
14584		The implementation supports clock selection. This symbol shall always be set to the value
14585		200809L.
14586	CPT	_POSIX_CPUTIME
14587		The implementation supports the Process CPU-Time Clocks option. If this symbol is
14588		defined in <unistd.h>, it shall be defined to be -1, 0, or 200809L. The value of this symbol
14589		reported by <i>sysconf()</i> shall either be -1 or 200809L.
14590	FSC	_POSIX_FSYNC
14591		The implementation supports the File Synchronization option. If this symbol is defined in
14592		<unistd.h>, it shall be defined to be -1, 0, or 200809L. The value of this symbol reported by
14593		<i>sysconf()</i> shall either be -1 or 200809L.
14594		_POSIX_IPV6
14595		The implementation supports the IPv6 option. If this symbol is defined in <unistd.h>, it
14596		shall be defined to be -1, 0, or 200809L. The value of this symbol reported by <i>sysconf()</i> shall
14597		either be -1 or 200809L.
14598		_POSIX_JOB_CONTROL
14599		The implementation supports job control. This symbol shall always be set to a value greater
14600		than zero.
14601		_POSIX_MAPPED_FILES
14602		The implementation supports memory mapped Files. This symbol shall always be set to the
14603		value 200809L.
14604	ML	_POSIX_MEMLOCK
14605		The implementation supports the Process Memory Locking option. If this symbol is defined
14606		in <unistd.h>, it shall be defined to be -1, 0, or 200809L. The value of this symbol reported
14607		by <i>sysconf()</i> shall either be -1 or 200809L.
14608	MLR	_POSIX_MEMLOCK_RANGE
14609		The implementation supports the Range Memory Locking option. If this symbol is defined
14610		in <unistd.h>, it shall be defined to be -1, 0, or 200809L. The value of this symbol reported
14611		by <i>sysconf()</i> shall either be -1 or 200809L.
14612		_POSIX_MEMORY_PROTECTION
14613		The implementation supports memory protection. This symbol shall always be set to the
14614		value 200809L.
14615	MSG	_POSIX_MESSAGE_PASSING
14616		The implementation supports the Message Passing option. If this symbol is defined in
14617		<unistd.h>, it shall be defined to be -1, 0, or 200809L. The value of this symbol reported by
14618		<i>sysconf()</i> shall either be -1 or 200809L.

14619	MON	_POSIX_MONOTONIC_CLOCK
14620		The implementation supports the Monotonic Clock option. If this symbol is defined in
14621		<unistd.h>, it shall be defined to be -1, 0, or 200809L. The value of this symbol reported by
14622		<i>sysconf()</i> shall either be -1 or 200809L.
14623		_POSIX_NO_TRUNC
14624		Pathname components longer than {NAME_MAX} generate an error. This symbol shall be
14625		defined with a value other than -1.
14626	PIO	_POSIX_PRIORITIZED_IO
14627		The implementation supports the Prioritized Input and Output option. If this symbol is
14628		defined in <unistd.h>, it shall be defined to be -1, 0, or 200809L. The value of this symbol
14629		reported by <i>sysconf()</i> shall either be -1 or 200809L.
14630	PS	_POSIX_PRIORITY_SCHEDULING
14631		The implementation supports the Process Scheduling option. If this symbol is defined in
14632		<unistd.h>, it shall be defined to be -1, 0, or 200809L. The value of this symbol reported by
14633		<i>sysconf()</i> shall either be -1 or 200809L.
14634	RS	_POSIX_RAW_SOCKETS
14635		The implementation supports the Raw Sockets option. If this symbol is defined in
14636		<unistd.h>, it shall be defined to be -1, 0, or 200809L. The value of this symbol reported by
14637		<i>sysconf()</i> shall either be -1 or 200809L.
14638		_POSIX_READER_WRITER_LOCKS
14639		The implementation supports read-write locks. This symbol shall always be set to the value
14640		200809L.
14641		_POSIX_REALTIME_SIGNALS
14642		The implementation supports realtime signals. This symbol shall always be set to the value
14643		200809L.
14644		_POSIX_REGEX
14645		The implementation supports the Regular Expression Handling option. This symbol shall
14646		always be set to a value greater than zero.
14647		_POSIX_SAVED_IDS
14648		Each process has a saved set-user-ID and a saved set-group-ID. This symbol shall always be
14649		set to a value greater than zero.
14650		_POSIX_SEMAPHORES
14651		The implementation supports semaphores. This symbol shall always be set to the value
14652		200809L.
14653	SHM	_POSIX_SHARED_MEMORY_OBJECTS
14654		The implementation supports the Shared Memory Objects option. If this symbol is defined
14655		in <unistd.h>, it shall be defined to be -1, 0, or 200809L. The value of this symbol reported
14656		by <i>sysconf()</i> shall either be -1 or 200809L.
14657		_POSIX_SHELL
14658		The implementation supports the POSIX shell. This symbol shall always be set to a value
14659		greater than zero.
14660	SPN	_POSIX_SPAWN
14661		The implementation supports the Spawn option. If this symbol is defined in <unistd.h>, it
14662		shall be defined to be -1, 0, or 200809L. The value of this symbol reported by <i>sysconf()</i> shall
14663		either be -1 or 200809L.

14664		_POSIX_SPIN_LOCKS
14665		The implementation supports spin locks. This symbol shall always be set to the value
14666		200809L.
14667	SS	_POSIX_SPORADIC_SERVER
14668		The implementation supports the Process Sporadic Server option. If this symbol is defined
14669		in <unistd.h>, it shall be defined to be -1, 0, or 200809L. The value of this symbol reported
14670		by <i>sysconf()</i> shall either be -1 or 200809L.
14671	SIO	_POSIX_SYNCHRONIZED_IO
14672		The implementation supports the Synchronized Input and Output option. If this symbol is
14673		defined in <unistd.h>, it shall be defined to be -1, 0, or 200809L. The value of this symbol
14674		reported by <i>sysconf()</i> shall either be -1 or 200809L.
14675	TSA	_POSIX_THREAD_ATTR_STACKADDR
14676		The implementation supports the Thread Stack Address Attribute option. If this symbol is
14677		defined in <unistd.h>, it shall be defined to be -1, 0, or 200809L. The value of this symbol
14678		reported by <i>sysconf()</i> shall either be -1 or 200809L.
14679	TSS	_POSIX_THREAD_ATTR_STACKSIZE
14680		The implementation supports the Thread Stack Size Attribute option. If this symbol is
14681		defined in <unistd.h>, it shall be defined to be -1, 0, or 200809L. The value of this symbol
14682		reported by <i>sysconf()</i> shall either be -1 or 200809L.
14683	TCT	_POSIX_THREAD_CPU_TIME
14684		The implementation supports the Thread CPU-Time Clocks option. If this symbol is
14685		defined in <unistd.h>, it shall be defined to be -1, 0, or 200809L. The value of this symbol
14686		reported by <i>sysconf()</i> shall either be -1 or 200809L.
14687	TPI	_POSIX_THREAD_PRIO_INHERIT
14688		The implementation supports the Non-Robust Mutex Priority Inheritance option. If this
14689		symbol is defined in <unistd.h>, it shall be defined to be -1, 0, or 200809L. The value of this
14690		symbol reported by <i>sysconf()</i> shall either be -1 or 200809L.
14691	TPP	_POSIX_THREAD_PRIO_PROTECT
14692		The implementation supports the Non-Robust Mutex Priority Protection option. If this
14693		symbol is defined in <unistd.h>, it shall be defined to be -1, 0, or 200809L. The value of this
14694		symbol reported by <i>sysconf()</i> shall either be -1 or 200809L.
14695	TPS	_POSIX_THREAD_PRIORITY_SCHEDULING
14696		The implementation supports the Thread Execution Scheduling option. If this symbol is
14697		defined in <unistd.h>, it shall be defined to be -1, 0, or 200809L. The value of this symbol
14698		reported by <i>sysconf()</i> shall either be -1 or 200809L.
14699	TSH	_POSIX_THREAD_PROCESS_SHARED
14700		The implementation supports the Thread Process-Shared Synchronization option. If this
14701		symbol is defined in <unistd.h>, it shall be defined to be -1, 0, or 200809L. The value of this
14702		symbol reported by <i>sysconf()</i> shall either be -1 or 200809L.
14703	RPI	_POSIX_THREAD_ROBUST_PRIO_INHERIT
14704		The implementation supports the Robust Mutex Priority Inheritance option. If this symbol
14705		is defined in <unistd.h>, it shall be defined to be -1, 0, or 200809L. The value of this symbol
14706		reported by <i>sysconf()</i> shall either be -1 or 200809L.
14707	RPP	_POSIX_THREAD_ROBUST_PRIO_PROTECT
14708		The implementation supports the Robust Mutex Priority Protection option. If this symbol is
14709		defined in <unistd.h>, it shall be defined to be -1, 0, or 200809L. The value of this symbol

14710		reported by <i>sysconf()</i> shall either be <code>-1</code> or <code>200809L</code> .
14711		<code>_POSIX_THREAD_SAFE_FUNCTIONS</code>
14712		The implementation supports thread-safe functions. This symbol shall always be set to the
14713		value <code>200809L</code> .
14714	TSP	<code>_POSIX_THREAD_SPORADIC_SERVER</code>
14715		The implementation supports the Thread Sporadic Server option. If this symbol is defined
14716		in <code><unistd.h></code> , it shall be defined to be <code>-1</code> , <code>0</code> , or <code>200809L</code> . The value of this symbol reported
14717		by <i>sysconf()</i> shall either be <code>-1</code> or <code>200809L</code> .
14718		<code>_POSIX_THREADS</code>
14719		The implementation supports threads. This symbol shall always be set to the value
14720		<code>200809L</code> .
14721		<code>_POSIX_TIMEOUTS</code>
14722		The implementation supports timeouts. This symbol shall always be set to the value
14723		<code>200809L</code> .
14724		<code>_POSIX_TIMERS</code>
14725		The implementation supports timers. This symbol shall always be set to the value <code>200809L</code> .
14726	OB TRC	<code>_POSIX_TRACE</code>
14727		The implementation supports the Trace option. If this symbol is defined in <code><unistd.h></code> , it
14728		shall be defined to be <code>-1</code> , <code>0</code> , or <code>200809L</code> . The value of this symbol reported by <i>sysconf()</i> shall
14729		either be <code>-1</code> or <code>200809L</code> .
14730	OB TEF	<code>_POSIX_TRACE_EVENT_FILTER</code>
14731		The implementation supports the Trace Event Filter option. If this symbol is defined in
14732		<code><unistd.h></code> , it shall be defined to be <code>-1</code> , <code>0</code> , or <code>200809L</code> . The value of this symbol reported by
14733		<i>sysconf()</i> shall either be <code>-1</code> or <code>200809L</code> .
14734	OB TRI	<code>_POSIX_TRACE_INHERIT</code>
14735		The implementation supports the Trace Inherit option. If this symbol is defined in
14736		<code><unistd.h></code> , it shall be defined to be <code>-1</code> , <code>0</code> , or <code>200809L</code> . The value of this symbol reported by
14737		<i>sysconf()</i> shall either be <code>-1</code> or <code>200809L</code> .
14738	OB TRL	<code>_POSIX_TRACE_LOG</code>
14739		The implementation supports the Trace Log option. If this symbol is defined in <code><unistd.h></code> ,
14740		it shall be defined to be <code>-1</code> , <code>0</code> , or <code>200809L</code> . The value of this symbol reported by <i>sysconf()</i>
14741		shall either be <code>-1</code> or <code>200809L</code> .
14742	TYM	<code>_POSIX_TYPED_MEMORY_OBJECTS</code>
14743		The implementation supports the Typed Memory Objects option. If this symbol is defined
14744		in <code><unistd.h></code> , it shall be defined to be <code>-1</code> , <code>0</code> , or <code>200809L</code> . The value of this symbol reported
14745		by <i>sysconf()</i> shall either be <code>-1</code> or <code>200809L</code> .
14746	OB	<code>_POSIX_V6_ILP32_OFF32</code>
14747		The implementation provides a C-language compilation environment with 32-bit int , long ,
14748		pointer , and off_t types.
14749	OB	<code>_POSIX_V6_ILP32_OFFBIG</code>
14750		The implementation provides a C-language compilation environment with 32-bit int , long ,
14751		and pointer types and an off_t type using at least 64 bits.
14752	OB	<code>_POSIX_V6_LP64_OFF64</code>
14753		The implementation provides a C-language compilation environment with 32-bit int and
14754		64-bit long , pointer , and off_t types.

14755	OB	_POSIX_V6_LPBIG_OFFBIG
14756		The implementation provides a C-language compilation environment with an int type
14757		using at least 32 bits and long , pointer , and off_t types using at least 64 bits.
14758		_POSIX_V7_ILP32_OFF32
14759		The implementation provides a C-language compilation environment with 32-bit int , long ,
14760		pointer , and off_t types.
14761		_POSIX_V7_ILP32_OFFBIG
14762		The implementation provides a C-language compilation environment with 32-bit int , long ,
14763		and pointer types and an off_t type using at least 64 bits.
14764		_POSIX_V7_LP64_OFF64
14765		The implementation provides a C-language compilation environment with 32-bit int and
14766		64-bit long , pointer , and off_t types.
14767		_POSIX_V7_LPBIG_OFFBIG
14768		The implementation provides a C-language compilation environment with an int type
14769		using at least 32 bits and long , pointer , and off_t types using at least 64 bits.
14770		_POSIX2_C_BIND
14771		The implementation supports the C-Language Binding option. This symbol shall always
14772		have the value 200809L.
14773	CD	_POSIX2_C_DEV
14774		The implementation supports the C-Language Development Utilities option. If this symbol
14775		is defined in <unistd.h>, it shall be defined to be -1, 0, or 200809L. The value of this symbol
14776		reported by <i>sysconf()</i> shall either be -1 or 200809L.
14777		_POSIX2_CHAR_TERM
14778		The implementation supports the Terminal Characteristics option. The value of this symbol
14779		reported by <i>sysconf()</i> shall either be -1 or a value greater than zero.
14780	FD	_POSIX2_FORT_DEV
14781		The implementation supports the FORTRAN Development Utilities option. If this symbol
14782		is defined in <unistd.h>, it shall be defined to be -1, 0, or 200809L. The value of this symbol
14783		reported by <i>sysconf()</i> shall either be -1 or 200809L.
14784	FR	_POSIX2_FORT_RUN
14785		The implementation supports the FORTRAN Runtime Utilities option. If this symbol is
14786		defined in <unistd.h>, it shall be defined to be -1, 0, or 200809L. The value of this symbol
14787		reported by <i>sysconf()</i> shall either be -1 or 200809L.
14788		_POSIX2_LOCALEDEF
14789		The implementation supports the creation of locales by the <i>localedef</i> utility. If this symbol is
14790		defined in <unistd.h>, it shall be defined to be -1, 0, or 200809L. The value of this symbol
14791		reported by <i>sysconf()</i> shall either be -1 or 200809L.
14792	OB BE	_POSIX2_PBS
14793		The implementation supports the Batch Environment Services and Utilities option. If this
14794		symbol is defined in <unistd.h>, it shall be defined to be -1, 0, or 200809L. The value of this
14795		symbol reported by <i>sysconf()</i> shall either be -1 or 200809L.
14796	OB BE	_POSIX2_PBS_ACCOUNTING
14797		The implementation supports the Batch Accounting option. If this symbol is defined in
14798		<unistd.h>, it shall be defined to be -1, 0, or 200809L. The value of this symbol reported by
14799		<i>sysconf()</i> shall either be -1 or 200809L.

14800	OB BE	_POSIX2_PBS_CHECKPOINT	
14801			The implementation supports the Batch Checkpoint/Restart option. If this symbol is
14802			defined in <unistd.h>, it shall be defined to be -1, 0, or 200809L. The value of this symbol
14803			reported by <i>sysconf()</i> shall either be -1 or 200809L.
14804	OB BE	_POSIX2_PBS_LOCATE	
14805			The implementation supports the Locate Batch Job Request option. If this symbol is defined
14806			in <unistd.h>, it shall be defined to be -1, 0, or 200809L. The value of this symbol reported
14807			by <i>sysconf()</i> shall either be -1 or 200809L.
14808	OB BE	_POSIX2_PBS_MESSAGE	
14809			The implementation supports the Batch Job Message Request option. If this symbol is
14810			defined in <unistd.h>, it shall be defined to be -1, 0, or 200809L. The value of this symbol
14811			reported by <i>sysconf()</i> shall either be -1 or 200809L.
14812	OB BE	_POSIX2_PBS_TRACK	
14813			The implementation supports the Track Batch Job Request option. If this symbol is defined
14814			in <unistd.h>, it shall be defined to be -1, 0, or 200809L. The value of this symbol reported
14815			by <i>sysconf()</i> shall either be -1 or 200809L.
14816	SD	_POSIX2_SW_DEV	
14817			The implementation supports the Software Development Utilities option. If this symbol is
14818			defined in <unistd.h>, it shall be defined to be -1, 0, or 200809L. The value of this symbol
14819			reported by <i>sysconf()</i> shall either be -1 or 200809L.
14820	UP	_POSIX2_UPE	
14821			The implementation supports the User Portability Utilities option. If this symbol is defined
14822			in <unistd.h>, it shall be defined to be -1, 0, or 200809L. The value of this symbol reported
14823			by <i>sysconf()</i> shall either be -1 or 200809L.
14824	XSI	_XOPEN_CRYPT	
14825			The implementation supports the X/Open Encryption Option Group.
14826		_XOPEN_ENH_I18N	
14827			The implementation supports the Issue 4, Version 2 Enhanced Internationalization Option
14828			Group. This symbol shall always be set to a value other than -1.
14829		_XOPEN_REALTIME	
14830			The implementation supports the X/Open Realtime Option Group.
14831		_XOPEN_REALTIME_THREADS	
14832			The implementation supports the X/Open Realtime Threads Option Group.
14833		_XOPEN_SHM	
14834			The implementation supports the Issue 4, Version 2 Shared Memory Option Group. This
14835			symbol shall always be set to a value other than -1.
14836	OB XSR	_XOPEN_STREAMS	
14837			The implementation supports the XSI STREAMS Option Group.
14838	XSI	_XOPEN_UNIX	
14839			The implementation supports the XSI option.
14840	UU	_XOPEN_UUCP	
14841			The implementation supports the UUCP Utilities option. If this symbol is defined in
14842			<unistd.h>, it shall be defined to be -1, 0, or 200809L. The value of this symbol reported by
14843			<i>sysconf()</i> shall be either -1 or 200809L.

Execution-Time Symbolic Constants

If any of the following symbolic constants are not defined in the **<unistd.h>** header, the value shall vary depending on the file to which it is applied. If defined, they shall have values suitable for use in **#if** preprocessing directives.

If any of the following symbolic constants are defined to have value **-1** in the **<unistd.h>** header, the implementation shall not provide the option on any file; if any are defined to have a value other than **-1** in the **<unistd.h>** header, the implementation shall provide the option on all applicable files.

All of the following values, whether defined as symbolic constants in **<unistd.h>** or not, may be queried with respect to a specific file using the *pathconf()* or *fpathconf()* functions:

_POSIX_ASYNC_IO

Asynchronous input or output operations may be performed for the associated file.

_POSIX_PRIO_IO

Prioritized input or output operations may be performed for the associated file.

_POSIX_SYNC_IO

Synchronized input or output operations may be performed for the associated file.

If the following symbolic constants are defined in the **<unistd.h>** header, they apply to files and all paths in all file systems on the implementation:

_POSIX_TIMESTAMP_RESOLUTION

The resolution in nanoseconds for all file timestamps.

_POSIX2_SYMLINKS

Symbolic links can be created.

Constants for Functions

The **<unistd.h>** header shall define **NULL** as described in **<stddef.h>**.

The **<unistd.h>** header shall define the following symbolic constants for use with the *access()* function. The values shall be suitable for use in **#if** preprocessing directives.

F_OK Test for existence of file.

R_OK Test for read permission.

W_OK Test for write permission.

X_OK Test for execute (search) permission.

The constants **F_OK**, **R_OK**, **W_OK**, and **X_OK** and the expressions **R_OK | W_OK**, **R_OK | X_OK**, and **R_OK | W_OK | X_OK** shall all have distinct values.

The **<unistd.h>** header shall define the following symbolic constants for the *confstr()* function:

_CS_PATH

This is the value for the *PATH* environment variable that finds all standard utilities.

_CS_POSIX_V7_ILP32_OFF32_CFLAGS

If *sysconf(SC_V7_ILP32_OFF32)* returns **-1**, the meaning of this value is unspecified. Otherwise, this value is the set of initial options to be given to the *c99* utility to build an application using a programming model with 32-bit **int**, **long**, **pointer**, and **off_t** types.

14883 `_CS_POSIX_V7_ILP32_OFF32_LDFLAGS`
 14884 If `sysconf(_SC_V7_ILP32_OFF32)` returns `-1`, the meaning of this value is unspecified.
 14885 Otherwise, this value is the set of final options to be given to the `c99` utility to build an
 14886 application using a programming model with 32-bit **int**, **long**, **pointer**, and **off_t** types.

14887 `_CS_POSIX_V7_ILP32_OFF32_LIBS`
 14888 If `sysconf(_SC_V7_ILP32_OFF32)` returns `-1`, the meaning of this value is unspecified.
 14889 Otherwise, this value is the set of libraries to be given to the `c99` utility to build an
 14890 application using a programming model with 32-bit **int**, **long**, **pointer**, and **off_t** types.

14891 `_CS_POSIX_V7_ILP32_OFFBIG_CFLAGS`
 14892 If `sysconf(_SC_V7_ILP32_OFFBIG)` returns `-1`, the meaning of this value is unspecified.
 14893 Otherwise, this value is the set of initial options to be given to the `c99` utility to build an
 14894 application using a programming model with 32-bit **int**, **long**, and **pointer** types, and an
 14895 **off_t** type using at least 64 bits.

14896 `_CS_POSIX_V7_ILP32_OFFBIG_LDFLAGS`
 14897 If `sysconf(_SC_V7_ILP32_OFFBIG)` returns `-1`, the meaning of this value is unspecified.
 14898 Otherwise, this value is the set of final options to be given to the `c99` utility to build an
 14899 application using a programming model with 32-bit **int**, **long**, and **pointer** types, and an
 14900 **off_t** type using at least 64 bits.

14901 `_CS_POSIX_V7_ILP32_OFFBIG_LIBS`
 14902 If `sysconf(_SC_V7_ILP32_OFFBIG)` returns `-1`, the meaning of this value is unspecified.
 14903 Otherwise, this value is the set of libraries to be given to the `c99` utility to build an
 14904 application using a programming model with 32-bit **int**, **long**, and **pointer** types, and an
 14905 **off_t** type using at least 64 bits.

14906 `_CS_POSIX_V7_LP64_OFF64_CFLAGS`
 14907 If `sysconf(_SC_V7_LP64_OFF64)` returns `-1`, the meaning of this value is unspecified.
 14908 Otherwise, this value is the set of initial options to be given to the `c99` utility to build an
 14909 application using a programming model with 32-bit **int** and 64-bit **long**, **pointer**, and **off_t**
 14910 types.

14911 `_CS_POSIX_V7_LP64_OFF64_LDFLAGS`
 14912 If `sysconf(_SC_V7_LP64_OFF64)` returns `-1`, the meaning of this value is unspecified.
 14913 Otherwise, this value is the set of final options to be given to the `c99` utility to build an
 14914 application using a programming model with 32-bit **int** and 64-bit **long**, **pointer**, and **off_t**
 14915 types.

14916 `_CS_POSIX_V7_LP64_OFF64_LIBS`
 14917 If `sysconf(_SC_V7_LP64_OFF64)` returns `-1`, the meaning of this value is unspecified.
 14918 Otherwise, this value is the set of libraries to be given to the `c99` utility to build an
 14919 application using a programming model with 32-bit **int** and 64-bit **long**, **pointer**, and **off_t**
 14920 types.

14921 `_CS_POSIX_V7_LPBIG_OFFBIG_CFLAGS`
 14922 If `sysconf(_SC_V7_LPBIG_OFFBIG)` returns `-1`, the meaning of this value is unspecified.
 14923 Otherwise, this value is the set of initial options to be given to the `c99` utility to build an
 14924 application using a programming model with an **int** type using at least 32 bits and **long**,
 14925 **pointer**, and **off_t** types using at least 64 bits.

14926 `_CS_POSIX_V7_LPBIG_OFFBIG_LDFLAGS`
 14927 If `sysconf(_SC_V7_LPBIG_OFFBIG)` returns `-1`, the meaning of this value is unspecified.
 14928 Otherwise, this value is the set of final options to be given to the `c99` utility to build an
 14929 application using a programming model with an **int** type using at least 32 bits and **long**,

14930 **pointer**, and **off_t** types using at least 64 bits.

14931 **_CS_POSIX_V7_LPBIG_OFFBIG_LIBS**

14932 If *sysconf*(*_SC_V7_LPBIG_OFFBIG*) returns -1, the meaning of this value is unspecified.

14933 Otherwise, this value is the set of libraries to be given to the *c99* utility to build an

14934 application using a programming model with an **int** type using at least 32 bits and **long**,

14935 **pointer**, and **off_t** types using at least 64 bits.

14936 **_CS_POSIX_V7_THREADS_CFLAGS**

14937 If *sysconf*(*_SC_POSIX_THREADS*) returns -1, the meaning of this value is unspecified.

14938 Otherwise, this value is the set of initial options to be given to the *c99* utility to build a

14939 multi-threaded application. These flags are in addition to those associated with any of the

14940 other *_CS_POSIX_V7_*_CFLAGS* values used to specify particular type size programming

14941 environments.

14942 **_CS_POSIX_V7_THREADS_LDFLAGS**

14943 If *sysconf*(*_SC_POSIX_THREADS*) returns -1, the meaning of this value is unspecified.

14944 Otherwise, this value is the set of final options to be given to the *c99* utility to build a multi-

14945 threaded application. These flags are in addition to those associated with any of the other

14946 *_CS_POSIX_V7_*_LDFLAGS* values used to specify particular type size programming

14947 environments.

14948 **_CS_POSIX_V7_WIDTH_RESTRICTED_ENVS**

14949 This value is a <newline>-separated list of names of programming environments supported

14950 by the implementation in which the widths of the **blksize_t**, **cc_t**, **mode_t**, **nfds_t**, **pid_t**,

14951 **ptrdiff_t**, **size_t**, **speed_t**, **ssize_t**, **suseconds_t**, **tcflag_t**, **wchar_t**, and **wint_t** types are no

14952 greater than the width of type **long**. The format of each name shall be suitable for use with

14953 the *getconf* -v option.

14954 **_CS_V7_ENV**

14955 This is the value that provides the environment variable information (other than that

14956 provided by *_CS_PATH*) that is required by the implementation to create a conforming

14957 environment, as described in the implementation's conformance documentation.

14958 OB The following symbolic constants are reserved for compatibility with Issue 6:

14959 **_CS_POSIX_V6_ILP32_OFF32_CFLAGS**

14960 **_CS_POSIX_V6_ILP32_OFF32_LDFLAGS**

14961 **_CS_POSIX_V6_ILP32_OFF32_LIBS**

14962 **_CS_POSIX_V6_ILP32_OFFBIG_CFLAGS**

14963 **_CS_POSIX_V6_ILP32_OFFBIG_LDFLAGS**

14964 **_CS_POSIX_V6_ILP32_OFFBIG_LIBS**

14965 **_CS_POSIX_V6_LP64_OFF64_CFLAGS**

14966 **_CS_POSIX_V6_LP64_OFF64_LDFLAGS**

14967 **_CS_POSIX_V6_LP64_OFF64_LIBS**

14968 **_CS_POSIX_V6_LPBIG_OFFBIG_CFLAGS**

14969 **_CS_POSIX_V6_LPBIG_OFFBIG_LDFLAGS**

14970 **_CS_POSIX_V6_LPBIG_OFFBIG_LIBS**

14971 **_CS_POSIX_V6_WIDTH_RESTRICTED_ENVS**

14972 **_CS_V6_ENV**

14973 The <unistd.h> header shall define *SEEK_CUR*, *SEEK_END*, and *SEEK_SET* as described in

14974 <stdio.h>.

14975 XSI The <unistd.h> header shall define the following symbolic constants as possible values for the
14976 *function* argument to the *lockf()* function:

14977	F_LOCK	Lock a section for exclusive use.
14978	F_TEST	Test section for locks by other processes.
14979	F_TLOCK	Test and lock a section for exclusive use.
14980	F_ULOCK	Unlock locked sections.

14981 The <unistd.h> header shall define the following symbolic constants for *pathconf()*:

14982	_PC_2_SYMLINKS
14983	_PC_ALLOC_SIZE_MIN
14984	_PC_ASYNC_IO
14985	_PC_CHOWN_RESTRICTED
14986	_PC_FILESIZEBITS
14987	_PC_LINK_MAX
14988	_PC_MAX_CANON
14989	_PC_MAX_INPUT
14990	_PC_NAME_MAX
14991	_PC_NO_TRUNC
14992	_PC_PATH_MAX
14993	_PC_PIPE_BUF
14994	_PC_PRIO_IO
14995	_PC_REC_INCR_XFER_SIZE
14996	_PC_REC_MAX_XFER_SIZE
14997	_PC_REC_MIN_XFER_SIZE
14998	_PC_REC_XFER_ALIGN
14999	_PC_SYMLINK_MAX
15000	_PC_SYNC_IO
15001	_PC_TIMESTAMP_RESOLUTION
15002	_PC_VDISABLE

15003 The <unistd.h> header shall define the following symbolic constants for *sysconf()*:

15004	_SC_2_C_BIND
15005	_SC_2_C_DEV
15006	_SC_2_CHAR_TERM
15007	_SC_2_FORT_DEV
15008	_SC_2_FORT_RUN
15009	_SC_2_LOCALEDEF
15010	_SC_2_PBS
15011	_SC_2_PBS_ACCOUNTING
15012	_SC_2_PBS_CHECKPOINT
15013	_SC_2_PBS_LOCATE
15014	_SC_2_PBS_MESSAGE
15015	_SC_2_PBS_TRACK
15016	_SC_2_SW_DEV
15017	_SC_2_UPE
15018	_SC_2_VERSION
15019	_SC_ADVISORY_INFO
15020	_SC_AIO_LISTIO_MAX
15021	_SC_AIO_MAX

15022 _SC_AIO_PRIO_DELTA_MAX
15023 _SC_ARG_MAX
15024 _SC_ASYNCCHRONOUS_IO
15025 _SC_ATEXIT_MAX
15026 _SC_BARRIERS
15027 _SC_BC_BASE_MAX
15028 _SC_BC_DIM_MAX
15029 _SC_BC_SCALE_MAX
15030 _SC_BC_STRING_MAX
15031 _SC_CHILD_MAX
15032 _SC_CLK_TCK
15033 _SC_CLOCK_SELECTION
15034 _SC_COLL_WEIGHTS_MAX
15035 _SC_CPUTIME
15036 _SC_DELAYTIMER_MAX
15037 _SC_EXPR_NEST_MAX
15038 _SC_FSYNC
15039 _SC_GETGR_R_SIZE_MAX
15040 _SC_GETPW_R_SIZE_MAX
15041 _SC_HOST_NAME_MAX
15042 _SC_IOV_MAX
15043 _SC_IPV6
15044 _SC_JOB_CONTROL
15045 _SC_LINE_MAX
15046 _SC_LOGIN_NAME_MAX
15047 _SC_MAPPED_FILES
15048 _SC_MEMLOCK
15049 _SC_MEMLOCK_RANGE
15050 _SC_MEMORY_PROTECTION
15051 _SC_MESSAGE_PASSING
15052 _SC_MONOTONIC_CLOCK
15053 _SC_MQ_OPEN_MAX
15054 _SC_MQ_PRIO_MAX
15055 _SC_NGROUPS_MAX
15056 _SC_OPEN_MAX
15057 _SC_PAGE_SIZE
15058 _SC_PAGESIZE
15059 _SC_PRIORITIZED_IO
15060 _SC_PRIORITY_SCHEDULING
15061 _SC_RAW_SOCKETS
15062 _SC_RE_DUP_MAX
15063 _SC_READER_WRITER_LOCKS
15064 _SC_REALTIME_SIGNALS
15065 _SC_REGEX
15066 _SC_RTSIG_MAX
15067 _SC_SAVED_IDS
15068 _SC_SEM_NSEMS_MAX
15069 _SC_SEM_VALUE_MAX
15070 _SC_SEMAPHORES
15071 _SC_SHARED_MEMORY_OBJECTS
15072 _SC_SHELL
15073 _SC_SIGQUEUE_MAX

```

15074     _SC_SPAWN
15075     _SC_SPIN_LOCKS
15076     _SC_SPORADIC_SERVER
15077     _SC_SS_REPL_MAX
15078     _SC_STREAM_MAX
15079     _SC_SYMLINK_MAX
15080     _SC_SYNCHRONIZED_IO
15081     _SC_THREAD_ATTR_STACKADDR
15082     _SC_THREAD_ATTR_STACKSIZE
15083     _SC_THREAD_CPU_TIME
15084     _SC_THREAD_DESTRUCTOR_ITERATIONS
15085     _SC_THREAD_KEYS_MAX
15086     _SC_THREAD_PRIO_INHERIT
15087     _SC_THREAD_PRIO_PROTECT
15088     _SC_THREAD_PRIORITY_SCHEDULING
15089     _SC_THREAD_PROCESS_SHARED
15090     _SC_THREAD_ROBUST_PRIO_INHERIT
15091     _SC_THREAD_ROBUST_PRIO_PROTECT
15092     _SC_THREAD_SAFE_FUNCTIONS
15093     _SC_THREAD_SPARADIC_SERVER
15094     _SC_THREAD_STACK_MIN
15095     _SC_THREAD_THREADS_MAX
15096     _SC_THREADS
15097     _SC_TIMEOUTS
15098     _SC_TIMER_MAX
15099     _SC_TIMERS
15100     _SC_TRACE
15101     _SC_TRACE_EVENT_FILTER
15102     _SC_TRACE_EVENT_NAME_MAX
15103     _SC_TRACE_INHERIT
15104     _SC_TRACE_LOG
15105     _SC_TRACE_NAME_MAX
15106     _SC_TRACE_SYS_MAX
15107     _SC_TRACE_USER_EVENT_MAX
15108     _SC_TTY_NAME_MAX
15109     _SC_TYPED_MEMORY_OBJECTS
15110     _SC_TZNAME_MAX
15111     _SC_V7_ILP32_OFF32
15112     _SC_V7_ILP32_OFFBIG
15113     _SC_V7_LP64_OFF64
15114     _SC_V7_LPBIG_OFFBIG
15115     OB _SC_V6_ILP32_OFF32
15116     _SC_V6_ILP32_OFFBIG
15117     _SC_V6_LP64_OFF64
15118     _SC_V6_LPBIG_OFFBIG
15119     _SC_VERSION
15120     _SC_XOPEN_CRYPT
15121     _SC_XOPEN_ENH_I18N
15122     _SC_XOPEN_REALTIME
15123     _SC_XOPEN_REALTIME_THREADS
15124     _SC_XOPEN_SHM
15125     _SC_XOPEN_STREAMS

```

```

15126 _SC_XOPEN_UNIX
15127 _SC_XOPEN_UUCP
15128 _SC_XOPEN_VERSION

```

15129 The two constants `_SC_PAGESIZE` and `_SC_PAGE_SIZE` may be defined to have the same value.

15130 The <unistd.h> header shall define the following symbolic constants for file streams:

```

15131 STDERR_FILENO    File number of stderr; 2.
15132 STDIN_FILENO     File number of stdin; 0.
15133 STDOUT_FILENO    File number of stdout; 1.

```

15134 The <unistd.h> header shall define the following symbolic constant for terminal special
15135 character handling:

```

15136 _POSIX_VDISABLE This symbol shall be defined to be the value of a character that shall
15137                  disable terminal special character handling as described in Section 11.2.6
15138                  (on page 212). This symbol shall always be set to a value other than -1.

```

15139 Type Definitions

15140 The <unistd.h> header shall define the `size_t`, `ssize_t`, `uid_t`, `gid_t`, `off_t`, and `pid_t` types as
15141 described in <sys/types.h>.

15142 The <unistd.h> header shall define the `intptr_t` type as described in <inttypes.h>.

15143 Declarations

15144 The following shall be declared as functions and may also be defined as macros. Function
15145 prototypes shall be provided.

```

15146 int      access(const char *, int);
15147 unsigned alarm(unsigned);
15148 int      chdir(const char *);
15149 int      chown(const char *, uid_t, gid_t);
15150 int      close(int);
15151 size_t   confstr(int, char *, size_t);
15152 XSI char *crypt(const char *, const char *);
15153 int      dup(int);
15154 int      dup2(int, int);
15155 void     _exit(int);
15156 XSI void encrypt(char [64], int);
15157 int      execl(const char *, const char *, ...);
15158 int      execlp(const char *, const char *, ...);
15159 int      execlp(const char *, const char *, ...);
15160 int      execv(const char *, char *const []);
15161 int      execve(const char *, char *const [], char *const []);
15162 int      execvp(const char *, char *const []);
15163 int      faccessat(int, const char *, int, int);
15164 int      fchdir(int);
15165 int      fchown(int, uid_t, gid_t);
15166 int      fchownat(int, const char *, uid_t, gid_t, int);
15167 SIO int  fdatsync(int);
15168 int      fexecve(int, char *const [], char *const []);
15169 pid_t    fork(void);

```

15170		long	fpathconf(int, int);
15171	FSC	int	fsync(int);
15172		int	ftruncate(int, off_t);
15173		char	*getcwd(char *, size_t);
15174		gid_t	getegid(void);
15175		uid_t	geteuid(void);
15176		gid_t	getgid(void);
15177		int	getgroups(int, gid_t []);
15178	XSI	long	gethostid(void);
15179		int	gethostname(char *, size_t);
15180		char	*getlogin(void);
15181		int	getlogin_r(char *, size_t);
15182		int	getopt(int, char * const [], const char *);
15183		pid_t	getpgid(pid_t);
15184		pid_t	getpgrp(void);
15185		pid_t	getpid(void);
15186		pid_t	getppid(void);
15187		pid_t	getsid(pid_t);
15188		uid_t	getuid(void);
15189		int	isatty(int);
15190		int	lchown(const char *, uid_t, gid_t);
15191		int	link(const char *, const char *);
15192		int	linkat(int, const char *, int, const char *, int);
15193	XSI	int	lockf(int, int, off_t);
15194		off_t	lseek(int, off_t, int);
15195	XSI	int	nice(int);
15196		long	pathconf(const char *, int);
15197		int	pause(void);
15198		int	pipe(int [2]);
15199		ssize_t	pread(int, void *, size_t, off_t);
15200		ssize_t	pwrite(int, const void *, size_t, off_t);
15201		ssize_t	read(int, void *, size_t);
15202		ssize_t	readlink(const char *restrict, char *restrict, size_t);
15203		ssize_t	readlinkat(int, const char *restrict, char *restrict, size_t);
15204		int	rmdir(const char *);
15205		int	setegid(gid_t);
15206		int	seteuid(uid_t);
15207		int	setgid(gid_t);
15208		int	setpgid(pid_t, pid_t);
15209	OB XSI	pid_t	setpgrp(void);
15210	XSI	int	setregid(gid_t, gid_t);
15211		int	setreuid(uid_t, uid_t);
15212		pid_t	setsid(void);
15213		int	setuid(uid_t);
15214		unsigned	sleep(unsigned);
15215	XSI	void	swab(const void *restrict, void *restrict, ssize_t);
15216		int	symlink(const char *, const char *);
15217		int	symlinkat(const char *, int, const char *);
15218	XSI	void	sync(void);
15219		long	sysconf(int);
15220		pid_t	tcgetpgrp(int);
15221		int	tcsetpgrp(int, pid_t);

```

15222     int            truncate(const char *, off_t);
15223     char          *ttyname(int);
15224     int           ttyname_r(int, char *, size_t);
15225     int           unlink(const char *);
15226     int           unlinkat(int, const char *, int);
15227     ssize_t       write(int, const void *, size_t);

```

15228 OB Implementations may also include the `pthread_atfork()` prototype as defined in `<pthread.h>`. +
 15229 Implementations may also include the `ctermid()` prototype as defined in `<stdio.h>`.

15230 The `<unistd.h>` header shall declare the following external variables:

```

15231     extern char    *optarg;
15232     extern int     opterr, optind, optopt;

```

15233 APPLICATION USAGE

15234 POSIX.1-2008 only describes the behavior of systems that claim conformance to it. However,
 15235 application developers who want to write applications that adapt to other versions of this
 15236 standard (or to systems that do not conform to any POSIX standard) may find it useful to code
 15237 them so as to conditionally compile different code depending on the value of
 15238 `_POSIX_VERSION`, for example:

```

15239     #if _POSIX_VERSION >= 200112L
15240     /* Use the newer function that copes with large files. */
15241     off_t pos=ftello(fp);
15242     #else
15243     /* Either this is an old version of POSIX, or _POSIX_VERSION is
15244        not even defined, so use the traditional function. */
15245     long pos=ftell(fp);
15246     #endif

```

15247 Earlier versions of POSIX.1-2008 and of the Single UNIX Specification can be identified by the
 15248 following macros:

15249 POSIX.1-1988 standard
 15250 `_POSIX_VERSION == 198808L`

15251 POSIX.1-1990 standard
 15252 `_POSIX_VERSION == 199009L`

15253 ISO POSIX-1: 1996 standard
 15254 `_POSIX_VERSION == 199506L`

15255 Single UNIX Specification, Version 1
 15256 `_XOPEN_UNIX` and `_XOPEN_VERSION == 4`

15257 Single UNIX Specification, Version 2
 15258 `_XOPEN_UNIX` and `_XOPEN_VERSION == 500`

15259 ISO POSIX-1: 2001 and Single UNIX Specification, Version 3
 15260 `_POSIX_VERSION == 200112L`, plus (if the XSI option is supported) `_XOPEN_UNIX` and
 15261 `_XOPEN_VERSION == 600`

15262 POSIX.1-2008 does not make any attempt to define application binary interaction with the
 15263 underlying operating system. However, application developers may find it useful to query
 15264 `_SC_VERSION` at runtime via `sysconf()` to determine whether the current version of the
 15265 operating system supports the necessary functionality as in the following program fragment:

```

15266     if (sysconf(_SC_VERSION) < 200809L) {

```

```

15267         fprintf(stderr, "POSIX.1-2008 system required, terminating \n");
15268         exit(1);
15269     }

```

15270 New applications should not use `_XOPEN_SHM` or `_XOPEN_ENH_I18N`.

15271 RATIONALE

15272 As POSIX.1-2008 evolved, certain options became sufficiently standardized that it was
 15273 concluded that simply requiring one of the option choices was simpler than retaining the option.
 15274 However, for backwards-compatibility, the option flags (with required constant values) are
 15275 retained.

15276 Version Test Macros

15277 The standard developers considered altering the definition of `_POSIX_VERSION` and removing
 15278 `_SC_VERSION` from the specification of `sysconf()` since the utility to an application was deemed
 15279 by some to be minimal, and since the implementation of the functionality is potentially
 15280 problematic. However, they recognized that support for existing application binaries is a
 15281 concern to manufacturers, application developers, and the users of implementations conforming
 15282 to POSIX.1-2008.

15283 While the example using `_SC_VERSION` in the APPLICATION USAGE section does not provide
 15284 the greatest degree of imaginable utility to the application developer or user, it is arguably better
 15285 than a **core** file or some other equally obscure result. (It is also possible for implementations to
 15286 encode and recognize application binaries compiled in various POSIX.1-conforming
 15287 environments, and modify the semantics of the underlying system to conform to the
 15288 expectations of the application.) For the reasons outlined in the preceding paragraphs and in the
 15289 APPLICATION USAGE section, the standard developers elected to retain the `_POSIX_VERSION`
 15290 and `_SC_VERSION` functionality.

15291 Compile-Time Symbolic Constants for System-Wide Options

15292 POSIX.1-2008 includes support in certain areas for the newly adopted policy governing options
 15293 and stubs.

15294 This policy provides flexibility for implementations in how they support options. It also
 15295 specifies how conforming applications can adapt to different implementations that support
 15296 different sets of options. It allows the following:

- 15297 1. If an implementation has no interest in supporting an option, it does not have to provide
 15298 anything associated with that option beyond the announcement that it does not support
 15299 it.
- 15300 2. An implementation can support a partial or incompatible version of an option (as a non-
 15301 standard extension) as long as it does not claim to support the option.
- 15302 3. An application can determine whether the option is supported. A strictly conforming
 15303 application must check this announcement mechanism before first using anything
 15304 associated with the option.

15305 There is an important implication of this policy. POSIX.1-2008 cannot dictate the behavior of
 15306 interfaces associated with an option when the implementation does not claim to support the
 15307 option. In particular, it cannot require that a function associated with an unsupported option
 15308 will fail if it does not perform as specified. However, this policy does not prevent a standard
 15309 from requiring certain functions to always be present, but that they shall always fail on some
 15310 implementations. The `setpgid()` function in the POSIX.1-1990 standard, for example, is
 15311 considered appropriate.

The POSIX standards include various options, and the C-language binding support for an option implies that the implementation must supply data types and function interfaces. An application must be able to discover whether the implementation supports each option.

Any application must consider the following three cases for each option:

1. Option never supported.

The implementation advertises at compile time that the option will never be supported. In this case, it is not necessary for the implementation to supply any of the data types or function interfaces that are provided only as part of the option. The implementation might provide data types and functions that are similar to those defined by POSIX.1-2008, but there is no guarantee for any particular behavior.

2. Option always supported.

The implementation advertises at compile time that the option will always be supported. In this case, all data types and function interfaces shall be available and shall operate as specified.

3. Option might or might not be supported.

Some implementations might not provide a mechanism to specify support of options at compile time. In addition, the implementation might be unable or unwilling to specify support or non-support at compile time. In either case, any application that might use the option at runtime must be able to compile and execute. The implementation must provide, at compile time, all data types and function interfaces that are necessary to allow this. In this situation, there must be a mechanism that allows the application to query, at runtime, whether the option is supported. If the application attempts to use the option when it is not supported, the result is unspecified unless explicitly specified otherwise in POSIX.1-2008.

FUTURE DIRECTIONS

None.

SEE ALSO

[<inttypes.h>](#), [<limits.h>](#), [<stddef.h>](#), [<stdio.h>](#), [<sys/socket.h>](#), [<sys/types.h>](#), [<termios.h>](#), [<wctype.h>](#)

XSH *access()*, *alarm()*, *chown()*, *close()*, *confstr()*, *crypt()*, *ctermid()*, *dup()*, *_Exit()*, *encrypt()*, *exec*, *fchdir()*, *fchown()*, *fdatasync()*, *fork()*, *fpathconf()*, *fsync()*, *ftruncate()*, *getcwd()*, *getegid()*, *geteuid()*, *getgid()*, *getgroups()*, *gethostid()*, *gethostname()*, *getlogin()*, *getopt()*, *getpgid()*, *getpgrp()*, *getpid()*, *getppid()*, *getsid()*, *getuid()*, *isatty()*, *lchown()*, *link()*, *lockf()*, *lseek()*, *nice()*, *pause()*, *pipe()*, *read()*, *readlink()*, *rmdir()*, *setegid()*, *seteuid()*, *setgid()*, *setpgid()*, *setpgrp()*, *setregid()*, *setreuid()*, *setsid()*, *setuid()*, *sleep()*, *swab()*, *symlink()*, *sync()*, *sysconf()*, *tcgetpgrp()*, *tcsetpgrp()*, *truncate()*, *ttyname()*, *unlink()*, *write()*

CHANGE HISTORY

First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 5

The DESCRIPTION is updated for alignment with the POSIX Realtime Extension and the POSIX Threads Extension.

The symbolic constants `_XOPEN_REALTIME` and `_XOPEN_REALTIME_THREADS` are added. `_POSIX2_C_BIND`, `_XOPEN_ENH_I18N`, and `_XOPEN_SHM` must now be set to a value other than `-1` by a conforming implementation.

Large File System extensions are added.

15357 The type of the argument to *sbrk()* is changed from **int** to **intptr_t**.

15358 *_XBS_* constants are added to the list of constants for Options and Option Groups, to the list of
 15359 constants for the *confstr()* function, and to the list of constants to the *sysconf()* function. These
 15360 are all marked EX.

15361 **Issue 6**

15362 *_POSIX2_C_VERSION* is removed.

15363 The Open Group Corrigendum U026/4 is applied, adding the prototype for *fdatasync()*.

15364 The Open Group Corrigendum U026/1 is applied, adding the symbols *_SC_XOPEN_LEGACY*,
 15365 *_SC_XOPEN_REALTIME*, and *_SC_XOPEN_REALTIME_THREADS*.

15366 The symbols *_XOPEN_STREAMS* and *_SC_XOPEN_STREAMS* are added to support the XSI
 15367 STREAMS Option Group.

15368 Text in the DESCRIPTION relating to conformance requirements is moved elsewhere in
 15369 IEEE Std 1003.1-2001.

15370 The LEGACY symbol *_SC_PASS_MAX* is removed.

15371 The following new requirements on POSIX implementations derive from alignment with the
 15372 Single UNIX Specification:

- 15373 • The *_CS_POSIX_** and *_CS_XBS5_** constants are added for the *confstr()* function.
- 15374 • The *_SC_XBS5_** constants are added for the *sysconf()* function.
- 15375 • The symbolic constants *F_ULOCK*, *F_LOCK*, *F_TLOCK*, and *F_TEST* are added.
- 15376 • The **uid_t**, **gid_t**, **off_t**, **pid_t**, and **useconds_t** types are mandated.

15377 The *gethostname()* prototype is added for sockets.

15378 A new section is added for System-Wide Options.

15379 Function prototypes for *setegid()* and *seteuid()* are added.

15380 Option symbolic constants are added for *_POSIX_ADVISORY_INFO*, *_POSIX_CPUTIME*,
 15381 *_POSIX_SPAWN*, *_POSIX_SPORADIC_SERVER*, *_POSIX_THREAD_CPUTIME*,
 15382 *_POSIX_THREAD_SPORADIC_SERVER*, and *_POSIX_TIMEOUTS*, and *pathconf()* variables are
 15383 added for *_PC_ALLOC_SIZE_MIN*, *_PC_REC_INCR_XFER_SIZE*, *_PC_REC_MAX_XFER_SIZE*,
 15384 *_PC_REC_MIN_XFER_SIZE*, and *_PC_REC_XFER_ALIGN* for alignment with IEEE Std
 15385 1003.1d-1999.

15386 The following are added for alignment with IEEE Std 1003.1j-2000:

- 15387 • Option symbolic constants *_POSIX_BARRIERS*, *_POSIX_CLOCK_SELECTION*,
 15388 *_POSIX_MONOTONIC_CLOCK*, *_POSIX_READER_WRITER_LOCKS*,
 15389 *_POSIX_SPIN_LOCKS*, and *_POSIX_TYPED_MEMORY_OBJECTS*
- 15390 • *sysconf()* variables *_SC_BARRIERS*, *_SC_CLOCK_SELECTION*,
 15391 *_SC_MONOTONIC_CLOCK*, *_SC_READER_WRITER_LOCKS*, *_SC_SPIN_LOCKS*, and
 15392 *_SC_TYPED_MEMORY_OBJECTS*

15393 The *_SC_XBS5* macros associated with the ISO/IEC 9899:1990 standard are marked LEGACY,
 15394 and new equivalent *_SC_V6* macros associated with the ISO/IEC 9899:1999 standard are
 15395 introduced.

15396 The *getwd()* function is marked LEGACY.

15397 The **restrict** keyword is added to the prototypes for *readlink()* and *swab()*.

Constants for options are now harmonized, so when supported they take the year of approval of IEEE Std 1003.1-2001 as the value.

The following are added for alignment with IEEE Std 1003.1q-2000:

- Optional symbolic constants `_POSIX_TRACE`, `_POSIX_TRACE_EVENT_FILTER`, `_POSIX_TRACE_LOG`, and `_POSIX_TRACE_INHERIT`
- The `sysconf()` symbolic constants `_SC_TRACE`, `_SC_TRACE_EVENT_FILTER`, `_SC_TRACE_LOG`, and `_SC_TRACE_INHERIT`

The `brk()` and `sbrk()` LEGACY functions are removed.

The Open Group Base Resolution bwg2001-006 is applied, which reworks the XSI versioning information.

The Open Group Base Resolution bwg2001-008 is applied, changing the *namelen* parameter for `gethostname()` from `socklen_t` to `size_t`.

IEEE Std 1003.1-2001/Cor 1-2002, item XBD/TC1/D6/2 is applied, changing “Thread Stack Address Size” to “Thread Stack Size Attribute”.

IEEE Std 1003.1-2001/Cor 1-2002, item XBD/TC1/D6/20 is applied, adding the `_POSIX_IPV6`, `_SC_V6`, and `_SC_RAW_SOCKETS` symbols.

IEEE Std 1003.1-2001/Cor 1-2002, item XBD/TC1/D6/21 is applied, correcting the description in “Constants for Functions” for the `_CS_POSIX_V6_LP64_OFF64_CFLAGS`, `_CS_POSIX_V6_LP64_OFF64_LDFLAGS`, and `_CS_POSIX_V6_LP64_OFF64_LIBS` symbols.

IEEE Std 1003.1-2001/Cor 1-2002, item XBD/TC1/D6/22 is applied, removing the shading for the `_PC*` and `_SC*` constants, since these are mandatory on all implementations.

IEEE Std 1003.1-2001/Cor 1-2002, item XBD/TC1/D6/23 is applied, adding the `_PC_SYMLINK_MAX` and `_SC_SYMLINK_MAX` constants.

IEEE Std 1003.1-2001/Cor 1-2002, item XBD/TC1/D6/24 is applied, correcting the shading and margin code for the `fsync()` function.

IEEE Std 1003.1-2001/Cor 1-2002, item XBD/TC1/D6/25 is applied, adding the following text to the APPLICATION USAGE section: “New applications should not use `_XOPEN_SHM` or `_XOPEN_ENH_I18N`.”.

IEEE Std 1003.1-2001/Cor 2-2004, item XBD/TC2/D6/29 is applied, clarifying the requirements for when constants for Options and Option Groups can be defined or undefined.

IEEE Std 1003.1-2001/Cor 2-2004, item XBD/TC2/D6/30 is applied, changing the `_V6_ILP32_OFF32`, `_V6_ILP32_OFFBIG`, `_V6_LP64_OFF64`, and `_V6_LP64_OFFBIG` symbols to `_POSIX_V6_ILP32_OFF32`, `_POSIX_V6_ILP32_OFFBIG`, `_POSIX_V6_LP64_OFF64`, and `_POSIX_V6_LP64_OFFBIG`, respectively. This is for consistency with the `sysconf()` and *c99* reference pages.

IEEE Std 1003.1-2001/Cor 2-2004, item XBD/TC2/D6/31 is applied, adding that the format of names of programming environments can be obtained using the `getconf -v` option.

IEEE Std 1003.1-2001/Cor 2-2004, item XBD/TC2/D6/32 is applied, deleting the `_SC_FILE_LOCKING`, `_SC_2_C_VERSION`, and `_SC_XOPEN_XCU_VERSION` constants.

IEEE Std 1003.1-2001/Cor 2-2004, item XBD/TC2/D6/33 is applied, adding `_SC_SS_REPL_MAX`, `_SC_TRACE_EVENT_NAME_MAX`, `_SC_TRACE_NAME_MAX`, `_SC_TRACE_SYS_MAX`, and `_SC_TRACE_USER_EVENT_MAX` to the list of symbolic constants for `sysconf()`.

15441 IEEE Std 1003.1-2001/Cor 2-2004, item XBD/TC2/D6/34 is applied, updating the prototype for
15442 the *symlink()* function to match that in the System Interfaces volume of IEEE Std 1003.1-2001.

15443 IEEE Std 1003.1-2001/Cor 2-2004, item XBD/TC2/D6/35 is applied, adding *_PC_2_SYMLINKS*
15444 to the symbolic constants list for *pathconf()*. This corresponds to the definition of
15445 *POSIX2_SYMLINKS* in the Shell and Utilities volume of IEEE Std 1003.1-2001.

15446 **Issue 7**

15447 Austin Group Interpretations 1003.1-2001 #026 and #047 are applied.

15448 Austin Group Interpretation 1003.1-2001 #166 is applied to permit an additional compiler flag to
15449 enable threads.

15450 Austin Group Interpretation 1003.1-2001 #178 is applied, clarifying the values allowed for
15451 *_POSIX2_CHAR_TERM*.

15452 SD5-XBD-ERN-41 is applied, adding the *_POSIX2_SYMLINKS* constant.

15453 SD5-XBD-ERN-76 and SD5-XBD-ERN-77 are applied.

15454 Symbols to support the UUCP Utilities option are added.

15455 The variables for the supported programming environments are updated to be V7.

15456 The LEGACY and obsolescent symbols are removed.

15457 The *faccessat()*, *fchownat()*, *fexecve()*, *linkat()*, *readlinkat()*, *symlinkat()*, and *unlinkat()* functions
15458 are added from The Open Group Technical Standard, 2006, Extended API Set Part 2.

15459 The *_POSIX_TRACE** constants from the Trace option are marked obsolescent.

15460 The *_POSIX2_PBS** constants from the Batch Environment Services and Utilities option are
15461 marked obsolescent.

15462 Functionality relating to the Asynchronous Input and Output, Barriers, Clock Selection, Memory
15463 Mapped Files, Memory Protection, Realtime Signals Extension, Semaphores, Spin Locks,
15464 Threads, Timeouts, and Timers options is moved to the Base.

15465 Functionality relating to the Thread Priority Protection and Thread Priority Inheritance options
15466 is changed to be Non-Robust Mutex or Robust Mutex Priority Protection and Non-Robust Mutex
15467 or Robust Mutex Priority Inheritance, respectively.

15468 This reference page is clarified with respect to macros and symbolic constants.

15469 Changes are made related to support for finegrained timestamps and the
15470 *_POSIX_TIMESTAMP_RESOLUTION* constant is added.

15471 The *_SC_THREAD_ROBUST_PRIO_INHERIT* and *_SC_THREAD_ROBUST_PRIO_PROTECT*
15472 symbolic constants are added.

15473 POSIX.1-2008, Technical Corrigendum 1, XBD/TC1-2008/0078 [311], XBD/TC1-2008/0079 [209], +
15474 and XBD/TC1-2008/0080 [360] are applied.

15475 **NAME**

15476 utime.h — access and modification times structure

15477 **SYNOPSIS**15478 OB `#include <utime.h>`15479 **DESCRIPTION**15480 The <utime.h> header shall declare the **utimbuf** structure, which shall include the following
15481 members:15482 `time_t` `actime` Access time.
15483 `time_t` `modtime` Modification time.

15484 The times shall be measured in seconds since the Epoch.

15485 The <utime.h> header shall define the **time_t** type as described in <sys/types.h>.15486 The following shall be declared as a function and may also be defined as a macro. A function
15487 prototype shall be provided.15488 `int utime(const char *, const struct utimbuf *);`15489 **APPLICATION USAGE**15490 The *utime()* function only allows setting file timestamps to the nearest second. Applications
15491 should use the *utimensat()* function instead. See <sys/stat.h>.15492 **RATIONALE**

15493 None.

15494 **FUTURE DIRECTIONS**

15495 The <utime.h> header may be removed in a future version.

15496 **SEE ALSO**15497 [<sys/stat.h>](#), [<sys/types.h>](#)15498 XSH *futimens()*, *utime()*15499 **CHANGE HISTORY**

15500 First released in Issue 3.

15501 **Issue 6**15502 The following new requirements on POSIX implementations derive from alignment with the
15503 Single UNIX Specification:

- 15504
- The **time_t** type is defined.

15505 **Issue 7**

15506 The <utime.h> header is marked obsolescent.

15507 **NAME**

15508 utmpx.h — user accounting database definitions

15509 **SYNOPSIS**

15510 XSI `#include <utmpx.h>`

15511 **DESCRIPTION**

15512 The <utmpx.h> header shall define the **utmpx** structure that shall include at least the following
15513 members:

15514	char	ut_user[]	User login name.
15515	char	ut_id[]	Unspecified initialization process identifier.
15516	char	ut_line[]	Device name.
15517	pid_t	ut_pid	Process ID.
15518	short	ut_type	Type of entry.
15519	struct timeval	ut_tv	Time entry was made.

15520 The <utmpx.h> header shall define the **pid_t** type through **typedef**, as described in
15521 <sys/types.h>.

15522 The <utmpx.h> header shall define the **timeval** structure as described in <sys/time.h>.

15523 Inclusion of the <utmpx.h> header may also make visible all symbols from <sys/time.h>.

15524 The <utmpx.h> header shall define the following symbolic constants as possible values for the
15525 *ut_type* member of the **utmpx** structure:

15526	EMPTY	No valid user accounting information.
15527	BOOT_TIME	Identifies time of system boot.
15528	OLD_TIME	Identifies time when system clock changed.
15529	NEW_TIME	Identifies time after system clock changed.
15530	USER_PROCESS	Identifies a process.
15531	INIT_PROCESS	Identifies a process spawned by the init process.
15532	LOGIN_PROCESS	Identifies the session leader of a logged-in user.
15533	DEAD_PROCESS	Identifies a session leader who has exited.

15534 The following shall be declared as functions and may also be defined as macros. Function
15535 prototypes shall be provided.

```

15536 void          endutxent(void);
15537 struct utmpx *getutxent(void);
15538 struct utmpx *getutxid(const struct utmpx *);
15539 struct utmpx *getutxline(const struct utmpx *);
15540 struct utmpx *pututxline(const struct utmpx *);
15541 void          setutxent(void);

```

15542 **APPLICATION USAGE**

15543 None.

15544 **RATIONALE**

15545 None.

15546 **FUTURE DIRECTIONS**

15547 None.

15548 **SEE ALSO**15549 [**<sys/time.h>**](#), [**<sys/types.h>**](#)15550 XSH *endutxent()*15551 **CHANGE HISTORY**

15552 First released in Issue 4, Version 2.

15553 **NAME**

15554 wchar.h — wide-character handling

15555 **SYNOPSIS**

15556 #include <wchar.h>

15557 **DESCRIPTION**

15558 CX Some of the functionality described on this reference page extends the ISO C standard.
15559 Applications shall define the appropriate feature test macro (see XSH [Section 2.2](#), on page 472) to
15560 enable the visibility of these symbols in this header.

15561 The <wchar.h> header shall define the following types:

15562 CX **FILE** As described in <stdio.h>.

15563 CX **locale_t** As described in <locale.h>.

15564 **mbstate_t** An object type other than an array type that can hold the conversion state
15565 information necessary to convert between sequences of (possibly multi-byte)
15566 CX characters and wide characters. If a codeset is being used such that an
15567 **mbstate_t** needs to preserve more than two levels of reserved state, the results
15568 are unspecified.

15569 **size_t** As described in <stddef.h>.

15570 CX **va_list** As described in <stdarg.h>.

15571 **wchar_t** As described in <stddef.h>.

15572 OB XSI **wctype_t** A scalar type of a data object that can hold values which represent locale-
15573 specific character classification.

15574 **wint_t** An integer type capable of storing any valid value of **wchar_t** or WEOF.

15575 The tag **tm** shall be declared as naming an incomplete structure type, the contents of which are
15576 described in the <time.h> header.

15577 The implementation shall support one or more programming environments in which the width
15578 of **wint_t** is no greater than the width of type **long**. The names of these programming
15579 environments can be obtained using the *confstr()* function or the *getconf* utility.

15580 The <wchar.h> header shall define the following macros:

15581 **WCHAR_MAX** As described in <stdint.h>.

15582 **WCHAR_MIN** As described in <stdint.h>.

15583 **WEOF** Constant expression of type **wint_t** that is returned by several WP functions to
15584 indicate end-of-file.

15585 **NULL** As described in <stddef.h>.

15586 CX Inclusion of the <wchar.h> header may make visible all symbols from the headers <ctype.h>,
15587 <string.h>, <stdarg.h>, <stddef.h>, <stdio.h>, <stdlib.h>, and <time.h>.

15588 The following shall be declared as functions and may also be defined as macros. Function
15589 prototypes shall be provided for use with ISO C standard compilers.

15590 **wint_t** btowc(int);
15591 **wint_t** fgetwc(FILE *);
15592 **wchar_t** *fgetws(wchar_t *restrict, int, FILE *restrict);
15593 **wint_t** fputwc(wchar_t, FILE *);
15594 **int** fputws(const wchar_t *restrict, FILE *restrict);

```

15595     int          fwide(FILE *, int);
15596     int          fwprintf(FILE *restrict, const wchar_t *restrict, ...);
15597     int          fwscanf(FILE *restrict, const wchar_t *restrict, ...);
15598     wint_t       getwc(FILE *);
15599     wint_t       getwchar(void);
15600     OB XSI      int          iswalnum(wint_t);
15601                int          iswalpha(wint_t);
15602                int          iswcntrl(wint_t);
15603                int          iswctype(wint_t, wctype_t);
15604                int          iswdigit(wint_t);
15605                int          iswgraph(wint_t);
15606                int          iswlower(wint_t);
15607                int          iswprint(wint_t);
15608                int          iswpunct(wint_t);
15609                int          iswspace(wint_t);
15610                int          iswupper(wint_t);
15611                int          iswxdigit(wint_t);
15612     size_t       mbrlen(const char *restrict, size_t, mbstate_t *restrict);
15613     size_t       mbrtowc(wchar_t *restrict, const char *restrict, size_t,
15614                          mbstate_t *restrict);
15615     int          mbsinit(const mbstate_t *);
15616     CX          size_t       mbsnrtowcs(wchar_t *restrict, const char **restrict,
15617                                         size_t, size_t, mbstate_t *restrict);
15618     size_t       mbsrtowcs(wchar_t *restrict, const char **restrict, size_t,
15619                          mbstate_t *restrict);
15620     CX          FILE        *open_wmemstream(wchar_t **, size_t *);
15621     wint_t       putwc(wchar_t, FILE *);
15622     wint_t       putwchar(wchar_t);
15623     int          swprintf(wchar_t *restrict, size_t,
15624                          const wchar_t *restrict, ...);
15625     int          swscanf(const wchar_t *restrict,
15626                          const wchar_t *restrict, ...);
15627     OB XSI      wint_t       tolower(wint_t);
15628     wint_t       towupper(wint_t);
15629     wint_t       ungetwc(wint_t, FILE *);
15630     int          vfwprintf(FILE *restrict, const wchar_t *restrict, va_list);
15631     int          vfwscanf(FILE *restrict, const wchar_t *restrict, va_list);
15632     int          vswprintf(wchar_t *restrict, size_t,
15633                          const wchar_t *restrict, va_list);
15634     int          vswscanf(const wchar_t *restrict, const wchar_t *restrict,
15635                          va_list);
15636     int          vwprintf(const wchar_t *restrict, va_list);
15637     int          vwscanf(const wchar_t *restrict, va_list);
15638     CX          wchar_t      *wcpcpy(wchar_t *restrict, const wchar_t *restrict);
15639     wchar_t      *wcpncpy(wchar_t *restrict, const wchar_t *restrict, size_t);
15640     size_t       wcrtoomb(char *restrict, wchar_t, mbstate_t *restrict);
15641     CX          int          wcscasecmp(const wchar_t *, const wchar_t *);
15642     int          wcscasecmp_l(const wchar_t *, const wchar_t *, locale_t);
15643     wchar_t      *wcscat(wchar_t *restrict, const wchar_t *restrict);
15644     wchar_t      *wcschr(const wchar_t *, wchar_t);
15645     int          wcscmp(const wchar_t *, const wchar_t *);
15646     int          wcscoll(const wchar_t *, const wchar_t *);

```


15647	CX	int	wscoll_l(const wchar_t *, const wchar_t *, locale_t);
15648		wchar_t	*wcscpy(wchar_t *restrict, const wchar_t *restrict);
15649		size_t	wcscspn(const wchar_t *, const wchar_t *);
15650	CX	wchar_t	*wcsdup(const wchar_t *);
15651		size_t	wcsftime(wchar_t *restrict, size_t,
15652			const wchar_t *restrict, const struct tm *restrict);
15653		size_t	wcslen(const wchar_t *);
15654	CX	int	wcsncasecmp(const wchar_t *, const wchar_t *, size_t);
15655		int	wcsncasecmp_l(const wchar_t *, const wchar_t *, size_t,
15656			locale_t);
15657		wchar_t	*wcsncat(wchar_t *restrict, const wchar_t *restrict, size_t);
15658		int	wcsncmp(const wchar_t *, const wchar_t *, size_t);
15659		wchar_t	*wcsncpy(wchar_t *restrict, const wchar_t *restrict, size_t);
15660	CX	size_t	wcsnlen(const wchar_t *, size_t);
15661		size_t	wcsnrtombs(char *restrict, const wchar_t **restrict, size_t,
15662			size_t, mbstate_t *restrict);
15663		wchar_t	*wcpbrk(const wchar_t *, const wchar_t *);
15664		wchar_t	*wcrchr(const wchar_t *, wchar_t);
15665		size_t	wcrtombs(char *restrict, const wchar_t **restrict,
15666			size_t, mbstate_t *restrict);
15667		size_t	wcsspn(const wchar_t *, const wchar_t *);
15668		wchar_t	*wcssstr(const wchar_t *restrict, const wchar_t *restrict);
15669		double	wcstod(const wchar_t *restrict, wchar_t **restrict);
15670		float	wcstof(const wchar_t *restrict, wchar_t **restrict);
15671		wchar_t	*wcstok(wchar_t *restrict, const wchar_t *restrict,
15672			wchar_t **restrict);
15673		long	wcstol(const wchar_t *restrict, wchar_t **restrict, int);
15674		long double	wcstold(const wchar_t *restrict, wchar_t **restrict);
15675		long long	wcstoll(const wchar_t *restrict, wchar_t **restrict, int);
15676		unsigned long	wcstoul(const wchar_t *restrict, wchar_t **restrict, int);
15677		unsigned long long	wcstoull(const wchar_t *restrict, wchar_t **restrict, int);
15678			
15679	XSI	int	wcswidth(const wchar_t *, size_t);
15680		size_t	wcsxfrm(wchar_t *restrict, const wchar_t *restrict, size_t);
15681	CX	size_t	wcsxfrm_l(wchar_t *restrict, const wchar_t *restrict,
15682			size_t, locale_t);
15683		int	wctob(wint_t);
15684	OB XSI	wctype_t	wctype(const char *);
15685	XSI	int	wcwidth(wchar_t);
15686		wchar_t	*wmemchr(const wchar_t *, wchar_t, size_t);
15687		int	wmemcmp(const wchar_t *, const wchar_t *, size_t);
15688		wchar_t	*wmemcpy(wchar_t *restrict, const wchar_t *restrict, size_t);
15689		wchar_t	*wmemmove(wchar_t *, const wchar_t *, size_t);
15690		wchar_t	*wmemset(wchar_t *, wchar_t, size_t);
15691		int	wprintf(const wchar_t *restrict, ...);
15692		int	wscanf(const wchar_t *restrict, ...);

APPLICATION USAGE

The `iswblank()` function was a late addition to the ISO C standard and was introduced at the same time as the ISO C standard introduced **<wctype.h>**, which contains all of the `isw*()` functions. The Open Group Base Specifications had previously aligned with the MSE working draft and had introduced the rest of the `isw*()` functions into **<wchar.h>**. For backwards-compatibility, the original set of `isw*()` functions, without `iswblank()`, are permitted (as part of the XSI option) in **<wchar.h>**. For maximum portability, applications should include **<wctype.h>** in order to obtain declarations for the `isw*()` functions. This compatibility has been made obsolescent.

RATIONALE

In the ISO C standard, the symbols referenced as XSI extensions are in **<wctype.h>**. Their presence here is thus an extension.

FUTURE DIRECTIONS

None.

SEE ALSO

<ctype.h>, **<locale.h>**, **<stdarg.h>**, **<stddef.h>**, **<stdint.h>**, **<stdio.h>**, **<stdlib.h>**, **<string.h>**, **<time.h>**, **<wctype.h>**

XSH Section 2.2 (on page 472), `btowc()`, `confstr()`, `fgetwc()`, `fgetws()`, `fputwc()`, `fputws()`, `fwide()`, `fwprintf()`, `fwscanf()`, `getwc()`, `getwchar()`, `iswalnum()`, `iswalpunct()`, `iswalpha()`, `iswcntrl()`, `iswctype()`, `iswdigit()`, `iswgraph()`, `iswlower()`, `iswprint()`, `iswpunct()`, `iswspace()`, `iswupper()`, `iswxdigit()`, `mbrlen()`, `mbrtowc()`, `mbsinit()`, `mbsrtowcs()`, `open_memstream()`, `putwc()`, `putwchar()`, `towlower()`, `towupper()`, `ungetwc()`, `vfwprintf()`, `vfwscanf()`, `wcrtomb()`, `wcscasecmp()`, `wcscat()`, `wcschr()`, `wcscmp()`, `wcscoll()`, `wcscpy()`, `wcscspn()`, `wcsdup()`, `wcsftime()`, `wcslen()`, `wcsncat()`, `wcsncmp()`, `wcsncpy()`, `wcspbrk()`, `wcsrchr()`, `wcsrtombs()`, `wcsspn()`, `wcsstr()`, `wcstod()`, `wcstok()`, `wcstol()`, `wcstoul()`, `wcswidth()`, `wcsxfrm()`, `wctob()`, `wctype()`, `wcwidth()`, `wmemchr()`, `wmemcmp()`, `wmemcpy()`, `wmemmove()`, `wmemset()`

XCU `getconf`

CHANGE HISTORY

First released in Issue 4.

Issue 5

Aligned with the ISO/IEC 9899:1990/Amendment 1:1995 (E).

Issue 6

The Open Group Corrigendum U021/10 is applied. The prototypes for `wcswidth()` and `wcwidth()` are marked as extensions.

The Open Group Corrigendum U028/5 is applied, correcting the prototype for the `mbsinit()` function.

The following changes are made for alignment with the ISO/IEC 9899:1999 standard:

- Various function prototypes are updated to add the **restrict** keyword.
- The functions `vfwscanf()`, `vwscanf()`, `wcstof()`, `wcstold()`, `wcstoll()`, and `wcstoull()` are added.

The type **wctype_t**, the `isw*()`, `to*()`, and `wctype()` functions are marked as XSI extensions.

IEEE Std 1003.1-2001/Cor 1-2002, item XBD/TC1/D6/26 is applied, adding the APPLICATION USAGE section.

15736 **Issue 7**

15737 The *mbsnrtowcs()*, *open_wmemstream()*, *wcpcpy()*, *wcpncpy()*, *wscasecmp()*, *wcsdup()*,
15738 *wcsncasecmp()*, *wcsnlen()*, and *wcsnrtombs()* functions are added from The Open Group
15739 Technical Standard, 2006, Extended API Set Part 1.

15740 The *wscasecmp_l()*, *wcsncasecmp_l()*, *wscoll_l()*, and *wcsxfrm_l()* functions are added from The
15741 Open Group Technical Standard, 2006, Extended API Set Part 4.

15742 The **wctype_t** type, and the *isw**, *towlower()*, and *towupper()* functions are marked obsolescent in
15743 <wchar.h> since the ISO C standard requires the declarations to be in <wctype.h>.

15744 This reference page is clarified with respect to macros and symbolic constants, and a declaration
15745 for the **locale_t** type is added.

15746 POSIX.1-2008, Technical Corrigendum 1, XBD/TC1-2008/0081 [380] is applied.

+

15747 **NAME**15748 `wctype.h` — wide-character classification and mapping utilities15749 **SYNOPSIS**15750 `#include <wctype.h>`15751 **DESCRIPTION**

15752 CX Some of the functionality described on this reference page extends the ISO C standard.
 15753 Applications shall define the appropriate feature test macro (see XSH [Section 2.2](#), on page 472) to
 15754 enable the visibility of these symbols in this header.

15755 The `<wctype.h>` header shall define the following types:15756 **wint_t** As described in `<wchar.h>`.

15757 **wctrans_t** A scalar type that can hold values which represent locale-specific character
 15758 mappings.

15759 **wctype_t** As described in `<wchar.h>`.15760 CX The `<wctype.h>` header shall define the **locale_t** type as described in `<locale.h>`.15761 The `<wctype.h>` header shall define the following macro:15762 **WEOF** As described in `<wchar.h>`.

15763 For all functions described in this header that accept an argument of type **wint_t**, the value is
 15764 representable as a **wchar_t** or equals the value of WEOF. If this argument has any other value,
 15765 the behavior is undefined.

15766 The behavior of these functions shall be affected by the `LC_CTYPE` category of the current locale.

15767 CX Inclusion of the `<wctype.h>` header may make visible all symbols from the headers `<ctype.h>`,
 15768 `<stdarg.h>`, `<stddef.h>`, `<stdio.h>`, `<stdlib.h>`, `<string.h>`, `<time.h>`, and `<wchar.h>`.

15769 The following shall be declared as functions and may also be defined as macros. Function
 15770 prototypes shall be provided for use with ISO C standard compilers.

```

15771 int      iswalnum(wint_t);
15772 CX int    iswalnum_l(wint_t, locale_t);
15773 int      iswalpha(wint_t);
15774 CX int    iswalpha_l(wint_t, locale_t);
15775 int      iswblank(wint_t);
15776 CX int    iswblank_l(wint_t, locale_t);
15777 int      iswcntrl(wint_t);
15778 CX int    iswcntrl_l(wint_t, locale_t);
15779 int      iswctype(wint_t, wctype_t);
15780 CX int    iswctype_l(wint_t, wctype_t, locale_t);
15781 int      iswdigit(wint_t);
15782 CX int    iswdigit_l(wint_t, locale_t);
15783 int      iswgraph(wint_t);
15784 CX int    iswgraph_l(wint_t, locale_t);
15785 int      iswlower(wint_t);
15786 CX int    iswlower_l(wint_t, locale_t);
15787 int      iswprint(wint_t);
15788 CX int    iswprint_l(wint_t, locale_t);
15789 int      iswpunct(wint_t);
15790 CX int    iswpunct_l(wint_t, locale_t);
15791 int      iswspace(wint_t);
  
```

```

15792 CX      int      iswspace_l(wint_t, locale_t);
15793          int      iswupper(wint_t);
15794 CX      int      iswupper_l(wint_t, locale_t);
15795          int      iswxdigit(wint_t);
15796 CX      int      iswxdigit_l(wint_t, locale_t);
15797          wint_t    towctrans(wint_t, wctrans_t);
15798 CX      wint_t    towctrans_l(wint_t, wctrans_t, locale_t);
15799          wint_t    towlower(wint_t);
15800 CX      wint_t    towlower_l(wint_t, locale_t);
15801          wint_t    towupper(wint_t);
15802 CX      wint_t    towupper_l(wint_t, locale_t);
15803          wctrans_t wctrans(const char *);
15804 CX      wctrans_t wctrans_l(const char *, locale_t);
15805          wctype_t  wctype(const char *);
15806 CX      wctype_t  wctype_l(const char *, locale_t);

```

15807 APPLICATION USAGE

15808 None.

15809 RATIONALE

15810 None.

15811 FUTURE DIRECTIONS

15812 None.

15813 SEE ALSO

15814 <ctype.h>, <locale.h>, <stdarg.h>, <stddef.h>, <stdio.h>, <stdlib.h>, <string.h>, <time.h>,
15815 <wchar.h>

15816 XSH Section 2.2 (on page 472), *iswalnum()*, *iswalpha()*, *iswblank()*, *iswcntrl()*, *iswctype()*,
15817 *iswdigit()*, *iswgraph()*, *iswlower()*, *iswprint()*, *iswpunct()*, *iswspace()*, *iswupper()*, *iswxdigit()*,
15818 *setlocale()*, *towctrans()*, *towlower()*, *towupper()*, *wctrans()*, *wctype()*

15819 CHANGE HISTORY

15820 First released in Issue 5. Derived from the ISO/IEC 9899:1990/Amendment 1:1995 (E).

15821 Issue 6

15822 The *iswblank()* function is added for alignment with the ISO/IEC 9899:1999 standard.

15823 Issue 7

15824 SD5-XBD-ERN-6 is applied.

15825 The **l()* functions are added from The Open Group Technical Standard, 2006, Extended API Set
15826 Part 4.

15827 This reference page is clarified with respect to macros and symbolic constants.

15828 **NAME**

15829 wordexp.h — word-expansion types

15830 **SYNOPSIS**

15831 #include <wordexp.h>

15832 **DESCRIPTION**

15833 The <wordexp.h> header shall define the structures and symbolic constants used by the
 15834 *wordexp()* and *wordfree()* functions.

15835 The <wordexp.h> header shall define the **wordexp_t** structure type, which shall include at least
 15836 the following members:

15837 `size_t we_wordc` Count of words matched by *words*.
 15838 `char **we_wordv` Pointer to list of expanded words.
 15839 `size_t we_offs` Slots to reserve at the beginning of *we_wordv*.

15840 The <wordexp.h> header shall define the following symbolic constants for use as flags for the
 15841 *wordexp()* function:

15842 **WRDE_APPEND** Append words to those previously generated.
 15843 **WRDE_DOOFFS** Number of null pointers to prepend to *we_wordv*.
 15844 **WRDE_NOCMD** Fail if command substitution is requested.
 15845 **WRDE_REUSE** The *pwordexp* argument was passed to a previous successful call to
 15846 *wordexp()*, and has not been passed to *wordfree()*. The result is the same
 15847 as if the application had called *wordfree()* and then called *wordexp()*
 15848 without **WRDE_REUSE**.

15849 **WRDE_SHOWERR** Do not redirect *stderr* to **/dev/null**.

15850 **WRDE_UNDEF** Report error on an attempt to expand an undefined shell variable.

15851 The <wordexp.h> header shall define the following symbolic constants as error return values:

15852 **WRDE_BADCHAR** One of the unquoted characters—<newline>, ' | ', ' & ', ' ; ', ' < ', ' > ',
 15853 ' (', ') ', ' { ', ' } '—appears in *words* in an inappropriate context.
 15854 **WRDE_BADVAL** Reference to undefined shell variable when **WRDE_UNDEF** is set in *flags*.
 15855 **WRDE_CMDSUB** Command substitution requested when **WRDE_NOCMD** was set in *flags*.
 15856 **WRDE_NOSPACE** Attempt to allocate memory failed.
 15857 **WRDE_SYNTAX** Shell syntax error, such as unbalanced parentheses or unterminated
 15858 string.

15859 The <wordexp.h> header shall define the **size_t** type as described in <stddef.h>.

15860 The following shall be declared as functions and may also be defined as macros. Function
 15861 prototypes shall be provided.

15862 `int wordexp(const char *restrict, wordexp_t *restrict, int);`
 15863 `void wordfree(wordexp_t *);`

15864 **APPLICATION USAGE**

15865 None.

15866 **RATIONALE**

15867 None.

15868 **FUTURE DIRECTIONS**

15869 None.

15870 **SEE ALSO**

15871 <stddef.h>

15872 XSH [Section 2.6](#)

15873 **CHANGE HISTORY**

15874 First released in Issue 4. Derived from the ISO POSIX-2 standard.

15875 **Issue 6**

15876 The **restrict** keyword is added to the prototype for *wordexp()*.

15877 The WRDE_NOSYS constant is marked obsolescent.

15878 **Issue 7**

15879 The obsolescent WRDE_NOSYS constant is removed.

15880 This reference page is clarified with respect to macros and symbolic constants.

15881

Technical Standard

15882

Vol. 2:

15883

System Interfaces, Issue 7

15884

The Open Group

15885

The Institute of Electrical and Electronics Engineers, Inc.

15886
15887

Chapter 1 Introduction

15888
15889

The System Interfaces volume of POSIX.1-2008 describes the interfaces offered to application programs by POSIX-conformant systems.

1.1 Relationship to Other Formal Standards

15891
15892

Great care has been taken to ensure that this volume of POSIX.1-2008 is fully aligned with the following standards:

15893
15894
15895
15896

ISO C (1999)
ISO/IEC 9899:1999, Programming Languages — C, including ISO/IEC 9899:1999/Cor.1:2001(E), ISO/IEC 9899:1999/Cor.2:2004(E), and ISO/IEC 9899:1999/Cor.3.

15897
15898
15899
15900
15901
15902

Parts of the ISO/IEC 9899:1999 standard (hereinafter referred to as the ISO C standard) are referenced to describe requirements also mandated by this volume of POSIX.1-2008. Some functions and headers included within this volume of POSIX.1-2008 have a version in the ISO C standard; in this case CX markings are added as appropriate to show where the ISO C standard has been extended (see [Section 1.7.1](#), on page 7). Any conflict between this volume of POSIX.1-2008 and the ISO C standard is unintentional.

15903
15904

This volume of POSIX.1-2008 also allows, but does not require, mathematics functions to support IEEE Std 754-1985 and IEEE Std 854-1987.

1.2 Format of Entries

15906
15907

The entries in [Chapter 3](#) are based on a common format as follows. The only sections relating to conformance are the SYNOPSIS, DESCRIPTION, RETURN VALUE, and ERRORS sections.

15908
15909

NAME

This section gives the name or names of the entry and briefly states its purpose.

15910
15911
15912
15913

SYNOPSIS

This section summarizes the use of the entry being described. If it is necessary to include a header to use this function, the names of such headers are shown, for example:

15914

```
#include <stdio.h>
```

15915
15916

DESCRIPTION

This section describes the functionality of the function or header.

15917
15918

RETURN VALUE

This section indicates the possible return values, if any.

15919

If the implementation can detect errors, “successful completion” means that no error

15920 has been detected during execution of the function. If the implementation does detect
15921 an error, the error is indicated.

15922 For functions where no errors are defined, “successful completion” means that if the
15923 implementation checks for errors, no error has been detected. If the implementation can
15924 detect errors, and an error is detected, the indicated return value is returned and *errno*
15925 may be set.

15926 **ERRORS**

15927 This section gives the symbolic names of the error values returned by a function or
15928 stored into a variable accessed through the symbol *errno* if an error occurs.

15929 “No errors are defined” means that error values returned by a function or stored into a
15930 variable accessed through the symbol *errno*, if any, depend on the implementation.

15931 **EXAMPLES**

15932 This section is informative.

15933 This section gives examples of usage, where appropriate. In the event of conflict
15934 between an example and a normative part of this volume of POSIX.1-2008, the
15935 normative material is to be taken as correct.

15936 **APPLICATION USAGE**

15937 This section is informative.

15938 This section gives warnings and advice to application developers about the entry. In the
15939 event of conflict between warnings and advice and a normative part of this volume of
15940 POSIX.1-2008, the normative material is to be taken as correct.

15941 **RATIONALE**

15942 This section is informative.

15943 This section contains historical information concerning the contents of this volume of
15944 POSIX.1-2008 and why features were included or discarded by the standard
15945 developers.

15946 **FUTURE DIRECTIONS**

15947 This section is informative.

15948 This section provides comments which should be used as a guide to current thinking;
15949 there is not necessarily a commitment to adopt these future directions.

15950 **SEE ALSO**

15951 This section is informative.

15952 This section gives references to related information.

15953 **CHANGE HISTORY**

15954 This section is informative.

15955 This section shows the derivation of the entry and any significant changes that have
15956 been made to it.

15957

15958

15959

This chapter covers information that is relevant to all the functions specified in [Chapter 3](#) and XBD [Chapter 13](#) (on page 219).

15960

15961

2.1 Use and Implementation of Interfaces

15962

2.1.1 Use and Implementation of Functions

15963

Each of the following statements shall apply to all functions unless explicitly stated otherwise in the detailed descriptions that follow:

15964

15965

1. If an argument to a function has an invalid value (such as a value outside the domain of the function, or a pointer outside the address space of the program, or a null pointer), the behavior is undefined.

15966

15967

15968

2. Any function declared in a header may also be implemented as a macro defined in the header, so a function should not be declared explicitly if its header is included. Any macro definition of a function can be suppressed locally by enclosing the name of the function in parentheses, because the name is then not followed by the <left-parenthesis> that indicates expansion of a macro function name. For the same syntactic reason, it is permitted to take the address of a function even if it is also defined as a macro. The use of the C-language **#undef** construct to remove any such macro definition shall also ensure that an actual function is referred to.

15969

15970

15971

15972

15973

15974

15975

15976

3. Any invocation of a function that is implemented as a macro shall expand to code that evaluates each of its arguments exactly once, fully protected by parentheses where necessary, so it is generally safe to use arbitrary expressions as arguments.

15977

15978

15979

4. Provided that a function can be declared without reference to any type defined in a header, it is also permissible to declare the function explicitly and use it without including its associated header.

15980

15981

15982

5. If a function that accepts a variable number of arguments is not declared (explicitly or by including its associated header), the behavior is undefined.

15983

2.1.2 Use and Implementation of Macros

Each of the following statements shall apply to all macros unless explicitly stated otherwise:

1. Any definition of an object-like macro in a header shall expand to code that is fully protected by parentheses where necessary, so that it groups in an arbitrary expression as if it were a single identifier.
2. All object-like macros listed as expanding to integer constant expressions shall additionally be suitable for use in **#if** preprocessing directives.
3. Any definition of a function-like macro in a header shall expand to code that evaluates each of its arguments exactly once, fully protected by parentheses where necessary, so that it is generally safe to use arbitrary expressions as arguments.
4. Any definition of a function-like macro in a header can be invoked in an expression anywhere a function with a compatible return type could be called.

2.2 The Compilation Environment

2.2.1 POSIX.1 Symbols

Certain symbols in this volume of POSIX.1-2008 are defined in headers (see XBD [Chapter 13](#), on page 219). Some of those headers could also define symbols other than those defined by POSIX.1-2008, potentially conflicting with symbols used by the application. Also, POSIX.1-2008 defines symbols that are not permitted by other standards to appear in those headers without some control on the visibility of those symbols.

Symbols called “feature test macros” are used to control the visibility of symbols that might be included in a header. Implementations, future versions of this standard, and other standards may define additional feature test macros.

In the compilation of an application that **#defines** a feature test macro specified by POSIX.1-2008, no header defined by POSIX.1-2008 shall be included prior to the definition of the feature test macro. This restriction also applies to any implementation-provided header in which these feature test macros are used. If the definition of the macro does not precede the **#include**, the result is undefined.

Feature test macros shall begin with the <underscore> character (`'_'`).

2.2.1.1 The `_POSIX_C_SOURCE` Feature Test Macro

A POSIX-conforming application shall ensure that the feature test macro `_POSIX_C_SOURCE` is defined before inclusion of any header.

When an application includes a header described by POSIX.1-2008, and when this feature test macro is defined to have the value 200809L:

1. All symbols required by POSIX.1-2008 to appear when the header is included shall be made visible.

2. Symbols that are explicitly permitted, but not required, by POSIX.1-2008 to appear in that header (including those in reserved name spaces) may be made visible.
3. Additional symbols not required or explicitly permitted by POSIX.1-2008 to be in that header shall not be made visible, except when enabled by another feature test macro.

Identifiers in POSIX.1-2008 may only be undefined using the **#undef** directive as described in [Section 2.1](#) (on page 471) or [Section 2.2.2](#). These **#undef** directives shall follow all **#include** directives of any header in POSIX.1-2008.

Note: The POSIX.1-1990 standard specified a macro called `_POSIX_SOURCE`. This has been superseded by `_POSIX_C_SOURCE`.

2.2.1.2 The `_XOPEN_SOURCE` Feature Test Macro

XSI An XSI-conforming application shall ensure that the feature test macro `_XOPEN_SOURCE` is defined with the value 700 before inclusion of any header. This is needed to enable the functionality described in [Section 2.2.1.1](#) (on page 472) and to ensure that the XSI option is enabled.

Since this volume of POSIX.1-2008 is aligned with the ISO C standard, and since all functionality enabled by `_POSIX_C_SOURCE` set equal to 200809L is enabled by `_XOPEN_SOURCE` set equal to 700, there should be no need to define `_POSIX_C_SOURCE` if `_XOPEN_SOURCE` is so defined. Therefore, if `_XOPEN_SOURCE` is set equal to 700 and `_POSIX_C_SOURCE` is set equal to 200809L, the behavior is the same as if only `_XOPEN_SOURCE` is defined and set equal to 700. However, should `_POSIX_C_SOURCE` be set to a value greater than 200809L, the behavior is unspecified.

If `_XOPEN_SOURCE` is defined with the value 700 and `_POSIX_C_SOURCE` is undefined before inclusion of any header, then the header may define the `_POSIX_C_SOURCE` macro with the value 200809L.

2.2.2 The Name Space

XSI All identifiers in this volume of POSIX.1-2008, except *environ*, are defined in at least one of the headers, as shown in XBD [Chapter 13](#) (on page 219). When `_XOPEN_SOURCE` or `_POSIX_C_SOURCE` is defined, each header defines or declares some identifiers, potentially conflicting with identifiers used by the application. The set of identifiers visible to the application consists of precisely those identifiers from the header pages of the included headers, as well as additional identifiers reserved for the implementation. In addition, some headers may make visible identifiers from other headers as indicated on the relevant header pages.

Implementations may also add members to a structure or union without controlling the visibility of those members with a feature test macro, as long as a user-defined macro with the same name cannot interfere with the correct interpretation of the program. The identifiers reserved for use by the implementation are described below:

1. Each identifier with external linkage described in the header section is reserved for use as an identifier with external linkage if the header is included.
2. Each macro described in the header section is reserved for any use if the header is included.
3. Each identifier with file scope described in the header section is reserved for use as an identifier with file scope in the same name space if the header is included.

The prefixes `posix_`, `POSIX_`, and `_POSIX_` are reserved for use by POSIX.1-2008 and other

16062 POSIX standards. Implementations may add symbols to the headers shown in the following
16063 table, provided the identifiers for those symbols either:

- 16064 1. Begin with the corresponding reserved prefixes in the table, or
- 16065 2. Have one of the corresponding complete names in the table, or
- 16066 3. End in the string indicated as a reserved suffix in the table and do not use the reserved
16067 prefixes `posix_`, `POSIX_`, or `_POSIX_`, as long as the reserved suffix is in that part of the
16068 name considered significant by the implementation.

16069 Symbols that use the reserved prefix `_POSIX_` may be made visible by implementations in any
16070 header defined by POSIX.1-2008.

16071				
16072		Header	Prefix	Suffix
16073		<aio.h>	aio_, lio_, AIO_, LIO_	
16074		<arpa/inet.h>	inet_	
16075		<ctype.h>	to[a-z], is[a-z]	
16076		<dlfcn.h>	RTLD_	
16077		<dirent.h>	d_	
16078		<fcntl.h>	l_	
16079	XSI	<fmtmsg.h>	MM_	
16080		<fnmatch.h>	FNM_	
16081	XSI	<ftw.h>	FTW	
16082		<glob.h>	gl_, GLOB_	
16083		<grp.h>	gr_	
16084		<limits.h>		_MAX, _MIN
16085	MSG	<mqueue.h>	mq_, MQ_	
16086	XSI	<ndbm.h>	dbm_, DBM_	
16087		<netdb.h>	ai_, h_, n_, p_, s_	
16088		<net/if.h>	if_, IF_	
16089		<netinet/in.h>	in_, ip_, s_, sin_, INADDR_, IPPROTO_	
16090	IP6		in6_, s6_, sin6_, IPV6_	
16091		<netinet/tcp.h>	TCP_	
16092		<nl_types.h>	NL_	
16093		<poll.h>	pd_, ph_, ps_, POLL	
16094		<pthread.h>	pthread_, PTHREAD_	
16095		<pwd.h>	pw_	
16096		<regex.h>	re_, rm_, REG_	
16097		<sched.h>	sched_, SCHED_	
16098		<semaphore.h>	sem_, SEM_	
16099		<signal.h>	sa_, si_, sigev_, sival_, uc_, BUS_, CLD_, FPE_, ILL_, SA_, SEGV_, SI_, SIGEV_	
16100			ss_, sv_, SS_, TRAP_, POLL_	
16101	XSI			
16102	OB XSR	<stropts.h>	bi_, ic_, l_, sl_, str_, FLUSH[A-Z], I_, S_, SND[A-Z]	
16103				
16104				
16105		<stdint.h>		int[0-9a-z-]*_t, uint[0-9a-z-]*_t
16106				
16107		<stdlib.h>	str[a-z]	
16108		<string.h>	str[a-z], mem[a-z], wcs[a-z]	
16109	XSI	<sys/ipc.h>	ipc_, IPC_	key, pad, seq
16110		<sys/mman.h>	shm_, MAP_, MCL_, MS_, PROT_	
16111				
16112	XSI	<sys/msg.h>	msg, MSG_[A-Z]	msg
16113	XSI	<sys/resource.h>	rlim_, ru_, PRIO_, RLIMIT_, RUSAGE_	
16114		<sys/select.h>	fd_, fds_, FD_	

		Header	Prefix	Suffix	Complete Name
16115					
16116					
16117	XSI	<sys/sem.h>	sem, SEM_		sem
16118	XSI	<sys/shm.h>	shm, SHM[A-Z], SHM_[A-Z]		
16119		<sys/socket.h>	cmsg_, if_, ifc_, ifra_, ifru_, infu_, l_, msg_, sa_, ss_, AF_, MSG_, PF_, SCM_, SHUT_, SO		
16120					
16121	XSI				
16122					
16123		<sys/stat.h>	st_		
16124		<sys/statvfs.h>	f_, ST_		
16125	XSI	<sys/time.h>	it_, tv_, ITIMER_		
16126		<sys/times.h>	tms_		
16127	XSI	<sys/uio.h>	iov_		UIO_MAXIOV
16128		<sys/un.h>	sun_		
16129		<sys/utsname.h>	uts_		
16130		<sys/wait.h>	P_, W[A-Z]		
16131	XSI	<syslog.h>	LOG_		
16132		<termios.h>	c_, B[0-9], TC		
16133		<time.h>	tm_ clock_, it_, timer_, tv_, CLOCK_, TIMER_		
16134					
16135					
16136	XSI	<ulimit.h>	UL_		
16137		<utime.h>	utim_		
16138	XSI	<utmpx.h>	ut_	_LVL, _PROCESS, _TIME	
16139					
16140		<wchar.h>	wcs[a-z]		
16141		<wctype.h>	is[a-z], to[a-z]		
16142		<wordexp.h>	we_, WRDE_		
16143		ANY header		_t	

Note: The notation [A–Z] indicates any uppercase letter in the portable character set. The notation [a–z] indicates any lowercase letter in the portable character set. Commas and spaces in the lists of prefixes and complete names in the above table are not part of any prefix or complete name.

If any header in the following table is included, macros with the prefixes shown may be defined. After the last inclusion of a given header, an application may use identifiers with the corresponding prefixes for its own purpose, provided their use is preceded by a **#undef** of the corresponding macro.

Header	Prefix
<errno.h>	E[0-9], E[A-Z]
<fcntl.h>	F_, O_
<fenv.h>	FE_[A-Z]
<inttypes.h>	PRI[Xa-z], SCN[Xa-z]
<locale.h>	LC_[A-Z]
<math.h>	FP_[A-Z]
<netinet/in.h>	IMPLINK_, IN_, IP_, IPPORT_, SOCK_,
	IN6_
<signal.h>	SIG_, SIG[A-Z],
	SV_
<stdio.h>	SEEK_
<stropts.h>	M_, MUXID_R[A-Z], STR
<sys/resource.h>	RLIM_
<sys/socket.h>	CMSG_
<sys/stat.h>	S_
<sys/uio.h>	IOV_
<termios.h>	I, O, V (See below.)
<unistd.h>	SEEK_

The following are used to reserve complete names for the **<stdint.h>** header:

```

INT[0-9A-Za-z-]*_MIN
INT[0-9A-Za-z-]*_MAX
INT[0-9A-Za-z-]*_C
UINT[0-9A-Za-z-]*_MIN
UINT[0-9A-Za-z-]*_MAX
UINT[0-9A-Za-z-]*_C

```

Note: The notation [0–9] indicates any digit. The notation [A–Z] indicates any uppercase letter in the portable character set. The notation [0–9a–z–_] indicates any digit, any lowercase letter in the portable character set, or <underscore>.

The following reserved names are used as exact matches for **<termios.h>**:

```

CBAUD      EXTB      VDSUSP
DEFECHO    FLUSHO    VLNEXT
ECHOCTL    LOBLK     VREPRINT
ECHOKE     PENDIN    VSTATUS
ECHOPRT    SWTCH     VWERASE
EXTA       VDISCARD

```

- 16187 The following identifiers are reserved regardless of the inclusion of headers:
- 16188 1. With the exception of identifiers beginning with the prefix `_POSIX_`, all identifiers that
- 16189 begin with an `<underscore>` and either an uppercase letter or another `<underscore>` are
- 16190 always reserved for any use by the implementation.
- 16191 2. All identifiers that begin with an `<underscore>` are always reserved for use as identifiers
- 16192 with file scope in both the ordinary identifier and tag name spaces.
- 16193 3. All identifiers in the table below are reserved for use as identifiers with external linkage.
- 16194 Some of these identifiers do not appear in this volume of POSIX.1-2008, but are reserved for
- 16195 future use by the ISO C standard.
- 16196 4. All functions and external identifiers defined in XBD [Chapter 13](#) (on page 219) are reserved
- 16197 for use as identifiers with external linkage.
- 16198 5. All the identifiers defined in this volume of POSIX.1-2008 that have external linkage are
- 16199 always reserved for use as identifiers with external linkage.
- 16200 **Note:** The notation `[a–z]` indicates any lowercase letter in the portable character set. The notation `' * '`
- 16201 indicates any combination of digits, letters in the portable character set, or `<underscore>`.
- 16202 No other identifiers are reserved.

16203	_Exit	casinh	clog10	erfcl	fminl
16204	abort	casinhf	clog10f	erff	fmod
16205	abs	casinhl	clog10l	erfl	fmodf
16206	acos	casinl	clog1p	errno	fmodl
16207	acosh	catan	clog1pf	exit	fopen
16208	acoshf	catanf	clog1pl	exp	fprintf
16209	acoshf	catanh	clog2	exp2	fputc
16210	acoshl	catanh	clog2f	exp2f	fputs
16211	acosl	catanhf	clog2l	exp2l	fputwc
16212	acosl	catanhf	clogf	expf	fputws
16213	asctime	catanhl	clogl	expl	fread
16214	asin	catanhl	conj	expm1	free
16215	asinf	catanl	conjf	expm1f	freopen
16216	asinh	cbrt	conjl	expm1l	frexp
16217	asinhf	cbrtf	copysign	fabs	frexpf
16218	asinhf	cbrtl	copysignf	fabsf	frexpl
16219	asinl	ccos	copysignl	fabsl	fscanf
16220	asinl	ccosf	cos	fclose	fseek
16221	atan	ccosh	cosf	fdim	fsetpos
16222	atan2	ccoshf	cosh	fdimf	ftell
16223	atan2f	ccoshl	coshf	fdiml	fwide
16224	atan2l	ccosl	coshl	feclearexcept	fwprintf
16225	atanf	ceil	cosl	fegetenv	fwrite
16226	atanf	ceilf	cpow	fegetexceptflag	fwscanf
16227	atanh	ceilf	cpowf	fegetround	getc
16228	atanh	ceill	cpowl	feholdexcept	getchar
16229	atanhf	ceill	cproj	feof	getenv
16230	atanhl	cerf	cprojf	feraiseexcept	gets
16231	atanl	cerfc	cprojl	ferror	getwc
16232	atanl	cerfcf	creal	fesetenv	getwchar
16233	atexit	cerfcl	crealf	fesetexceptflag	gmtime
16234	atof	cerff	creall	fesetround	hypotf
16235	atoi	cerfl	csin	fetestexcept	hypotl
16236	atol	cexmp1	csinf	feupdateenv	ilogb
16237	atoll	cexmp1f	csinh	fflush	ilogbf
16238	bsearch	cexmp1l	csinhf	fgetc	ilogbl
16239	cabs	cexp	csinhl	fgetpos	imaxabs
16240	cabsf	cexp2	csinl	fgets	imaxdiv
16241	cabsf	cexp2f	csqrt	fgetwc	is[a-z]*
16242	cacos	cexp2l	csqrtf	fgetws	isblank
16243	cacosf	cexpf	csqrtl	floor	iswblank
16244	cacosh	cexpl	ctan	floorf	labs
16245	cacoshf	cimag	ctanf	floorl	ldexp
16246	cacoshl	cimagf	ctanl	fma	ldexpf
16247	cacosl	cimagl	ctgamma	fmaf	ldexpl
16248	calloc	clearerr	ctgammaf	fmal	ldiv
16249	carg	clgamma	ctgammaf	fmax	ldiv
16250	cargf	clgammaf	ltime	fmaxf	lgammaf
16251	cargl	clgammaf	difftime	fmaxl	lgammal
16252	casin	clock	div	fmin	llabs
16253	casinf	clog	erfcf	fminf	llrint

16254	llrintf	mbsinit	realloc	sprintf	ungetwc
16255	llrintl	mbsrtowcs	remainderf	sqrt	va_end
16256	llround	mbstowcs	remainderl	sqrtf	vfprintf
16257	llroundf	mbtowc	remove	sqrtrl	vfscanf
16258	llroundl	mem[a-z]*	remquo	srand	vfwprintf
16259	localeconv	mktime	remquof	sscanf	vfwscanf
16260	localtime	modf	remquol	str[a-z]*	vprintf
16261	log	modff	rename	strtof	vscanf
16262	log10	modfl	rewind	strtoimax	vsprintf
16263	log10f	nan	rint	strtold	vsscanf
16264	log10l	nanf	rintf	strtoll	vswprintf
16265	log1p	nanl	rintl	strtoull	vswscanf
16266	log1pf	nearbyint	round	strtoumax	vwprintf
16267	log1pl	nearbyintf	roundf	swprintf	vwscanf
16268	log2	nearbyintl	roundl	swscanf	wcrtomb
16269	log2f	nextafterf	scalbln	system	wcs[a-z]*
16270	log2l	nextafterl	scalblnf	tan	wcstof
16271	logb	nexttoward	scalblnl	tanf	wcstoimax
16272	logbf	nexttowardf	scalbn	tanh	wcstold
16273	logbl	nexttowardl	scalbnf	tanhf	wcstoll
16274	logf	perror	scalbnl	tanhf	wcstoull
16275	logl	pow	scanf	tanl	wcstoumax
16276	longjmp	powf	setbuf	tgamma	wctob
16277	lrint	powl	setjmp	tgammaf	wctomb
16278	lrintf	printf	setlocale	tgammaf	wctrans
16279	lrintl	putc	setvbuf	time	wctype
16280	lround	putchar	signal	tmpfile	wcwidth
16281	lroundf	puts	sin	tmpnam	wmem[a-z]*
16282	lroundl	putwc	sinf	to[a-z]*	wprintf
16283	malloc	putwchar	sinh	trunc	wscanf
16284	mblen	qsort	sinhf	truncf	
16285	mbrlen	raise	sinhl	truncl	
16286	mbrtowc	rand	sinl	ungetc	

Applications shall not declare or define identifiers with the same name as an identifier reserved in the same context. Since macro names are replaced whenever found, independent of scope and name space, macro names matching any of the reserved identifier names shall not be defined by an application if any associated header is included.

Except that the effect of each inclusion of **<assert.h>** depends on the definition of **NDEBUG**, headers may be included in any order, and each may be included more than once in a given scope, with no difference in effect from that of being included only once.

If used, the application shall ensure that a header is included outside of any external declaration or definition, and it shall be first included before the first reference to any type or macro it defines, or to any function or object it declares. However, if an identifier is declared or defined in more than one header, the second and subsequent associated headers may be included after the initial reference to the identifier. Prior to the inclusion of a header, the application shall not define any macros with names lexically identical to symbols defined by that header.

2.3 Error Numbers

Most functions can provide an error number. The means by which each function provides its error numbers is specified in its description.

Some functions provide the error number in a variable accessed through the symbol *errno*, defined by including the `<errno.h>` header. The value of *errno* should only be examined when it is indicated to be valid by a function's return value. No function in this volume of POSIX.1-2008 shall set *errno* to zero. For each thread of a process, the value of *errno* shall not be affected by function calls or assignments to *errno* by other threads.

Some functions return an error number directly as the function value. These functions return a value of zero to indicate success.

If more than one error occurs in processing a function call, any one of the possible errors may be returned, as the order of detection is undefined.

Implementations may support additional errors not included in this list, may generate errors included in this list under circumstances other than those described here, or may contain extensions or limitations that prevent some errors from occurring.

The ERRORS section on each reference page specifies which error conditions shall be detected by all implementations ("shall fail") and which may be optionally detected by an implementation ("may fail"). If no error condition is detected, the action requested shall be successful. If an error condition is detected, the action requested may have been partially performed, unless otherwise stated.

Implementations may generate error numbers listed here under circumstances other than those described, if and only if all those error conditions can always be treated identically to the error conditions as described in this volume of POSIX.1-2008. Implementations shall not generate a different error number from one required by this volume of POSIX.1-2008 for an error condition described in this volume of POSIX.1-2008, but may generate additional errors unless explicitly disallowed for a particular function.

Each implementation shall document, in the conformance document, situations in which each of the optional conditions defined in POSIX.1-2008 is detected. The conformance document may also contain statements that one or more of the optional error conditions are not detected.

Certain threads-related functions are not allowed to return an error code of [EINTR]. Where this applies it is stated in the ERRORS section on the individual function pages.

The following macro names identify the possible error numbers, in the context of the functions specifically defined in this volume of POSIX.1-2008; these general descriptions are more precisely defined in the ERRORS sections of the functions that return them. Only these macro names should be used in programs, since the actual value of the error number is unspecified. All values listed in this section shall be unique, except as noted below. The values for all these macros shall be found in the `<errno.h>` header defined in the Base Definitions volume of POSIX.1-2008. The actual values are unspecified by this volume of POSIX.1-2008.

[E2BIG]

Argument list too long. The sum of the number of bytes used by the new process image's argument list and environment list is greater than the system-imposed limit of {ARG_MAX} bytes.

or:

Lack of space in an output buffer.

or:

16345		Argument is greater than the system-imposed maximum.
16346		[EACCES]
16347		Permission denied. An attempt was made to access a file in a way forbidden by its file
16348		access permissions.
16349		[EADDRINUSE]
16350		Address in use. The specified address is in use.
16351		[EADDRNOTAVAIL]
16352		Address not available. The specified address is not available from the local system.
16353		[EAFNOSUPPORT]
16354		Address family not supported. The implementation does not support the specified address
16355		family, or the specified address is not a valid address for the address family of the specified
16356		socket.
16357		[EAGAIN]
16358		Resource temporarily unavailable. This is a temporary condition and later calls to the same
16359		routine may complete normally.
16360		[EALREADY]
16361		Connection already in progress. A connection request is already in progress for the specified
16362		socket.
16363		[EBADF]
16364		Bad file descriptor. A file descriptor argument is out of range, refers to no open file, or a
16365		read (write) request is made to a file that is only open for writing (reading).
16366		[EBADMSG]
16367	OB XSR	Bad message. During a <i>read()</i> , <i>getmsg()</i> , <i>getpmsg()</i> , or <i>ioctl()</i> I_RECVFD request to a
16368		STREAMS device, a message arrived at the head of the STREAM that is inappropriate for
16369		the function receiving the message.
16370		<i>read()</i> Message waiting to be read on a STREAM is not a data message.
16371		<i>getmsg()</i> or <i>getpmsg()</i>
16372		A file descriptor was received instead of a control message.
16373		<i>ioctl()</i> Control or data information was received instead of a file descriptor when
16374		I_RECVFD was specified.
16375		or:
16376		Bad Message. The implementation has detected a corrupted message.
16377		[EBUSY]
16378		Resource busy. An attempt was made to make use of a system resource that is not currently
16379		available, as it is being used by another process in a manner that would have conflicted
16380		with the request being made by this process.
16381		[ECANCELED]
16382		Operation canceled. The associated asynchronous operation was canceled before
16383		completion.
16384		[ECHILD]
16385		No child process. A <i>wait()</i> , <i>waitid()</i> , or <i>waitpid()</i> function was executed by a process that
16386		had no existing or unwaited-for child process.

16387	[ECONNABORTED]
16388	Connection aborted. The connection has been aborted.
16389	[ECONNREFUSED]
16390	Connection refused. An attempt to connect to a socket was refused because there was no
16391	process listening or because the queue of connection requests was full and the underlying
16392	protocol does not support retransmissions.
16393	[ECONNRESET]
16394	Connection reset. The connection was forcibly closed by the peer.
16395	[EDEADLK]
16396	Resource deadlock would occur. An attempt was made to lock a system resource that would
16397	have resulted in a deadlock situation.
16398	[EDESTADDRREQ]
16399	Destination address required. No bind address was established.
16400	[EDOM]
16401	Domain error. An input argument is outside the defined domain of the mathematical
16402	function (defined in the ISO C standard).
16403	[EDQUOT]
16404	Reserved.
16405	[EEXIST]
16406	File exists. An existing file was mentioned in an inappropriate context; for example, as a
16407	new link name in the <i>link()</i> function.
16408	[EFAULT]
16409	Bad address. The system detected an invalid address in attempting to use an argument of a
16410	call. The reliable detection of this error cannot be guaranteed, and when not detected may
16411	result in the generation of a signal, indicating an address violation, which is sent to the
16412	process.
16413	[EFBIG]
16414	File too large. The size of a file would exceed the maximum file size of an implementation
16415	or offset maximum established in the corresponding file description.
16416	[EHOSTUNREACH]
16417	Host is unreachable. The destination host cannot be reached (probably because the host is
16418	down or a remote router cannot reach it).
16419	[EIDRM]
16420	Identifier removed. Returned during XSI interprocess communication if an identifier has
16421	been removed from the system.
16422	[EILSEQ]
16423	Illegal byte sequence. A wide-character code has been detected that does not correspond to
16424	a valid character, or a byte sequence does not form a valid wide-character code (defined in
16425	the ISO C standard).
16426	[EINPROGRESS]
16427	Operation in progress. This code is used to indicate that an asynchronous operation has not
16428	yet completed.
16429	or:
16430	O_NONBLOCK is set for the socket file descriptor and the connection cannot be
16431	immediately established.

16432	[EINTR]	
16433		Interrupted function call. An asynchronous signal was caught by the process during the
16434		execution of an interruptible function. If the signal handler performs a normal return, the
16435		interrupted function call may return this condition (see the Base Definitions volume of
16436		POSIX.1-2008, <signal.h>).
16437	[EINVAL]	
16438		Invalid argument. Some invalid argument was supplied; for example, specifying an
16439		undefined signal in a <i>signal()</i> function or a <i>kill()</i> function.
16440	[EIO]	
16441		Input/output error. Some physical input or output error has occurred. This error may be
16442		reported on a subsequent operation on the same file descriptor. Any other error-causing
16443		operation on the same file descriptor may cause the [EIO] error indication to be lost.
16444	[EISCONN]	
16445		Socket is connected. The specified socket is already connected.
16446	[EISDIR]	
16447		Is a directory. An attempt was made to open a directory with write mode specified.
16448	[ELOOP]	
16449		Symbolic link loop. A loop exists in symbolic links encountered during pathname
16450		resolution. This error may also be returned if more than {SYMLOOP_MAX} symbolic links
16451		are encountered during pathname resolution.
16452	[EMFILE]	
16453		File descriptor value too large or too many open streams. An attempt was made to open a
16454	XSI	file descriptor with a value greater than or equal to {OPEN_MAX}, or greater than or equal
16455		to the soft limit RLIMIT_NOFILE for the process (if smaller than {OPEN_MAX}); or an
16456		attempt was made to open more than the maximum number of streams allowed in the
16457		process.
16458	[EMLINK]	
16459		Too many links. An attempt was made to have the link count of a single file exceed
16460		{LINK_MAX}.
16461	[EMSGSIZE]	
16462		Message too large. A message sent on a transport provider was larger than an internal
16463		message buffer or some other network limit.
16464		or:
16465		Inappropriate message buffer length.
16466	[EMULTIHOP]	
16467		Reserved.
16468	[ENAMETOOLONG]	
16469		Filename too long. The length of a pathname exceeds {PATH_MAX} and the
16470		implementation considers this to be an error, or a pathname component is longer than
16471		{NAME_MAX}. This error may also occur when pathname substitution, as a result of
16472		encountering a symbolic link during pathname resolution, results in a pathname string the
16473		size of which exceeds {PATH_MAX}.
16474	[ENETDOWN]	
16475		Network is down. The local network interface used to reach the destination is down.

16476		[ENETRESET]
16477		The connection was aborted by the network.
16478		[ENETUNREACH]
16479		Network unreachable. No route to the network is present.
16480		[ENFILE]
16481		Too many files open in system. Too many files are currently open in the system. The system
16482		has reached its predefined limit for simultaneously open files and temporarily cannot
16483		accept requests to open another one.
16484		[ENOBUFS]
16485		No buffer space available. Insufficient buffer resources were available in the system to
16486		perform the socket operation.
16487	OB XSR	[ENODATA]
16488		No message available. No message is available on the STREAM head read queue.
16489		[ENODEV]
16490		No such device. An attempt was made to apply an inappropriate function to a device; for
16491		example, trying to read a write-only device such as a printer.
16492		[ENOENT]
16493		No such file or directory. A component of a specified pathname does not exist, or the
16494		pathname is an empty string.
16495		[ENOEXEC]
16496		Executable file format error. A request is made to execute a file that, although it has
16497		appropriate privileges, is not in the format required by the implementation for executable
16498		files.
16499		[ENOLCK]
16500		No locks available. A system-imposed limit on the number of simultaneous file and record
16501		locks has been reached and no more are currently available.
16502		[ENOLINK]
16503		Reserved.
16504		[ENOMEM]
16505		Not enough space. The new process image requires more memory than is allowed by the
16506		hardware or system-imposed memory management constraints.
16507		[ENOMSG]
16508		No message of the desired type. The message queue does not contain a message of the
16509		required type during XSI interprocess communication.
16510		[ENOPROTOPT]
16511		Protocol not available. The protocol option specified to <i>setsockopt()</i> is not supported by the
16512		implementation.
16513		[ENOSPC]
16514		No space left on a device. During the <i>write()</i> function on a regular file or when extending a
16515		directory, there is no free space left on the device.
16516	OB XSR	[ENOSR]
16517		No STREAM resources. Insufficient STREAMS memory resources are available to perform a
16518		STREAMS-related function. This is a temporary condition; it may be recovered from if other
16519		processes release resources.

16520	OB XSR	[ENOSTR]	
16521		Not a STREAM. A STREAM function was attempted on a file descriptor that was not	
16522		associated with a STREAMS device.	
16523		[ENOSYS]	
16524		Function not implemented. An attempt was made to use a function that is not available in	
16525		this implementation.	
16526		[ENOTCONN]	
16527		Socket not connected. The socket is not connected.	
16528		[ENOTDIR]	
16529		Not a directory. A component of the specified pathname exists, but it is not a directory,	
16530		when a directory was expected; or an attempt was made to create a non-directory file, and	
16531		the specified pathname contains at least one non- <code><slash></code> character and ends with one or	
16532		more trailing <code><slash></code> characters.	
16533		[ENOTEMPTY]	
16534		Directory not empty. A directory other than an empty directory was supplied when an	
16535		empty directory was expected.	
16536		[ENOTRECOVERABLE]	
16537		State not recoverable. The state protected by a robust mutex is not recoverable.	
16538		[ENOTSOCK]	
16539		Not a socket. The file descriptor does not refer to a socket.	
16540		[ENOTSUP]	
16541		Not supported. The implementation does not support the requested feature or value.	
16542		[ENOTTY]	
16543		Inappropriate I/O control operation. A control function has been attempted for a file or	
16544		special file for which the operation is inappropriate.	
16545		[ENXIO]	
16546		No such device or address. Input or output on a special file refers to a device that does not	
16547		exist, or makes a request beyond the capabilities of the device. It may also occur when, for	
16548		example, a tape drive is not on-line.	
16549		[EOPNOTSUPP]	
16550		Operation not supported on socket. The type of socket (address family or protocol) does not	
16551		support the requested operation. A conforming implementation may assign the same values	
16552		for [EOPNOTSUPP] and [ENOTSUP].	
16553		[EOVERFLOW]	
16554		Value too large to be stored in data type. An operation was attempted which would	
16555		generate a value that is outside the range of values that can be represented in the relevant	
16556		data type or that are allowed for a given data item.	
16557		[EOWNERDEAD]	
16558		Previous owner died. The owner of a robust mutex terminated while holding the mutex	
16559		lock.	
16560		[EPERM]	
16561		Operation not permitted. An attempt was made to perform an operation limited to	
16562		processes with appropriate privileges or to the owner of a file or other resource.	
16563		[EPIPE]	
16564		Broken pipe. A write was attempted on a socket, pipe, or FIFO for which there is no process	
16565		to read the data.	

16566		[EPROTO]
16567		Protocol error. Some protocol error occurred. This error is device-specific, but is generally
16568		not related to a hardware failure.
16569		[EPROTONOSUPPORT]
16570		Protocol not supported. The protocol is not supported by the address family, or the protocol
16571		is not supported by the implementation.
16572		[EPROTOTYPE]
16573		Protocol wrong type for socket. The socket type is not supported by the protocol.
16574		[ERANGE]
16575		Result too large or too small. The result of the function is too large (overflow) or too small
16576		(underflow) to be represented in the available space (defined in the ISO C standard).
16577		[EROFS]
16578		Read-only file system. An attempt was made to modify a file or directory on a file system
16579		that is read-only.
16580		[ESPIPE]
16581		Invalid seek. An attempt was made to access the file offset associated with a pipe or FIFO.
16582		[ESRCH]
16583		No such process. No process can be found corresponding to that specified by the given
16584		process ID.
16585		[ESTALE]
16586		Reserved.
16587	OB XSR	[ETIME]
16588		STREAM <i>ioctl()</i> timeout. The timer set for a STREAMS <i>ioctl()</i> call has expired. The cause of
16589		this error is device-specific and could indicate either a hardware or software failure, or a
16590		timeout value that is too short for the specific operation. The status of the <i>ioctl()</i> operation is
16591		unspecified.
16592		[ETIMEDOUT]
16593		Connection timed out. The connection to a remote machine has timed out. If the connection
16594		timed out during execution of the function that reported this error (as opposed to timing
16595		out prior to the function being called), it is unspecified whether the function has completed
16596		some or all of the documented behavior associated with a successful completion of the
16597		function.
16598		or:
16599		Operation timed out. The time limit associated with the operation was exceeded before the
16600		operation completed.
16601		[ETXTBSY]
16602		Text file busy. An attempt was made to execute a pure-procedure program that is currently
16603		open for writing, or an attempt has been made to open for writing a pure-procedure
16604		program that is being executed.
16605		[EWOULDBLOCK]
16606		Operation would block. An operation on a socket marked as non-blocking has encountered
16607		a situation such as no data available that otherwise would have caused the function to
16608		suspend execution.
16609		A conforming implementation may assign the same values for [EWOULDBLOCK] and
16610		[EAGAIN].

16611 [EXDEV]
 16612 Improper link. A link to a file on another file system was attempted.

16613 2.3.1 Additional Error Numbers

16614 Additional implementation-defined error numbers may be defined in `<errno.h>`.

16615 2.4 Signal Concepts

16616 2.4.1 Signal Generation and Delivery

16617 A signal is said to be “generated” for (or sent to) a process or thread when the event that causes
 16618 the signal first occurs. Examples of such events include detection of hardware faults, timer
 16619 expiration, signals generated via the **sigevent** structure and terminal activity, as well as
 16620 invocations of the `kill()` and `sigqueue()` functions. In some circumstances, the same event
 16621 generates signals for multiple processes.

16622 At the time of generation, a determination shall be made whether the signal has been generated
 16623 for the process or for a specific thread within the process. Signals which are generated by some
 16624 action attributable to a particular thread, such as a hardware fault, shall be generated for the
 16625 thread that caused the signal to be generated. Signals that are generated in association with a
 16626 process ID or process group ID or an asynchronous event, such as terminal activity, shall be
 16627 generated for the process.

16628 Each process has an action to be taken in response to each signal defined by the system (see
 16629 [Section 2.4.3](#), on page 490). A signal is said to be “delivered” to a process when the appropriate
 16630 action for the process and signal is taken. A signal is said to be “accepted” by a process when the
 16631 signal is selected and returned by one of the `sigwait()` functions.

16632 During the time between the generation of a signal and its delivery or acceptance, the signal is
 16633 said to be “pending”. Ordinarily, this interval cannot be detected by an application. However, a
 16634 signal can be “blocked” from delivery to a thread. If the action associated with a blocked signal
 16635 is anything other than to ignore the signal, and if that signal is generated for the thread, the
 16636 signal shall remain pending until it is unblocked, it is accepted when it is selected and returned
 16637 by a call to the `sigwait()` function, or the action associated with it is set to ignore the signal.
 16638 Signals generated for the process shall be delivered to exactly one of those threads within the
 16639 process which is in a call to a `sigwait()` function selecting that signal or has not blocked delivery
 16640 of the signal. If there are no threads in a call to a `sigwait()` function selecting that signal, and if all
 16641 threads within the process block delivery of the signal, the signal shall remain pending on the
 16642 process until a thread calls a `sigwait()` function selecting that signal, a thread unblocks delivery
 16643 of the signal, or the action associated with the signal is set to ignore the signal. If the action
 16644 associated with a blocked signal is to ignore the signal and if that signal is generated for the
 16645 process, it is unspecified whether the signal is discarded immediately upon generation or
 16646 remains pending.

16647 Each thread has a “signal mask” that defines the set of signals currently blocked from delivery
 16648 to it. The signal mask for a thread shall be initialized from that of its parent or creating thread,
 16649 or from the corresponding thread in the parent process if the thread was created as the result of a
 16650 call to `fork()`. The `pthread_sigmask()`, `sigaction()`, `sigprocmask()`, and `sigsuspend()` functions control
 16651 the manipulation of the signal mask.

The determination of which action is taken in response to a signal is made at the time the signal is delivered, allowing for any changes since the time of generation. This determination is independent of the means by which the signal was originally generated. If a subsequent occurrence of a pending signal is generated, it is implementation-defined as to whether the signal is delivered or accepted more than once in circumstances other than those in which queuing is required. The order in which multiple, simultaneously pending signals outside the range SIGRTMIN to SIGRTMAX are delivered to or accepted by a process is unspecified.

When any stop signal (SIGSTOP, SIGTSTP, SIGTTIN, SIGTTOU) is generated for a process or thread, all pending SIGCONT signals for that process or any of the threads within that process shall be discarded. Conversely, when SIGCONT is generated for a process or thread, all pending stop signals for that process or any of the threads within that process shall be discarded. When SIGCONT is generated for a process that is stopped, the process shall be continued, even if the SIGCONT signal is ignored by the process or is blocked by all threads within the process and there are no threads in a call to a *sigwait()* function selecting SIGCONT. If SIGCONT is blocked by all threads within the process, there are no threads in a call to a *sigwait()* function selecting SIGCONT, and SIGCONT is not ignored by the process, the SIGCONT signal shall remain pending on the process until it is either unblocked by a thread or a thread calls a *sigwait()* function selecting SIGCONT, or a stop signal is generated for the process or any of the threads within the process.

An implementation shall document any condition not specified by this volume of POSIX.1-2008 under which the implementation generates signals.

2.4.2 Realtime Signal Generation and Delivery

This section describes functionality to support realtime signal generation and delivery.

Some signal-generating functions, such as high-resolution timer expiration, asynchronous I/O completion, interprocess message arrival, and the *sigqueue()* function, support the specification of an application-defined value, either explicitly as a parameter to the function or in a **sigevent** structure parameter. The **sigevent** structure is defined in **<signal.h>** and contains at least the following members:

Member Type	Member Name	Description
int	<i>sigev_notify</i>	Notification type.
int	<i>sigev_signo</i>	Signal number.
union signal	<i>sigev_value</i>	Signal value.
void(*)(union signal)	<i>sigev_notify_function</i>	Notification function.
(pthread_attr_t*)	<i>sigev_notify_attributes</i>	Notification attributes.

The *sigev_notify* member specifies the notification mechanism to use when an asynchronous event occurs. This volume of POSIX.1-2008 defines the following values for the *sigev_notify* member:

SIGEV_NONE No asynchronous notification shall be delivered when the event of interest occurs.

SIGEV_SIGNAL The signal specified in *sigev_signo* shall be generated for the process when the event of interest occurs. If the implementation supports the Realtime Signals Extension option and if the SA_SIGINFO flag is set for that signal number, then the signal shall be queued to the process and the value specified in *sigev_value* shall be the *si_value* component of the generated signal. If SA_SIGINFO is not set for that signal number, it is unspecified whether the signal is queued and what value, if any, is sent.

SIGEV_THREAD A notification function shall be called to perform notification.

An implementation may define additional notification mechanisms.

The *sigeval_signo* member specifies the signal to be generated. The *sigeval_value* member is the application-defined value to be passed to the signal-catching function at the time of the signal delivery or to be returned at signal acceptance as the *si_value* member of the **siginfo_t** structure.

The **sigval** union is defined in **<signal.h>** and contains at least the following members:

Member Type	Member Name	Description
int	<i>sival_int</i>	Integer signal value.
void*	<i>sival_ptr</i>	Pointer signal value.

The *sival_int* member shall be used when the application-defined value is of type **int**; the *sival_ptr* member shall be used when the application-defined value is a pointer.

When a signal is generated by the *sigqueue()* function or any signal-generating function that supports the specification of an application-defined value, the signal shall be marked pending and, if the SA_SIGINFO flag is set for that signal, the signal shall be queued to the process along with the application-specified signal value. Multiple occurrences of signals so generated are queued in FIFO order. It is unspecified whether signals so generated are queued when the SA_SIGINFO flag is not set for that signal.

Signals generated by the *kill()* function or other events that cause signals to occur, such as detection of hardware faults, *alarm()* timer expiration, or terminal activity, and for which the implementation does not support queuing, shall have no effect on signals already queued for the same signal number.

When multiple unblocked signals, all in the range SIGRTMIN to SIGRTMAX, are pending, the behavior shall be as if the implementation delivers the pending unblocked signal with the lowest signal number within that range. No other ordering of signal delivery is specified.

If, when a pending signal is delivered, there are additional signals queued to that signal number, the signal shall remain pending. Otherwise, the pending indication shall be reset.

Multi-threaded programs can use an alternate event notification mechanism. When a notification is processed, and the *sigeval_notify* member of the **sigevent** structure has the value SIGEV_THREAD, the function *sigeval_notify_function* is called with parameter *sigeval_value*.

The function shall be executed in an environment as if it were the *start_routine* for a newly created thread with thread attributes specified by *sigeval_notify_attributes*. If *sigeval_notify_attributes* is NULL, the behavior shall be as if the thread were created with the *detachstate* attribute set to PTHREAD_CREATE_DETACHED. Supplying an attributes structure with a *detachstate* attribute of PTHREAD_CREATE_JOINABLE results in undefined behavior. The signal mask of this thread is implementation-defined.

2.4.3 Signal Actions

There are three types of action that can be associated with a signal: SIG_DFL, SIG_IGN, or a pointer to a function. Initially, all signals shall be set to SIG_DFL or SIG_IGN prior to entry of the *main()* routine (see the *exec* functions). The actions prescribed by these values are as follows.

SIG_DFL

Signal-specific default action.

The default actions for the signals defined in this volume of POSIX.1-2008 are specified under **<signal.h>**. The default actions for the realtime signals in the range SIGRTMIN to SIGRTMAX shall be to terminate the process abnormally.

If the default action is to terminate the process abnormally, the process is terminated as if by a call to `_exit()`, except that the status made available to `wait()`, `waitid()`, and `waitpid()` indicates abnormal termination by the signal. If the default action is to terminate the process abnormally with additional actions, implementation-defined abnormal termination actions, such as creation of a core file, may also occur.

If the default action is to stop the process, the execution of that process is temporarily suspended. When a process stops, a SIGCHLD signal shall be generated for its parent process, unless the parent process has set the SA_NOCLDSTOP flag. While a process is stopped, any additional signals that are sent to the process shall not be delivered until the process is continued, except SIGKILL which always terminates the receiving process. A process that is a member of an orphaned process group shall not be allowed to stop in response to the SIGTSTP, SIGTTIN, or SIGTTOU signals. In cases where delivery of one of these signals would stop such a process, the signal shall be discarded.

If the default action is to ignore the signal, delivery of the signal shall have no effect on the process.

Setting a signal action to SIG_DFL for a signal that is pending, and whose default action is to ignore the signal (for example, SIGCHLD), shall cause the pending signal to be discarded, whether or not it is blocked. Any queued values pending shall be discarded and the resources used to queue them shall be released and returned to the system for other use.

The default action for SIGCONT is to resume execution at the point where the process was stopped, after first handling any pending unblocked signals.

When a stopped process is continued, a SIGCHLD signal may be generated for its parent process, unless the parent process has set the SA_NOCLDSTOP flag.

SIG_IGN

Ignore signal.

Delivery of the signal shall have no effect on the process. The behavior of a process is undefined after it ignores a SIGFPE, SIGILL, SIGSEGV, or SIGBUS signal that was not generated by `kill()`, `sigqueue()`, or `raise()`.

The system shall not allow the action for the signals SIGKILL or SIGSTOP to be set to SIG_IGN.

Setting a signal action to SIG_IGN for a signal that is pending shall cause the pending signal to be discarded, whether or not it is blocked.

If a process sets the action for the SIGCHLD signal to SIG_IGN, the behavior is unspecified, except as specified below.

If the action for the SIGCHLD signal is set to SIG_IGN, child processes of the calling processes shall not be transformed into zombie processes when they terminate. If the calling process subsequently waits for its children, and the process has no unwaited-for children that were transformed into zombie processes, it shall block until all of its children terminate, and `wait()`, `waitid()`, and `waitpid()` shall fail and set `errno` to [ECHILD].

Any queued values pending shall be discarded and the resources used to queue them shall be

released and made available to queue other signals.

Pointer to a Function

Catch signal.

On delivery of the signal, the receiving process is to execute the signal-catching function at the specified address. After returning from the signal-catching function, the receiving process shall resume execution at the point at which it was interrupted.

If the SA_SIGINFO flag for the signal is cleared, the signal-catching function shall be entered as a C-language function call as follows:

```
void func(int signo);
```

If the SA_SIGINFO flag for the signal is set, the signal-catching function shall be entered as a C-language function call as follows:

```
void func(int signo, siginfo_t *info, void *context);
```

where *func* is the specified signal-catching function, *signo* is the signal number of the signal being delivered, and *info* is a pointer to a **siginfo_t** structure defined in **<signal.h>** containing at least the following members:

Member Type	Member Name	Description
int	<i>si_signo</i>	Signal number.
int	<i>si_code</i>	Cause of the signal.
pid_t	<i>si_pid</i>	Sending process ID.
uid_t	<i>si_uid</i>	Real user ID of sending process.
void *	<i>si_addr</i>	Address of faulting instruction.
int	<i>si_status</i>	Exit value or signal.
union signal	<i>si_value</i>	Signal value.

The *si_signo* member shall contain the signal number. This shall be the same as the *signo* parameter. The *si_code* member shall contain a code identifying the cause of the signal. The following non-signal-specific values are defined for *si_code*:

SI_USER The signal was sent by the *kill()* function. The implementation may set *si_code* to SI_USER if the signal was sent by the *raise()* or *abort()* functions or any similar functions provided as implementation extensions.

SI_QUEUE The signal was sent by the *sigqueue()* function.

SI_TIMER The signal was generated by the expiration of a timer set by *timer_settime()*.

SI_ASYNCIO The signal was generated by the completion of an asynchronous I/O request.

MSG SI_MESGQ The signal was generated by the arrival of a message on an empty message queue.

Signal-specific values for *si_code* are also defined, as described in XBD **<signal.h>**.

If the signal was not generated by one of the functions or events listed above, *si_code* shall be set either to one of the signal-specific values described in XBD **<signal.h>**, or to an implementation-defined value that is not equal to any of the values defined above.

XSI If *si_code* is SI_USER or SI_QUEUE, or any value less than or equal to 0, then the signal was generated by a process and *si_pid* and *si_uid* shall be set to the process ID and the real user ID of the sender, respectively.

In addition, *si_addr*, *si_pid*, *si_status*, and *si_uid* shall be set for certain signal-specific values of

16823	<i>si_code</i> , as described in XBD <signal.h>.	
16824	If <i>si_code</i> is one of SI_QUEUE, SI_TIMER, SI_ASYNCIO, or SI_MESGQ, then <i>si_value</i> shall	
16825	contain the application-specified signal value. Otherwise, the contents of <i>si_value</i> are undefined.	
16826	The behavior of a process is undefined after it returns normally from a signal-catching function	
16827	for a SIGBUS, SIGFPE, SIGILL, or SIGSEGV signal that was not generated by <i>kill()</i> , <i>sigqueue()</i> ,	
16828	or <i>raise()</i> .	
16829	The system shall not allow a process to catch the signals SIGKILL and SIGSTOP.	
16830	If a process establishes a signal-catching function for the SIGCHLD signal while it has a	
16831	terminated child process for which it has not waited, it is unspecified whether a SIGCHLD	
16832	signal is generated to indicate that child process.	
16833	If the process is multi-threaded, or if the process is single-threaded and a signal handler is	
16834	executed other than as the result of:	
16835	• The process calling <i>abort()</i> , <i>raise()</i> , <i>kill()</i> , <i>pthread_kill()</i> , or <i>sigqueue()</i> to generate a signal	
16836	that is not blocked	
16837	• A pending signal being unblocked and being delivered before the call that unblocked it	
16838	returns	
16839	the behavior is undefined if the signal handler refers to any object other than <i>errno</i> with static	
16840	storage duration other than by assigning a value to an object declared as volatile sig_atomic_t ,	
16841	or if the signal handler calls any function defined in this standard other than one of the functions	
16842	listed in the following table.	
16843	The following table defines a set of functions that shall be async-signal-safe. Therefore,	
16844	applications can invoke them, without restriction, from signal-catching functions:	

16845	<code>_Exit()</code>	<code>fexecve()</code>	<code>posix_trace_event()</code>	<code>sigprocmask()</code>
16846	<code>_exit()</code>	<code>fork()</code>	<code>pselect()</code>	<code>sigqueue()</code>
16847	<code>abort()</code>	<code>fstat()</code>	<code>pthread_kill()</code>	<code>sigset()</code>
16848	<code>accept()</code>	<code>fstatat()</code>	<code>pthread_self()</code>	<code>sigsuspend()</code>
16849	<code>access()</code>	<code>fsync()</code>	<code>pthread_sigmask()</code>	<code>sleep()</code>
16850	<code>aio_error()</code>	<code>ftruncate()</code>	<code>raise()</code>	<code>socketatmark()</code>
16851	<code>aio_return()</code>	<code>futimens()</code>	<code>read()</code>	<code>socket()</code>
16852	<code>aio_suspend()</code>	<code>getegid()</code>	<code>readlink()</code>	<code>socketpair()</code>
16853	<code>alarm()</code>	<code>geteuid()</code>	<code>readlinkat()</code>	<code>stat()</code>
16854	<code>bind()</code>	<code>getgid()</code>	<code>recv()</code>	<code>symlink()</code>
16855	<code>cfgetispeed()</code>	<code>getgroups()</code>	<code>recvfrom()</code>	<code>symlinkat()</code>
16856	<code>cfgetospeed()</code>	<code>getpeername()</code>	<code>recvmsg()</code>	<code>tcdrain()</code>
16857	<code>cfsetispeed()</code>	<code>getpggrp()</code>	<code>rename()</code>	<code>tcflow()</code>
16858	<code>cfsetospeed()</code>	<code>getpid()</code>	<code>renameat()</code>	<code>tcflush()</code>
16859	<code>chdir()</code>	<code>getppid()</code>	<code>rmdir()</code>	<code>tcgetattr()</code>
16860	<code>chmod()</code>	<code>getsockname()</code>	<code>select()</code>	<code>tcgetpgrp()</code>
16861	<code>chown()</code>	<code>getsockopt()</code>	<code>sem_post()</code>	<code>tcsendbreak()</code>
16862	<code>clock_gettime()</code>	<code>getuid()</code>	<code>send()</code>	<code>tcsetattr()</code>
16863	<code>close()</code>	<code>kill()</code>	<code>sendmsg()</code>	<code>tcsetpgrp()</code>
16864	<code>connect()</code>	<code>link()</code>	<code>sendto()</code>	<code>time()</code>
16865	<code>creat()</code>	<code>linkat()</code>	<code>setgid()</code>	<code>timer_getoverrun()</code>
16866	<code>dup()</code>	<code>listen()</code>	<code>setpgid()</code>	<code>timer_gettime()</code>
16867	<code>dup2()</code>	<code>lseek()</code>	<code>setsid()</code>	<code>timer_settime()</code>
16868	<code>execl()</code>	<code>lstat()</code>	<code>setsockopt()</code>	<code>times()</code>
16869	<code>execle()</code>	<code>mkdir()</code>	<code>setuid()</code>	<code>umask()</code>
16870	<code>execv()</code>	<code>mkdirat()</code>	<code>shutdown()</code>	<code>uname()</code>
16871	<code>execve()</code>	<code>mkfifo()</code>	<code>sigaction()</code>	<code>unlink()</code>
16872	<code>faccessat()</code>	<code>mkfifoat()</code>	<code>sigaddset()</code>	<code>unlinkat()</code>
16873	<code>fchdir()</code>	<code>mknod()</code>	<code>sigdelset()</code>	<code>utime()</code>
16874	<code>fchmod()</code>	<code>mknodat()</code>	<code>sigemptyset()</code>	<code>utimensat()</code>
16875	<code>fchmodat()</code>	<code>open()</code>	<code>sigfillset()</code>	<code>utimes()</code>
16876	<code>fchown()</code>	<code>openat()</code>	<code>sigismember()</code>	<code>wait()</code>
16877	<code>fchownat()</code>	<code>pause()</code>	<code>signal()</code>	<code>waitpid()</code>
16878	<code>fcntl()</code>	<code>pipe()</code>	<code>sigpause()</code>	<code>write()</code>
16879	<code>fdatasync()</code>	<code>poll()</code>	<code>sigpending()</code>	

All functions not in the above table are considered to be unsafe with respect to signals. In the presence of signals, all functions defined by this volume of POSIX.1-2008 shall behave as defined when called from or interrupted by a signal-catching function, with a single exception: when a signal interrupts an unsafe function and the signal-catching function calls an unsafe function, the behavior is undefined.

Operations which obtain the value of *errno* and operations which assign a value to *errno* shall be async-signal-safe.

When a signal is delivered to a thread, if the action of that signal specifies termination, stop, or continue, the entire process shall be terminated, stopped, or continued, respectively.

2.4.4 Signal Effects on Other Functions

Signals affect the behavior of certain functions defined by this volume of POSIX.1-2008 if delivered to a process while it is executing such a function. If the action of the signal is to terminate the process, the process shall be terminated and the function shall not return. If the action of the signal is to stop the process, the process shall stop until continued or terminated. Generation of a SIGCONT signal for the process shall cause the process to be continued, and the original function shall continue at the point the process was stopped. If the action of the signal is to invoke a signal-catching function, the signal-catching function shall be invoked; in this case the original function is said to be “interrupted” by the signal. If the signal-catching function executes a **return** statement, the behavior of the interrupted function shall be as described individually for that function, except as noted for unsafe functions. Signals that are ignored shall not affect the behavior of any function; signals that are blocked shall not affect the behavior of any function until they are unblocked and then delivered, except as specified for the *sigpending()* and *sigwait()* functions.

2.5 Standard I/O Streams

A stream is associated with an external file (which may be a physical device) or memory buffer by “opening” a file or buffer. This may involve “creating” a new file. Creating an existing file causes its former contents to be discarded if necessary. If a file can support positioning requests (such as a disk file, as opposed to a terminal), then a “file position indicator” associated with the stream is positioned at the start (byte number 0) of the file, unless the file is opened with append mode, in which case it is implementation-defined whether the file position indicator is initially positioned at the beginning or end of the file. The file position indicator is maintained by subsequent reads, writes, and positioning requests, to facilitate an orderly progression through the file. All input takes place as if bytes were read by successive calls to *fgetc()*; all output takes place as if bytes were written by successive calls to *fputc()*.

When a stream is “unbuffered”, bytes are intended to appear from the source or at the destination as soon as possible; otherwise, bytes may be accumulated and transmitted as a block. When a stream is “fully buffered”, bytes are intended to be transmitted as a block when a buffer is filled. When a stream is “line buffered”, bytes are intended to be transmitted as a block when a <newline> byte is encountered. Furthermore, bytes are intended to be transmitted as a block when a buffer is filled, when input is requested on an unbuffered stream, or when input is requested on a line-buffered stream that requires the transmission of bytes. Support for these characteristics is implementation-defined, and may be affected via *setbuf()* and *setvbuf()*.

A file may be disassociated from a controlling stream by “closing” the file. Output streams are flushed (any unwritten buffer contents are transmitted) before the stream is disassociated from the file. The value of a pointer to a **FILE** object is unspecified after the associated file is closed (including the standard streams).

A file may be subsequently reopened, by the same or another program execution, and its contents reclaimed or modified (if it can be repositioned at its start). If the *main()* function returns to its original caller, or if the *exit()* function is called, all open files are closed (hence all output streams are flushed) before program termination. Other paths to program termination, such as calling *abort()*, need not close all files properly.

The address of the **FILE** object used to control a stream may be significant; a copy of a **FILE** object need not necessarily serve in place of the original.

At program start-up, three streams are predefined and need not be opened explicitly: *standard input* (for reading conventional input), *standard output* (for writing conventional output), and

standard error (for writing diagnostic output). When opened, the standard error stream is not fully buffered; the standard input and standard output streams are fully buffered if and only if the stream can be determined not to refer to an interactive device.

CX A stream associated with a memory buffer shall have the same operations for text files that a stream associated with an external file would have. In addition, the stream orientation shall be determined in exactly the same fashion.

Input and output operations on a stream associated with a memory buffer by a call to *fmemopen()* shall be constrained by the implementation to take place within the bounds of the memory buffer. In the case of a stream opened by *open_memstream()* or *open_wmemstream()*, the memory area shall grow dynamically to accommodate write operations as necessary. For output, data is moved from the buffer provided by *setvbuf()* to the memory stream during a flush or close operation.

2.5.1 Interaction of File Descriptors and Standard I/O Streams

CX This section describes the interaction of file descriptors and standard I/O streams. The functionality described in this section is an extension to the ISO C standard (and the rest of this section is not further CX shaded).

An open file description may be accessed through a file descriptor, which is created using functions such as *open()* or *pipe()*, or through a stream, which is created using functions such as *fopen()* or *popen()*. Either a file descriptor or a stream is called a “handle” on the open file description to which it refers; an open file description may have several handles.

Handles can be created or destroyed by explicit user action, without affecting the underlying open file description. Some of the ways to create them include *fcntl()*, *dup()*, *fdopen()*, *fileno()*, and *fork()*. They can be destroyed by at least *fclose()*, *close()*, and the *exec* functions.

A file descriptor that is never used in an operation that could affect the file offset (for example, *read()*, *write()*, or *lseek()*) is not considered a handle for this discussion, but could give rise to one (for example, as a consequence of *fdopen()*, *dup()*, or *fork()*). This exception does not include the file descriptor underlying a stream, whether created with *fopen()* or *fdopen()*, so long as it is not used directly by the application to affect the file offset. The *read()* and *write()* functions implicitly affect the file offset; *lseek()* explicitly affects it.

The result of function calls involving any one handle (the “active handle”) is defined elsewhere in this volume of POSIX.1-2008, but if two or more handles are used, and any one of them is a stream, the application shall ensure that their actions are coordinated as described below. If this is not done, the result is undefined.

A handle which is a stream is considered to be closed when either an *fclose()* or *freopen()* is executed on it (the result of *freopen()* is a new stream, which cannot be a handle on the same open file description as its previous value), or when the process owning that stream terminates with *exit()*, *abort()*, or due to a signal. A file descriptor is closed by *close()*, *_exit()*, or the *exec* functions when *FD_CLOEXEC* is set on that file descriptor.

For a handle to become the active handle, the application shall ensure that the actions below are performed between the last use of the handle (the current active handle) and the first use of the second handle (the future active handle). The second handle then becomes the active handle. All activity by the application affecting the file offset on the first handle shall be suspended until it again becomes the active file handle. (If a stream function has as an underlying function one that affects the file offset, the stream function shall be considered to affect the file offset.)

The handles need not be in the same process for these rules to apply.

Note that after a *fork()*, two handles exist where one existed before. The application shall ensure that, if both handles can ever be accessed, they are both in a state where the other could become the active handle first. The application shall prepare for a *fork()* exactly as if it were a change of active handle. (If the only action performed by one of the processes is one of the *exec* functions or *_exit()* (not *exit()*), the handle is never accessed in that process.)

For the first handle, the first applicable condition below applies. After the actions required below are taken, if the handle is still open, the application can close it.

- If it is a file descriptor, no action is required.
- If the only further action to be performed on any handle to this open file descriptor is to close it, no action need be taken.
- If it is a stream which is unbuffered, no action need be taken.
- If it is a stream which is line buffered, and the last byte written to the stream was a <newline> (that is, as if a:

```
putc ( '\n' )
```

was the most recent operation on that stream), no action need be taken.

- If it is a stream which is open for writing or appending (but not also open for reading), the application shall either perform an *fflush()*, or the stream shall be closed.
- If the stream is open for reading and it is at the end of the file (*feof()* is true), no action need be taken.
- If the stream is open with a mode that allows reading and the underlying open file description refers to a device that is capable of seeking, the application shall either perform an *fflush()*, or the stream shall be closed.

For the second handle:

- If any previous active handle has been used by a function that explicitly changed the file offset, except as required above for the first handle, the application shall perform an *lseek()* or *fseek()* (as appropriate to the type of handle) to an appropriate location.

If the active handle ceases to be accessible before the requirements on the first handle, above, have been met, the state of the open file description becomes undefined. This might occur during functions such as a *fork()* or *_exit()*.

The *exec* functions make inaccessible all streams that are open at the time they are called, independent of which streams or file descriptors may be available to the new process image.

When these rules are followed, regardless of the sequence of handles used, implementations shall ensure that an application, even one consisting of several processes, shall yield correct results: no data shall be lost or duplicated when writing, and all data shall be written in order, except as requested by seeks. It is implementation-defined whether, and under what conditions, all input is seen exactly once.

Each function that operates on a stream is said to have zero or more “underlying functions”. This means that the stream function shares certain traits with the underlying functions, but does not require that there be any relation between the implementations of the stream function and its underlying functions.

2.5.2 Stream Orientation and Encoding Rules

For conformance to the ISO/IEC 9899:1999 standard, the definition of a stream includes an “orientation”. After a stream is associated with an external file, but before any operations are performed on it, the stream is without orientation. Once a wide-character input/output function has been applied to a stream without orientation, the stream shall become “wide-oriented”. Similarly, once a byte input/output function has been applied to a stream without orientation, the stream shall become “byte-oriented”. Only a call to the *freopen()* function or the *fwide()* function can otherwise alter the orientation of a stream.

A successful call to *freopen()* shall remove any orientation. The three predefined streams *standard input*, *standard output*, and *standard error* shall be unoriented at program start-up.

Byte input/output functions cannot be applied to a wide-oriented stream, and wide-character input/output functions cannot be applied to a byte-oriented stream. The remaining stream operations shall not affect and shall not be affected by a stream’s orientation, except for the following additional restriction:

- For wide-oriented streams, after a successful call to a file-positioning function that leaves the file position indicator prior to the end-of-file, a wide-character output function can overwrite a partial character; any file contents beyond the byte(s) written are henceforth undefined.

Each wide-oriented stream has an associated **mbstate_t** object that stores the current parse state of the stream. A successful call to *fgetpos()* shall store a representation of the value of this **mbstate_t** object as part of the value of the **fpos_t** object. A later successful call to *fsetpos()* using the same stored **fpos_t** value shall restore the value of the associated **mbstate_t** object as well as the position within the controlled stream.

Implementations that support multiple encoding rules associate an encoding rule with the stream. The encoding rule shall be determined by the setting of the *LC_CTYPE* category in the current locale at the time when the stream becomes wide-oriented. As with the stream’s orientation, the encoding rule associated with a stream cannot be changed once it has been set, except by a successful call to *freopen()* which clears the encoding rule and resets the orientation to unoriented.

Although wide-oriented streams are conceptually sequences of wide characters, the external file associated with a wide-oriented stream is a sequence of (possibly multi-byte) characters generalized as follows:

- Multi-byte encodings within files may contain embedded null bytes (unlike multi-byte encodings valid for use internal to the program).
- A file need not begin nor end in the initial shift state.

Moreover, the encodings used for characters may differ among files. Both the nature and choice of such encodings are implementation-defined.

The wide-character input functions read characters from the stream and convert them to wide characters as if they were read by successive calls to the *fgetwc()* function. Each conversion shall occur as if by a call to the *mbrtowc()* function, with the conversion state described by the stream’s own **mbstate_t** object, except the encoding rule associated with the stream is used instead of the encoding rule implied by the *LC_CTYPE* category of the current locale.

The wide-character output functions convert wide characters to (possibly multi-byte) characters and write them to the stream as if they were written by successive calls to the *fputwc()* function. Each conversion shall occur as if by a call to the *wcrtomb()* function, with the conversion state described by the stream’s own **mbstate_t** object, except the encoding rule associated with the stream is used instead of the encoding rule implied by the *LC_CTYPE* category of the current

locale.

An “encoding error” shall occur if the character sequence presented to the underlying `mbrtowc()` function does not form a valid (generalized) character, or if the code value passed to the underlying `wcrtomb()` function does not correspond to a valid (generalized) character. The wide-character input/output functions and the byte input/output functions store the value of the macro `[EILSEQ]` in `errno` if and only if an encoding error occurs.

2.6 STREAMS

OB XSR STREAMS functionality is provided on implementations supporting the XSI STREAMS Option Group. The functionality described in this section is dependent on support of the XSI STREAMS option (and the rest of this section is not further shaded for this option).

STREAMS provides a uniform mechanism for implementing networking services and other character-based I/O. The STREAMS function provides direct access to protocol modules. STREAMS modules are unspecified objects. Access to STREAMS modules is provided by interfaces in POSIX.1-2008. Creation of STREAMS modules is outside the scope of POSIX.1-2008.

A STREAM is typically a full-duplex connection between a process and an open device or pseudo-device. However, since pipes may be STREAMS-based, a STREAM can be a full-duplex connection between two processes. The STREAM itself exists entirely within the implementation and provides a general character I/O function for processes. It optionally includes one or more intermediate processing modules that are interposed between the process end of the STREAM (STREAM head) and a device driver at the end of the STREAM (STREAM end).

STREAMS I/O is based on messages. There are three types of message:

- *Data messages* containing actual data for input or output
- *Control data* containing instructions for the STREAMS modules and underlying implementation
- Other messages, which include file descriptors

The interface between the STREAM and the rest of the implementation is provided by a set of functions at the STREAM head. When a process calls `write()`, `writew()`, `putmsg()`, `putpmsg()`, or `ioctl()`, messages are sent down the STREAM, and `read()`, `readv()`, `getmsg()`, or `getpmsg()` accepts data from the STREAM and passes it to a process. Data intended for the device at the downstream end of the STREAM is packaged into messages and sent downstream, while data and signals from the device are composed into messages by the device driver and sent upstream to the STREAM head.

When a STREAMS-based device is opened, a STREAM shall be created that contains the STREAM head and the STREAM end (driver). If pipes are STREAMS-based in an implementation, when a pipe is created, two STREAMS shall be created, each containing a STREAM head. Other modules are added to the STREAM using `ioctl()`. New modules are “pushed” onto the STREAM one at a time in last-in, first-out (LIFO) style, as though the STREAM was a push-down stack.

Priority

Message types are classified according to their queuing priority and may be *normal* (non-priority), *priority*, or *high-priority* messages. A message belongs to a particular priority band that determines its ordering when placed on a queue. Normal messages have a priority band of 0 and shall always be placed at the end of the queue following all other messages in the queue. High-priority messages are always placed at the head of a queue, but shall be discarded if there is already a high-priority message in the queue. Their priority band shall be ignored; they are high-priority by virtue of their type. Priority messages have a priority band greater than 0. Priority messages are always placed after any messages of the same or higher priority. High-priority and priority messages are used to send control and data information outside the normal flow of control. By convention, high-priority messages shall not be affected by flow control. Normal and priority messages have separate flow controls.

Message Parts

A process may access STREAMS messages that contain a data part, control part, or both. The data part is that information which is transmitted over the communication medium and the control information is used by the local STREAMS modules. The other types of messages are used between modules and are not accessible to processes. Messages containing only a data part are accessible via *putmsg()*, *putpmsg()*, *getmsg()*, *getpmsg()*, *read()*, *readv()*, *write()*, or *writv()*. Messages containing a control part with or without a data part are accessible via calls to *putmsg()*, *putpmsg()*, *getmsg()*, or *getpmsg()*.

2.6.1 Accessing STREAMS

A process accesses STREAMS-based files using the standard functions *close()*, *ioctl()*, *getmsg()*, *getpmsg()*, *open()*, *pipe()*, *poll()*, *putmsg()*, *putpmsg()*, *read()*, or *write()*. Refer to the applicable function definitions for general properties and errors.

Calls to *ioctl()* shall perform control functions on the STREAM associated with the file descriptor *fdes*. The control functions may be performed by the STREAM head, a STREAMS module, or the STREAMS driver for the STREAM.

STREAMS modules and drivers can detect errors, sending an error message to the STREAM head, thus causing subsequent functions to fail and set *errno* to the value specified in the message. In addition, STREAMS modules and drivers can elect to fail a particular *ioctl()* request alone by sending a negative acknowledgement message to the STREAM head. This shall cause just the pending *ioctl()* request to fail and set *errno* to the value specified in the message.

2.7 XSI Interprocess Communication

This section describes extensions to support interprocess communication. The functionality described in this section shall be provided on implementations that support the XSI option (and the rest of this section is not further shaded).

The following message passing, semaphore, and shared memory services form an XSI interprocess communication facility. Certain aspects of their operation are common, and are defined as follows.

IPC Functions		
<i>msgctl()</i>	<i>semctl()</i>	<i>shmat()</i>
<i>msgget()</i>	<i>semget()</i>	<i>shmctl()</i>
<i>msgrcv()</i>	<i>semop()</i>	<i>shmdt()</i>
<i>msgsnd()</i>		<i>shmget()</i>

Another interprocess communication facility is provided by functions in the Realtime Option Group; see [Section 2.8](#) (on page 502).

2.7.1 IPC General Description

Each individual shared memory segment, message queue, and semaphore set shall be identified by a unique positive integer, called, respectively, a shared memory identifier, *shmid*, a semaphore identifier, *semid*, and a message queue identifier, *msqid*. The identifiers shall be returned by calls to *shmget()*, *semget()*, and *msgget()*, respectively.

Associated with each identifier is a data structure which contains data related to the operations which may be or may have been performed; see the Base Definitions volume of POSIX.1-2008, [<sys/shm.h>](#), [<sys/sem.h>](#), and [<sys/msg.h>](#) for their descriptions.

Each of the data structures contains both ownership information and an **ipc_perm** structure (see the Base Definitions volume of POSIX.1-2008, [<sys/ipc.h>](#)) which are used in conjunction to determine whether or not read/write (read/alter for semaphores) permissions should be granted to processes using the IPC facilities. The *mode* member of the **ipc_perm** structure acts as a bit field which determines the permissions.

The values of the bits are given below in octal notation.

Bit	Meaning
0400	Read by user.
0200	Write by user.
0040	Read by group.
0020	Write by group.
0004	Read by others.
0002	Write by others.

The name of the **ipc_perm** structure is *shm_perm*, *sem_perm*, or *msg_perm*, depending on which service is being used. In each case, read and write/alter permissions shall be granted to a process if one or more of the following are true ("*xxx*" is replaced by *shm*, *sem*, or *msg*, as appropriate):

- The process has appropriate privileges.
- The effective user ID of the process matches *xxx_perm.cuid* or *xxx_perm.uid* in the data structure associated with the IPC identifier, and the appropriate bit of the *user* field in *xxx_perm.mode* is set.

- The effective user ID of the process does not match *xxx_perm.cuid* or *xxx_perm.uid* but the effective group ID of the process matches *xxx_perm.cgid* or *xxx_perm.gid* in the data structure associated with the IPC identifier, and the appropriate bit of the *group* field in *xxx_perm.mode* is set.
- The effective user ID of the process does not match *xxx_perm.cuid* or *xxx_perm.uid* and the effective group ID of the process does not match *xxx_perm.cgid* or *xxx_perm.gid* in the data structure associated with the IPC identifier, but the appropriate bit of the *other* field in *xxx_perm.mode* is set.

Otherwise, the permission shall be denied.

In addition to the **ipc_perm** structure, each associated data structure includes several **time_t** fields for recording timestamps of particular operations. When an operation is described as setting a timestamp to the current time, that particular timestamp member of the associated data structure shall be set to the largest **time_t** value which is not greater than the current time.

2.8 Realtime

This section defines functions to support the source portability of applications with realtime requirements. The presence of some of these functions is dependent on support for implementation options described in the text.

The specific functional areas included in this section and their scope include the following. Full definitions of these terms can be found in XBD [Chapter 3](#) (on page 33).

- Semaphores
- Process Memory Locking
- Memory Mapped Files and Shared Memory Objects
- Priority Scheduling
- Realtime Signal Extension
- Timers
- Interprocess Communication
- Synchronized Input and Output
- Asynchronous Input and Output

All the realtime functions defined in this volume of POSIX.1-2008 are portable, although some of the numeric parameters used by an implementation may have hardware dependencies.

2.8.1 Realtime Signals

See [Section 2.4.2](#) (on page 489).

2.8.2 Asynchronous I/O

An asynchronous I/O control block structure **aiocb** is used in many asynchronous I/O functions. It is defined in the Base Definitions volume of POSIX.1-2008, <**aio.h**> and has at least the following members:

Member Type	Member Name	Description
int	<i>aio_fildes</i>	File descriptor.
off_t	<i>aio_offset</i>	File offset.
volatile void*	<i>aio_buf</i>	Location of buffer.
size_t	<i>aio_nbytes</i>	Length of transfer.
int	<i>aio_reqprio</i>	Request priority offset.
struct sigevent	<i>aio_sigevent</i>	Signal number and value.
int	<i>aio_lio_opcode</i>	Operation to be performed.

The *aio_fildes* element is the file descriptor on which the asynchronous operation is performed.

If **O_APPEND** is not set for the file descriptor *aio_fildes* and if *aio_fildes* is associated with a device that is capable of seeking, then the requested operation takes place at the absolute position in the file as given by *aio_offset*, as if *lseek()* were called immediately prior to the operation with an *offset* argument equal to *aio_offset* and a *whence* argument equal to **SEEK_SET**. If **O_APPEND** is set for the file descriptor, or if *aio_fildes* is associated with a device that is incapable of seeking, write operations append to the file in the same order as the calls were made, with the following exception: under implementation-defined circumstances, such as operation on a multi-processor or when requests of differing priorities are submitted at the same time, the ordering restriction may be relaxed. Since there is no way for a strictly conforming application to determine whether this relaxation applies, all strictly conforming applications which rely on ordering of output shall be written in such a way that they will operate correctly if the relaxation applies. After a successful call to enqueue an asynchronous I/O operation, the value of the file offset for the file is unspecified. The *aio_nbytes* and *aio_buf* elements are the same as the *nbyte* and *buf* arguments defined by *read()* and *write()*, respectively.

If **_POSIX_PRIORITIZED_IO** and **_POSIX_PRIORITY_SCHEDULING** are defined, then asynchronous I/O is queued in priority order, with the priority of each asynchronous operation based on the current scheduling priority of the calling process. The *aio_reqprio* member can be used to lower (but not raise) the asynchronous I/O operation priority and is within the range zero through **{AIO_PRIO_DELTA_MAX}**, inclusive. Unless both **_POSIX_PRIORITIZED_IO** and **_POSIX_PRIORITY_SCHEDULING** are defined, the order of processing asynchronous I/O requests is unspecified. When both **_POSIX_PRIORITIZED_IO** and **_POSIX_PRIORITY_SCHEDULING** are defined, the order of processing of requests submitted by processes whose schedulers are not **SCHED_FIFO**, **SCHED_RR**, or **SCHED_SPORADIC** is unspecified. The priority of an asynchronous request is computed as (process scheduling priority) minus *aio_reqprio*. The priority assigned to each asynchronous I/O request is an indication of the desired order of execution of the request relative to other asynchronous I/O requests for this file. If **_POSIX_PRIORITIZED_IO** is defined, requests issued with the same priority to a character special file are processed by the underlying device in FIFO order; the order of processing of requests of the same priority issued to files that are not character special files is unspecified. Numerically higher priority values indicate requests of higher priority. The value of *aio_reqprio* has no effect on process scheduling priority. When prioritized asynchronous I/O requests to the same file are blocked waiting for a resource required for that I/O operation, the higher-priority I/O requests shall be granted the resource before lower-priority I/O requests are granted the resource. The relative priority of asynchronous I/O and synchronous I/O is implementation-defined. If **_POSIX_PRIORITIZED_IO** is defined, the implementation shall define for which files I/O prioritization is supported.

The *aio_sigevent* determines how the calling process shall be notified upon I/O completion, as specified in [Section 2.4.1](#) (on page 488). If *aio_sigevent.sigev_notify* is SIGEV_NONE, then no signal shall be posted upon I/O completion, but the error status for the operation and the return status for the operation shall be set appropriately.

The *aio_lio_opcode* field is used only by the *lio_listio()* call. The *lio_listio()* call allows multiple asynchronous I/O operations to be submitted at a single time. The function takes as an argument an array of pointers to **aioctx** structures. Each **aioctx** structure indicates the operation to be performed (read or write) via the *aio_lio_opcode* field.

The address of the **aioctx** structure is used as a handle for retrieving the error status and return status of the asynchronous operation while it is in progress.

The **aioctx** structure and the data buffers associated with the asynchronous I/O operation are being used by the system for asynchronous I/O while, and only while, the error status of the asynchronous operation is equal to [EINPROGRESS]. Applications shall not modify the **aioctx** structure while the structure is being used by the system for asynchronous I/O.

The return status of the asynchronous operation is the number of bytes transferred by the I/O operation. If the error status is set to indicate an error completion, then the return status is set to the return value that the corresponding *read()*, *write()*, or *fsync()* call would have returned. When the error status is not equal to [EINPROGRESS], the return status shall reflect the return status of the corresponding synchronous operation.

2.8.3 Memory Management

2.8.3.1 Memory Locking

Range memory locking operations are defined in terms of pages. Implementations may restrict the size and alignment of range lockings to be on page-size boundaries. The page size, in bytes, is the value of the configurable system variable {PAGESIZE}. If an implementation has no restrictions on size or alignment, it may specify a 1-byte page size.

Memory locking guarantees the residence of portions of the address space. It is implementation-defined whether locking memory guarantees fixed translation between virtual addresses (as seen by the process) and physical addresses. Per-process memory locks are not inherited across a *fork()*, and all memory locks owned by a process are unlocked upon *exec* or process termination. Unmapping of an address range removes any memory locks established on that address range by this process.

2.8.3.2 Memory Mapped Files

Range memory mapping operations are defined in terms of pages. Implementations may restrict the size and alignment of range mappings to be on page-size boundaries. The page size, in bytes, is the value of the configurable system variable {PAGESIZE}. If an implementation has no restrictions on size or alignment, it may specify a 1-byte page size.

Memory mapped files provide a mechanism that allows a process to access files by directly incorporating file data into its address space. Once a file is mapped into a process address space, the data can be manipulated as memory. If more than one process maps a file, its contents are shared among them. If the mappings allow shared write access, then data written into the memory object through the address space of one process appears in the address spaces of all processes that similarly map the same portion of the memory object.

17304 SHM Shared memory objects are named regions of storage that may be independent of the file system
 17305 and can be mapped into the address space of one or more processes to allow them to share the
 17306 associated memory.

17307 SHM An *unlink()* of a file or *shm_unlink()* of a shared memory object, while causing the removal of
 17308 the name, does not unmap any mappings established for the object. Once the name has been
 17309 removed, the contents of the memory object are preserved as long as it is referenced. The
 17310 memory object remains referenced as long as a process has the memory object open or has some
 17311 area of the memory object mapped.

17312 2.8.3.3 *Memory Protection*

17313 When an object is mapped, various application accesses to the mapped region may result in
 17314 signals. In this context, SIGBUS is used to indicate an error using the mapped object, and
 17315 SIGSEGV is used to indicate a protection violation or misuse of an address:

- 17316 • A mapping may be restricted to disallow some types of access.
- 17317 • Write attempts to memory that was mapped without write access, or any access to
 17318 memory mapped PROT_NONE, shall result in a SIGSEGV signal.
- 17319 • References to unmapped addresses shall result in a SIGSEGV signal.
- 17320 • Reference to whole pages within the mapping, but beyond the current length of the object,
 17321 shall result in a SIGBUS signal.
- 17322 • The size of the object is unaffected by access beyond the end of the object (even if a
 17323 SIGBUS is not generated).

17324 2.8.3.4 *Typed Memory Objects*

17325 TYM The functionality described in this section shall be provided on implementations that support
 17326 the Typed Memory Objects option (and the rest of this section is not further shaded for this
 17327 option).

17328 Implementations may support the Typed Memory Objects option independently of support for
 17329 memory mapped files or shared memory objects. Typed memory objects are implementation-
 17330 configurable named storage pools accessible from one or more processors in a system, each via
 17331 one or more ports, such as backplane buses, LANs, I/O channels, and so on. Each valid
 17332 combination of a storage pool and a port is identified through a name that is defined at system
 17333 configuration time, in an implementation-defined manner; the name may be independent of the
 17334 file system. Using this name, a typed memory object can be opened and mapped into process
 17335 address space. For a given storage pool and port, it is necessary to support both dynamic
 17336 allocation from the pool as well as mapping at an application-supplied offset within the pool;
 17337 when dynamic allocation has been performed, subsequent deallocation must be supported.
 17338 Lastly, accessing typed memory objects from different ports requires a method for obtaining the
 17339 offset and length of contiguous storage of a region of typed memory (dynamically allocated or
 17340 not); this allows typed memory to be shared among processes and/or processors while being
 17341 accessed from the desired port.

2.8.4 Process Scheduling

PS The functionality described in this section shall be provided on implementations that support the Process Scheduling option (and the rest of this section is not further shaded for this option).

Scheduling Policies

The scheduling semantics described in this volume of POSIX.1-2008 are defined in terms of a conceptual model that contains a set of thread lists. No implementation structures are necessarily implied by the use of this conceptual model. It is assumed that no time elapses during operations described using this model, and therefore no simultaneous operations are possible. This model discusses only processor scheduling for runnable threads, but it should be noted that greatly enhanced predictability of realtime applications results if the sequencing of other resources takes processor scheduling policy into account.

There is, conceptually, one thread list for each priority. A runnable thread will be on the thread list for that thread's priority. Multiple scheduling policies shall be provided. Each non-empty thread list is ordered, contains a head as one end of its order, and a tail as the other. The purpose of a scheduling policy is to define the allowable operations on this set of lists (for example, moving threads between and within lists).

The POSIX model treats a "process" as an aggregation of system resources, including one or more threads that may be scheduled by the operating system on the processor(s) it controls. Although a process has its own set of scheduling attributes, these have an indirect effect (if any) on the scheduling behavior of individual threads as described below.

Each thread shall be controlled by an associated scheduling policy and priority. These parameters may be specified by explicit application execution of the *pthread_setschedparam()* function. Additionally, the scheduling parameters of a thread (but not its scheduling policy) may be changed by application execution of the *pthread_setschedprio()* function.

Each process shall be controlled by an associated scheduling policy and priority. These parameters may be specified by explicit application execution of the *sched_setscheduler()* or *sched_setparam()* functions.

The effect of the process scheduling attributes on individual threads in the process is dependent on the scheduling contention scope of the threads (see [Section 2.9.4](#), on page 514):

- For threads with system scheduling contention scope, the process scheduling attributes shall have no effect on the scheduling attributes or behavior either of the thread or an underlying kernel scheduling entity dedicated to that thread.
- For threads with process scheduling contention scope, the process scheduling attributes shall have no effect on the scheduling attributes of the thread. However, any underlying kernel scheduling entity used by these threads shall at all times behave as specified by the scheduling attributes of the containing process, and this behavior may affect the scheduling behavior of the process contention scope threads. For example, a process contention scope thread with scheduling policy *SCHED_FIFO* and the system maximum priority *H* (the value returned by *sched_get_priority_max(SCHED_FIFO)*) in a process with scheduling policy *SCHED_RR* and system minimum priority *L* (the value returned by *sched_get_priority_min(SCHED_RR)*) shall be subject to timeslicing and to preemption by any thread with an effective priority higher than *L*.

Associated with each policy is a priority range. Each policy definition shall specify the minimum priority range for that policy. The priority ranges for each policy may but need not overlap the priority ranges of other policies.

A conforming implementation shall select the thread that is defined as being at the head of the

17388 highest priority non-empty thread list to become a running thread, regardless of its associated
17389 policy. This thread is then removed from its thread list.

17390 Four scheduling policies are specifically required. Other implementation-defined scheduling
17391 policies may be defined. The following symbols are defined in the Base Definitions volume of
17392 POSIX.1-2008, <sched.h>:

17393 SCHED_FIFO First in, first out (FIFO) scheduling policy.

17394 SCHED_RR Round robin scheduling policy.

17395 SS SCHED_SPORADIC Sporadic server scheduling policy.

17396 SCHED_OTHER Another scheduling policy.

17397 The values of these symbols shall be distinct.

17398 SCHED_FIFO

17399 Conforming implementations shall include a scheduling policy called the FIFO scheduling
17400 policy.

17401 Threads scheduled under this policy are chosen from a thread list that is ordered by the time its
17402 threads have been on the list without being executed; generally, the head of the list is the thread
17403 that has been on the list the longest time, and the tail is the thread that has been on the list the
17404 shortest time.

17405 Under the SCHED_FIFO policy, the modification of the definitional thread lists is as follows:

- 17406 1. When a running thread becomes a preempted thread, it becomes the head of the thread
17407 list for its priority.
- 17408 2. When a blocked thread becomes a runnable thread, it becomes the tail of the thread list
17409 for its priority.
- 17410 3. When a running thread calls the *sched_setscheduler()* function, the process specified in the
17411 function call is modified to the specified policy and the priority specified by the *param*
17412 argument.
- 17413 4. When a running thread calls the *sched_setparam()* function, the priority of the process
17414 specified in the function call is modified to the priority specified by the *param* argument.
- 17415 5. When a running thread calls the *pthread_setschedparam()* function, the thread specified in
17416 the function call is modified to the specified policy and the priority specified by the *param*
17417 argument.
- 17418 6. When a running thread calls the *pthread_setschedprio()* function, the thread specified in the
17419 function call is modified to the priority specified by the *prio* argument.
- 17420 7. If a thread whose policy or priority has been modified other than by *pthread_setschedprio()*
17421 is a running thread or is runnable, it then becomes the tail of the thread list for its new
17422 priority.
- 17423 8. If a thread whose priority has been modified by *pthread_setschedprio()* is a running thread
17424 or is runnable, the effect on its position in the thread list depends on the direction of the
17425 modification, as follows:
 - 17426 a. If the priority is raised, the thread becomes the tail of the thread list.
 - 17427 b. If the priority is unchanged, the thread does not change position in the thread list.

- 17428 c. If the priority is lowered, the thread becomes the head of the thread list.
- 17429 9. When a running thread issues the *sched_yield()* function, the thread becomes the tail of
- 17430 the thread list for its priority.
- 17431 10. At no other time is the position of a thread with this scheduling policy within the thread
- 17432 lists affected.

17433 For this policy, valid priorities shall be within the range returned by the *sched_get_priority_max()*

17434 and *sched_get_priority_min()* functions when SCHED_FIFO is provided as the parameter.

17435 Conforming implementations shall provide a priority range of at least 32 priorities for this

17436 policy.

17437 SCHED_RR

17438 Conforming implementations shall include a scheduling policy called the “round robin”

17439 scheduling policy. This policy shall be identical to the SCHED_FIFO policy with the additional

17440 condition that when the implementation detects that a running thread has been executing as a

17441 running thread for a time period of the length returned by the *sched_rr_get_interval()* function or

17442 longer, the thread shall become the tail of its thread list and the head of that thread list shall be

17443 removed and made a running thread.

17444 The effect of this policy is to ensure that if there are multiple SCHED_RR threads at the same

17445 priority, one of them does not monopolize the processor. An application should not rely only on

17446 the use of SCHED_RR to ensure application progress among multiple threads if the application

17447 includes threads using the SCHED_FIFO policy at the same or higher priority levels or

17448 SCHED_RR threads at a higher priority level.

17449 A thread under this policy that is preempted and subsequently resumes execution as a running

17450 thread completes the unexpired portion of its round robin interval time period.

17451 For this policy, valid priorities shall be within the range returned by the *sched_get_priority_max()*

17452 and *sched_get_priority_min()* functions when SCHED_RR is provided as the parameter.

17453 Conforming implementations shall provide a priority range of at least 32 priorities for this

17454 policy.

17455 SCHED_SPORADIC

17456 SS|TSP The functionality described in this section shall be provided on implementations that support

17457 the Process Sporadic Server or Thread Sporadic Server options (and the rest of this section is not

17458 further shaded for these options).

17459 If `_POSIX_SPORADIC_SERVER` or `_POSIX_THREAD_SPORADIC_SERVER` is defined, the

17460 implementation shall include a scheduling policy identified by the value SCHED_SPORADIC.

17461 The sporadic server policy is based primarily on two time parameters: the replenishment period

17462 and the available execution capacity. The replenishment period is given by the

17463 *sched_ss_repl_period* member of the **sched_param** structure. The available execution capacity is

17464 initialized to the value given by the *sched_ss_init_budget* member of the same parameter. The

17465 sporadic server policy is identical to the SCHED_FIFO policy with some additional conditions

17466 that cause the thread’s assigned priority to be switched between the values specified by the

17467 *sched_priority* and *sched_ss_low_priority* members of the **sched_param** structure.

17468 The priority assigned to a thread using the sporadic server scheduling policy is determined in

17469 the following manner: if the available execution capacity is greater than zero and the number of

17470 pending replenishment operations is strictly less than *sched_ss_max_repl*, the thread is assigned

17471 the priority specified by *sched_priority*; otherwise, the assigned priority shall be

17472 *sched_ss_low_priority*. If the value of *sched_priority* is less than or equal to the value of

sched_ss_low_priority, the results are undefined. When active, the thread shall belong to the thread list corresponding to its assigned priority level, according to the mentioned priority assignment. The modification of the available execution capacity and, consequently of the assigned priority, is done as follows:

1. When the thread at the head of the *sched_priority* list becomes a running thread, its execution time shall be limited to at most its available execution capacity, plus the resolution of the execution time clock used for this scheduling policy. This resolution shall be implementation-defined.
2. Each time the thread is inserted at the tail of the list associated with *sched_priority*—because as a blocked thread it became runnable with priority *sched_priority* or because a replenishment operation was performed—the time at which this operation is done is posted as the *activation_time*.
3. When the running thread with assigned priority equal to *sched_priority* becomes a preempted thread, it becomes the head of the thread list for its priority, and the execution time consumed is subtracted from the available execution capacity. If the available execution capacity would become negative by this operation, it shall be set to zero.
4. When the running thread with assigned priority equal to *sched_priority* becomes a blocked thread, the execution time consumed is subtracted from the available execution capacity, and a replenishment operation is scheduled, as described in 6 and 7. If the available execution capacity would become negative by this operation, it shall be set to zero.
5. When the running thread with assigned priority equal to *sched_priority* reaches the limit imposed on its execution time, it becomes the tail of the thread list for *sched_ss_low_priority*, the execution time consumed is subtracted from the available execution capacity (which becomes zero), and a replenishment operation is scheduled, as described in 6 and 7.
6. Each time a replenishment operation is scheduled, the amount of execution capacity to be replenished, *replenish_amount*, is set equal to the execution time consumed by the thread since the *activation_time*. The replenishment is scheduled to occur at *activation_time* plus *sched_ss_repl_period*. If the scheduled time obtained is before the current time, the replenishment operation is carried out immediately. Several replenishment operations may be pending at the same time, each of which will be serviced at its respective scheduled time. With the above rules, the number of replenishment operations simultaneously pending for a given thread that is scheduled under the sporadic server policy shall not be greater than *sched_ss_max_repl*.
7. A replenishment operation consists of adding the corresponding *replenish_amount* to the available execution capacity at the scheduled time. If, as a consequence of this operation, the execution capacity would become larger than *sched_ss_initial_budget*, it shall be rounded down to a value equal to *sched_ss_initial_budget*. Additionally, if the thread was runnable or running, and had assigned priority equal to *sched_ss_low_priority*, then it becomes the tail of the thread list for *sched_priority*.

Execution time is defined in XBD [Section 3.118](#) (on page 52).

For this policy, changing the value of a CPU-time clock via *clock_settime()* shall have no effect on its behavior.

For this policy, valid priorities shall be within the range returned by the *sched_get_priority_min()* and *sched_get_priority_max()* functions when SCHED_SPORADIC is provided as the parameter. Conforming implementations shall provide a priority range of at least 32 distinct priorities for this policy.

If the scheduling policy of the target process is either SCHED_FIFO or SCHED_RR, the *sched_ss_low_priority*, *sched_ss_repl_period*, and *sched_ss_init* budget members of the *param* argument shall have no effect on the scheduling behavior. If the scheduling policy of this process is not SCHED_FIFO, SCHED_RR, or SCHED_SPORADIC, the effects of these members are implementation-defined; this case includes the SCHED_OTHER policy.

SCHED_OTHER

Conforming implementations shall include one scheduling policy identified as SCHED_OTHER (which may execute identically with either the FIFO or round robin scheduling policy). The effect of scheduling threads with the SCHED_OTHER policy in a system in which other threads are executing under SCHED_FIFO, SCHED_RR, or SCHED_SPORADIC is implementation-defined.

This policy is defined to allow strictly conforming applications to be able to indicate in a portable manner that they no longer need a realtime scheduling policy.

For threads executing under this policy, the implementation shall use only priorities within the range returned by the *sched_get_priority_max()* and *sched_get_priority_min()* functions when SCHED_OTHER is provided as the parameter.

2.8.5 Clocks and Timers

The `<time.h>` header defines the types and manifest constants used by the timing facility.

Time Value Specification Structures

Many of the timing facility functions accept or return time value specifications. A time value structure **timespec** specifies a single time value and includes at least the following members:

Member Type	Member Name	Description
time_t	<i>tv_sec</i>	Seconds.
long	<i>tv_nsec</i>	Nanoseconds.

The *tv_nsec* member is only valid if greater than or equal to zero, and less than the number of nanoseconds in a second (1 000 million). The time interval described by this structure is $(tv\_sec * 10^9 + tv\_nsec)$ nanoseconds.

A time value structure **itimerspec** specifies an initial timer value and a repetition interval for use by the per-process timer functions. This structure includes at least the following members:

Member Type	Member Name	Description
struct timespec	<i>it_interval</i>	Timer period.
struct timespec	<i>it_value</i>	Timer expiration.

If the value described by *it_value* is non-zero, it indicates the time to or time of the next timer expiration (for relative and absolute timer values, respectively). If the value described by *it_value* is zero, the timer shall be disarmed.

If the value described by *it_interval* is non-zero, it specifies an interval which shall be used in reloading the timer when it expires; that is, a periodic timer is specified. If the value described by *it_interval* is zero, the timer is disarmed after its next expiration; that is, a one-shot timer is specified.

17559 Timer Event Notification Control Block

17560 Per-process timers may be created that notify the process of timer expirations by queuing a
 17561 realtime extended signal. The **sigevent** structure, defined in the Base Definitions volume of
 17562 POSIX.1-2008, **<signal.h>**, is used in creating such a timer. The **sigevent** structure contains the
 17563 signal number and an application-specific data value which shall be used when notifying the
 17564 calling process of timer expiration events.

17565 Manifest Constants

17566 The following constants are defined in the Base Definitions volume of POSIX.1-2008, **<time.h>**:

17567 **CLOCK_REALTIME** The identifier for the system-wide realtime clock.

17568 **TIMER_ABSTIME** Flag indicating time is absolute with respect to the clock associated
 17569 with a timer.

17570 **CLOCK_MONOTONIC** The identifier for the system-wide monotonic clock, which is defined
 17571 as a clock whose value cannot be set via *clock_settime()* and which
 17572 cannot have backward clock jumps. The maximum possible clock
 17573 jump is implementation-defined.

17574 **MON** The maximum allowable resolution for **CLOCK_REALTIME** and **CLOCK_MONOTONIC** clocks
 17575 and all time services based on these clocks is represented by **{_POSIX_CLOCKRES_MIN}** and
 17576 shall be defined as 20 ms (1/50 of a second). Implementations may support smaller values of
 17577 resolution for these clocks to provide finer granularity time bases. The actual resolution
 17578 supported by an implementation for a specific clock is obtained using the *clock_getres()* function.
 17579 If the actual resolution supported for a time service based on one of these clocks differs from the
 17580 resolution supported for that clock, the implementation shall document this difference.

17581 **MON** The minimum allowable maximum value for **CLOCK_REALTIME** and **CLOCK_MONOTONIC**
 17582 clocks and all absolute time services based on them is the same as that defined by the ISO C
 17583 standard for the **time_t** type. If the maximum value supported by a time service based on one of
 17584 these clocks differs from the maximum value supported by that clock, the implementation shall
 17585 document this difference.

17586 Execution Time Monitoring

17587 **CPT** If **_POSIX_CPUTIME** is defined, process CPU-time clocks shall be supported in addition to the
 17588 clocks described in [Manifest Constants](#).

17589 **TCT** If **_POSIX_THREAD_CPUTIME** is defined, thread CPU-time clocks shall be supported.

17590 **CPT|TCT** CPU-time clocks measure execution or CPU time, which is defined in XBD [Section 3.118](#) (on
 17591 page 52). The mechanism used to measure execution time is described in XBD [Section 4.10](#) (on
 17592 page 110).

17593 **CPT** If **_POSIX_CPUTIME** is defined, the following constant of the type **clockid_t** is defined in
 17594 **<time.h>**:

17595 **CLOCK_PROCESS_CPUTIME_ID**

17596 When this value of the type **clockid_t** is used in a *clock()* or *timer*()* function call, it is
 17597 interpreted as the identifier of the CPU-time clock associated with the process making the
 17598 function call.

17599 **TCT** If **_POSIX_THREAD_CPUTIME** is defined, the following constant of the type **clockid_t** is
 17600 defined in **<time.h>**:

CLOCK_THREAD_CPUTIME_ID

When this value of the type `clockid_t` is used in a `clock()` or `timer*()` function call, it is interpreted as the identifier of the CPU-time clock associated with the thread making the function call.

2.9 Threads

This section defines functionality to support multiple flows of control, called “threads”, within a process. For the definition of threads, see XBD [Section 3.398](#) (on page 97).

The specific functional areas covered by threads and their scope include:

- Thread management: the creation, control, and termination of multiple flows of control in the same process under the assumption of a common shared address space
- Synchronization primitives optimized for tightly coupled operation of multiple control flows in a common, shared address space

2.9.1 Thread-Safety

All functions defined by this volume of POSIX.1-2008 shall be thread-safe, except that the following functions⁷ need not be thread-safe.

<code>asctime()</code>	<code>ftw()</code>	<code>getservbyport()</code>	<code>nl_langinfo()</code>
<code>basename()</code>	<code>getc_unlocked()</code>	<code>getservent()</code>	<code>ptsname()</code>
<code>catgets()</code>	<code>getchar_unlocked()</code>	<code>getutxent()</code>	<code>putc_unlocked()</code>
<code>crypt()</code>	<code>getdate()</code>	<code>getutxid()</code>	<code>putchar_unlocked()</code>
<code>ctime()</code>	<code>getenv()</code>	<code>getutxline()</code>	<code>putenv()</code>
<code>dbm_clearerr()</code>	<code>getgrent()</code>	<code>gmtime()</code>	<code>pututxline()</code>
<code>dbm_close()</code>	<code>getgrgid()</code>	<code>hcreate()</code>	<code>rand()</code>
<code>dbm_delete()</code>	<code>getgrnam()</code>	<code>hdestroy()</code>	<code>readdir()</code>
<code>dbm_error()</code>	<code>gethostent()</code>	<code>hsearch()</code>	<code>setenv()</code>
<code>dbm_fetch()</code>	<code>getlogin()</code>	<code>inet_ntoa()</code>	<code>setgrent()</code>
<code>dbm_firstkey()</code>	<code>getnetbyaddr()</code>	<code>l64a()</code>	<code>setkey()</code>
<code>dbm_nextkey()</code>	<code>getnetbyname()</code>	<code>lgamma()</code>	<code>setpwent()</code>
<code>dbm_open()</code>	<code>getnetent()</code>	<code>lgammaf()</code>	<code>setutxent()</code>
<code>dbm_store()</code>	<code>getopt()</code>	<code>lgammal()</code>	<code>strerror()</code>
<code>dirname()</code>	<code>getprotobyname()</code>	<code>localeconv()</code>	<code>strsignal()</code>
<code>dlderror()</code>	<code>getprotobynumber()</code>	<code>localtime()</code>	<code>strtok()</code>
<code>drand48()</code>	<code>getprotoent()</code>	<code>lrand48()</code>	<code>system()</code>
<code>encrypt()</code>	<code>getpwent()</code>	<code>mblen()</code>	<code>ttyname()</code>
<code>endgrent()</code>	<code>getpwnam()</code>	<code>mbtowc()</code>	<code>unsetenv()</code>
<code>endpwent()</code>	<code>getpwuid()</code>	<code>mrand48()</code>	<code>wctomb()</code>
<code>endutxent()</code>	<code>getservbyname()</code>	<code>nftw()</code>	

The `ctermid()` and `tmpnam()` functions need not be thread-safe if passed a NULL argument. The `mbrlen()`, `mbrtowc()`, `mbsnrtowcs()`, `mbsrtowcs()`, `wcrtomb()`, `wcsnrtombs()`, and `wcsrtombs()` functions need not be thread-safe if passed a NULL `ps` argument.

⁷ The functions in the table are not shaded to denote applicable options. Individual reference pages should be consulted.

Implementations shall provide internal synchronization as necessary in order to satisfy this requirement.

Since multi-threaded applications are not allowed to use the *environ* variable to access or modify any environment variable while any other thread is concurrently modifying any environment variable, any function dependent on any environment variable is not thread-safe if another thread is modifying the environment; see XSH *exec* (on page 777).

2.9.2 Thread IDs

Although implementations may have thread IDs that are unique in a system, applications should only assume that thread IDs are usable and unique within a single process. The effect of calling any of the functions defined in this volume of POSIX.1-2008 and passing as an argument the thread ID of a thread from another process is unspecified. The lifetime of a thread ID ends after the thread terminates if it was created with the *detachstate* attribute set to `PTHREAD_CREATE_DETACHED` or if *pthread_detach()* or *pthread_join()* has been called for that thread. A conforming implementation is free to reuse a thread ID after its lifetime has ended. If an application attempts to use a thread ID whose lifetime has ended, the behavior is undefined.

If a thread is detached, its thread ID is invalid for use as an argument in a call to *pthread_detach()* or *pthread_join()*.

2.9.3 Thread Mutexes

A thread that has blocked shall not prevent any unblocked thread that is eligible to use the same processing resources from eventually making forward progress in its execution. Eligibility for processing resources is determined by the scheduling policy.

A thread shall become the owner of a mutex, *m*, when one of the following occurs:

- It calls *pthread_mutex_lock()* with *m* as the *mutex* argument and the call returns zero or `[EOWNERDEAD]`.
- It calls *pthread_mutex_trylock()* with *m* as the *mutex* argument and the call returns zero or `[EOWNERDEAD]`.
- It calls *pthread_mutex_timedlock()* with *m* as the *mutex* argument and the call returns zero or `[EOWNERDEAD]`.
- It calls *pthread_mutex_setprioceiling()* with *m* as the *mutex* argument and the call returns `[EOWNERDEAD]`.
- It calls *pthread_cond_wait()* with *m* as the *mutex* argument and the call returns zero or certain error numbers (see *pthread_cond_timedwait()*).
- It calls *pthread_cond_timedwait()* with *m* as the *mutex* argument and the call returns zero or certain error numbers (see *pthread_cond_timedwait()*).

The thread shall remain the owner of *m* until one of the following occurs:

- It executes *pthread_mutex_unlock()* with *m* as the *mutex* argument
- It blocks in a call to *pthread_cond_wait()* with *m* as the *mutex* argument.
- It blocks in a call to *pthread_cond_timedwait()* with *m* as the *mutex* argument.

The implementation shall behave as if at all times there is at most one owner of any mutex.

A thread that becomes the owner of a mutex is said to have “acquired” the mutex and the mutex

is said to have become “locked”; when a thread gives up ownership of a mutex it is said to have “released” the mutex and the mutex is said to have become “unlocked”.

A problem can occur if a process terminates while one of its threads holds a mutex lock. Depending on the mutex type, it might be possible for another thread to unlock the mutex and recover the state of the mutex. However, it is difficult to perform this recovery reliably.

Robust mutexes provide a means to enable the implementation to notify other threads in the event of a process terminating while one of its threads holds a mutex lock. The next thread that acquires the mutex is notified about the termination by the return value [EOWNERDEAD] from the locking function. The notified thread can then attempt to recover the state protected by the mutex, and if successful mark the state protected by the mutex as consistent by a call to `pthread_mutex_consistent()`. If the notified thread is unable to recover the state, it can declare the state as not recoverable by a call to `pthread_mutex_unlock()` without a prior call to `pthread_mutex_consistent()`.

Whether or not the state protected by a mutex can be recovered is dependent solely on the application using robust mutexes. The robust mutex support provided in the implementation provides notification only that a mutex owner has terminated while holding a lock, or that the state of the mutex is not recoverable.

2.9.4 Thread Scheduling

TPS

The functionality described in this section shall be provided on implementations that support the Thread Execution Scheduling option (and the rest of this section is not further shaded for this option).

Thread Scheduling Attributes

In support of the scheduling function, threads have attributes which are accessed through the `pthread_attr_t` thread creation attributes object.

The *contentionscope* attribute defines the scheduling contention scope of the thread to be either `PTHREAD_SCOPE_PROCESS` or `PTHREAD_SCOPE_SYSTEM`.

The *inheritsched* attribute specifies whether a newly created thread is to inherit the scheduling attributes of the creating thread or to have its scheduling values set according to the other scheduling attributes in the `pthread_attr_t` object.

The *schedpolicy* attribute defines the scheduling policy for the thread. The *schedparam* attribute defines the scheduling parameters for the thread. The interaction of threads having different policies within a process is described as part of the definition of those policies.

If the Thread Execution Scheduling option is defined, and the *schedpolicy* attribute specifies one of the priority-based policies defined under this option, the *schedparam* attribute contains the scheduling priority of the thread. A conforming implementation ensures that the priority value in *schedparam* is in the range associated with the scheduling policy when the thread attributes object is used to create a thread, or when the scheduling attributes of a thread are dynamically modified. The meaning of the priority value in *schedparam* is the same as that of *priority*.

TSP

If `_POSIX_THREAD_SPORADIC_SERVER` is defined, the *schedparam* attribute supports four new members that are used for the sporadic server scheduling policy. These members are *sched_ss_low_priority*, *sched_ss_repl_period*, *sched_ss_init_budget*, and *sched_ss_max_repl*. The meaning of these attributes is the same as in the definitions that appear under [Section 2.8.4](#) (on page 506).

When a process is created, its single thread has a scheduling policy and associated attributes

equal to the policy and attributes of the process. The default scheduling contention scope value is implementation-defined. The default values of other scheduling attributes are implementation-defined.

Thread Scheduling Contention Scope

The scheduling contention scope of a thread defines the set of threads with which the thread competes for use of the processing resources. The scheduling operation selects at most one thread to execute on each processor at any point in time and the thread's scheduling attributes (for example, *priority*), whether under process scheduling contention scope or system scheduling contention scope, are the parameters used to determine the scheduling decision.

The scheduling contention scope, in the context of scheduling a mixed scope environment, affects threads as follows:

- A thread created with `PTHREAD_SCOPE_SYSTEM` scheduling contention scope contends for resources with all other threads in the same scheduling allocation domain relative to their system scheduling attributes. The system scheduling attributes of a thread created with `PTHREAD_SCOPE_SYSTEM` scheduling contention scope are the scheduling attributes with which the thread was created. The system scheduling attributes of a thread created with `PTHREAD_SCOPE_PROCESS` scheduling contention scope are the implementation-defined mapping into system attribute space of the scheduling attributes with which the thread was created.
- Threads created with `PTHREAD_SCOPE_PROCESS` scheduling contention scope contend directly with other threads within their process that were created with `PTHREAD_SCOPE_PROCESS` scheduling contention scope. The contention is resolved based on the threads' scheduling attributes and policies. It is unspecified how such threads are scheduled relative to threads in other processes or threads with `PTHREAD_SCOPE_SYSTEM` scheduling contention scope.
- Conforming implementations shall support the `PTHREAD_SCOPE_PROCESS` scheduling contention scope, the `PTHREAD_SCOPE_SYSTEM` scheduling contention scope, or both.

Scheduling Allocation Domain

Implementations shall support scheduling allocation domains containing one or more processors. It should be noted that the presence of multiple processors does not automatically indicate a scheduling allocation domain size greater than one. Conforming implementations on multi-processors may map all or any subset of the CPUs to one or multiple scheduling allocation domains, and could define these scheduling allocation domains on a per-thread, per-process, or per-system basis, depending on the types of applications intended to be supported by the implementation. The scheduling allocation domain is independent of scheduling contention scope, as the scheduling contention scope merely defines the set of threads with which a thread contends for processor resources, while scheduling allocation domain defines the set of processors for which it contends. The semantics of how this contention is resolved among threads for processors is determined by the scheduling policies of the threads.

The choice of scheduling allocation domain size and the level of application control over scheduling allocation domains is implementation-defined. Conforming implementations may change the size of scheduling allocation domains and the binding of threads to scheduling allocation domains at any time.

For application threads with scheduling allocation domains of size equal to one, the scheduling rules defined for `SCHED_FIFO` and `SCHED_RR` shall be used; see [Scheduling Policies](#) (on page 506). All threads with system scheduling contention scope, regardless of the processes in which they reside, compete for the processor according to their priorities. Threads with process

17772 scheduling contention scope compete only with other threads with process scheduling
17773 contention scope within their process.

17774 For application threads with scheduling allocation domains of size greater than one, the rules
17775 TSP defined for SCHED_FIFO, SCHED_RR, and SCHED_SPORADIC shall be used in an
17776 implementation-defined manner. Each thread with system scheduling contention scope
17777 competes for the processors in its scheduling allocation domain in an implementation-defined
17778 manner according to its priority. Threads with process scheduling contention scope are
17779 scheduled relative to other threads within the same scheduling contention scope in the process.

17780 TSP If _POSIX_THREAD_SPORADIC_SERVER is defined, the rules defined for SCHED_SPORADIC
17781 in *Scheduling Policies* (on page 506) shall be used in an implementation-defined manner for
17782 application threads whose scheduling allocation domain size is greater than one.

17783 Scheduling Documentation

17784 If _POSIX_PRIORITY_SCHEDULING is defined, then any scheduling policies beyond
17785 TSP SCHED_OTHER, SCHED_FIFO, SCHED_RR, and SCHED_SPORADIC, as well as the effects of
17786 the scheduling policies indicated by these other values, and the attributes required in order to
17787 support such a policy, are implementation-defined. Furthermore, the implementation shall
17788 document the effect of all processor scheduling allocation domain values supported for these
17789 policies.

17790 2.9.5 Thread Cancellation

17791 The thread cancellation mechanism allows a thread to terminate the execution of any other
17792 thread in the process in a controlled manner. The target thread (that is, the one that is being
17793 canceled) is allowed to hold cancellation requests pending in a number of ways and to perform
17794 application-specific cleanup processing when the notice of cancellation is acted upon.

17795 Cancellation is controlled by the cancellation control functions. Each thread maintains its own
17796 cancelability state. Cancellation may only occur at cancellation points or when the thread is
17797 asynchronously cancelable.

17798 The thread cancellation mechanism described in this section depends upon programs having set
17799 *deferred* cancelability state, which is specified as the default. Applications shall also carefully
17800 follow static lexical scoping rules in their execution behavior. For example, use of *setjmp()*,
17801 *return*, *goto*, and so on, to leave user-defined cancellation scopes without doing the necessary
17802 scope pop operation results in undefined behavior.

17803 Use of asynchronous cancelability while holding resources which potentially need to be released
17804 may result in resource loss. Similarly, cancellation scopes may only be safely manipulated
17805 (pushed and popped) when the thread is in the *deferred* or *disabled* cancelability states.

17806 2.9.5.1 Cancelability States

17807 The cancelability state of a thread determines the action taken upon receipt of a cancellation
17808 request. The thread may control cancellation in a number of ways.

17809 Each thread maintains its own cancelability state, which may be encoded in two bits:

- 17810 1. Cancelability-Enable: When cancelability is PTHREAD_CANCEL_DISABLE (as defined
17811 in the Base Definitions volume of POSIX.1-2008, **<pthread.h>**), cancellation requests
17812 against the target thread are held pending. By default, cancelability is set to
17813 PTHREAD_CANCEL_ENABLE (as defined in **<pthread.h>**).

17814 2. Cancelability Type: When cancelability is enabled and the cancelability type is
 17815 PTHREAD_CANCEL_ASYNCHRONOUS (as defined in **<pthread.h>**), new or pending
 17816 cancellation requests may be acted upon at any time. When cancelability is enabled and
 17817 the cancelability type is PTHREAD_CANCEL_DEFERRED (as defined in **<pthread.h>**),
 17818 cancellation requests are held pending until a cancellation point (see below) is reached. If
 17819 cancelability is disabled, the setting of the cancelability type has no immediate effect as all
 17820 cancellation requests are held pending; however, once cancelability is enabled again the
 17821 new type is in effect. The cancelability type is PTHREAD_CANCEL_DEFERRED in all
 17822 newly created threads including the thread in which *main()* was first invoked.

17823 2.9.5.2 Cancellation Points

17824 Cancellation points shall occur when a thread is executing the following functions:

17825 <i>accept()</i>	<i>nanosleep()</i>	<i>select()</i>
17826 <i>aio_suspend()</i>	<i>open()</i>	<i>sem_timedwait()</i>
17827 <i>clock_nanosleep()</i>	<i>openat()</i>	<i>sem_wait()</i>
17828 <i>close()</i>	<i>pause()</i>	<i>send()</i>
17829 <i>connect()</i>	<i>poll()</i>	<i>sendmsg()</i>
17830 <i>creat()</i>	<i>pread()</i>	<i>sendto()</i>
17831 <i>fcntl()</i> [†]	<i>pselect()</i>	<i>sigsuspend()</i>
17832 <i>fdatasync()</i>	<i>pthread_cond_timedwait()</i>	<i>sigtimedwait()</i>
17833 <i>fsync()</i>	<i>pthread_cond_wait()</i>	<i>sigwait()</i>
17834 <i>getmsg()</i>	<i>pthread_join()</i>	<i>sigwaitinfo()</i>
17835 <i>getpmsg()</i>	<i>pthread_testcancel()</i>	<i>sleep()</i>
17836 <i>lockf()</i> ^{††}	<i>putmsg()</i>	<i>system()</i>
17837 <i>mq_receive()</i>	<i>putpmsg()</i>	<i>tcdrain()</i>
17838 <i>mq_send()</i>	<i>pwrite()</i>	<i>wait()</i>
17839 <i>mq_timedreceive()</i>	<i>read()</i>	<i>waitid()</i>
17840 <i>mq_timedsend()</i>	<i>readv()</i>	<i>waitpid()</i>
17841 <i>msgrcv()</i>	<i>recv()</i>	<i>write()</i>
17842 <i>msgsnd()</i>	<i>recvfrom()</i>	<i>writv()</i>
17843 <i>msync()</i>	<i>recvmsg()</i>	

17844 † When the *cmd* argument is F_SETLK.

17845 †† When the *function* argument is F_LOCK.

17846 A cancellation point may also occur when a thread is executing the following functions:

17847	<i>access()</i>	<i>fprintf()</i>	<i>getprotobyname()</i>
17848	<i>asctime()</i>	<i>fputc()</i>	<i>getprotoent()</i>
17849	<i>asctime_r()</i>	<i>fputs()</i>	<i>getpwent()</i>
17850	<i>catclose()</i>	<i>fputwc()</i>	<i>getpwnam()</i>
17851	<i>catgets()</i>	<i>fputws()</i>	<i>getpwnam_r()</i>
17852	<i>catopen()</i>	<i>fread()</i>	<i>getpwuid()</i>
17853	<i>chmod()</i>	<i>freopen()</i>	<i>getpwuid_r()</i>
17854	<i>chown()</i>	<i>fscanf()</i>	<i>gets()</i>
17855	<i>closedir()</i>	<i>fseek()</i>	<i>getservbyname()</i>
17856	<i>closelog()</i>	<i>fseeko()</i>	<i>getservbyport()</i>
17857	<i>ctermid()</i>	<i>fsetpos()</i>	<i>getservent()</i>
17858	<i>ctime()</i>	<i>fstat()</i>	<i>getutxent()</i>
17859	<i>ctime_r()</i>	<i>fstatat()</i>	<i>getutxid()</i>
17860	<i>dbm_close()</i>	<i>ftell()</i>	<i>getutxline()</i>
17861	<i>dbm_delete()</i>	<i>ftello()</i>	<i>getwc()</i>
17862	<i>dbm_fetch()</i>	<i>ftw()</i>	<i>getwchar()</i>
17863	<i>dbm_nextkey()</i>	<i>futimens()</i>	<i>glob()</i>
17864	<i>dbm_open()</i>	<i>fwprintf()</i>	<i>iconv_close()</i>
17865	<i>dbm_store()</i>	<i>fwrite()</i>	<i>iconv_open()</i>
17866	<i>dlclose()</i>	<i>fwscanf()</i>	<i>ioctl()</i>
17867	<i>dlopen()</i>	<i>getaddrinfo()</i>	<i>link()</i>
17868	<i>dprintf()</i>	<i>getc()</i>	<i>linkat()</i>
17869	<i>endgrent()</i>	<i>getc_unlocked()</i>	<i>lio_listio()</i>
17870	<i>endhostent()</i>	<i>getchar()</i>	<i>localtime()</i>
17871	<i>endnetent()</i>	<i>getchar_unlocked()</i>	<i>localtime_r()</i>
17872	<i>endprotoent()</i>	<i>getcwd()</i>	<i>lockf()</i>
17873	<i>endpwent()</i>	<i>getdate()</i>	<i>lseek()</i>
17874	<i>endservent()</i>	<i>getdelim()</i>	<i>lstat()</i>
17875	<i>endutxent()</i>	<i>getgrent()</i>	<i>mkdir()</i>
17876	<i>faccessat()</i>	<i>getgrgid()</i>	<i>mkdirat()</i>
17877	<i>fchmod()</i>	<i>getgrgid_r()</i>	<i>mkdtemp()</i>
17878	<i>fchmodat()</i>	<i>getgrnam()</i>	<i>mkfifo()</i>
17879	<i>fchown()</i>	<i>getgrnam_r()</i>	<i>mkfifoat()</i>
17880	<i>fchownat()</i>	<i>gethostent()</i>	<i>mknod()</i>
17881	<i>fclose()</i>	<i>gethostid()</i>	<i>mknodat()</i>
17882	<i>fcntl()</i> †	<i>gethostname()</i>	<i>mkstemp()</i>
17883	<i>fflush()</i>	<i>getline()</i>	<i>mktime()</i>
17884	<i>fgetc()</i>	<i>getlogin()</i>	<i>nftw()</i>
17885	<i>fgetpos()</i>	<i>getlogin_r()</i>	<i>opendir()</i>
17886	<i>fgets()</i>	<i>getnameinfo()</i>	<i>openlog()</i>
17887	<i>fgetwc()</i>	<i>getnetbyaddr()</i>	<i>pathconf()</i>
17888	<i>fgetws()</i>	<i>getnetbyname()</i>	<i>pclose()</i>
17889	<i>fntmsg()</i>	<i>getnetent()</i>	<i>perror()</i>
17890	<i>fopen()</i>	<i>getopt()</i> ††	<i>popen()</i>
17891	<i>fpathconf()</i>	<i>getprotobyname()</i>	<i>posix_fadvise()</i>

17892	<i>posix_fallocate()</i>	<i>putc_unlocked()</i>	<i>strerror_l()</i>
17893	<i>posix_madvise()</i>	<i>putchar()</i>	<i>strerror_r()</i>
17894	<i>posix_openpt()</i>	<i>putchar_unlocked()</i>	<i>strftime()</i>
17895	<i>posix_spawn()</i>	<i>puts()</i>	<i>strftime_l()</i>
17896	<i>posix_spawnnp()</i>	<i>pututxline()</i>	<i>symlink()</i>
17897	<i>posix_trace_clear()</i>	<i>putwc()</i>	<i>symlinkat()</i>
17898	<i>posix_trace_close()</i>	<i>putwchar()</i>	<i>sync()</i>
17899	<i>posix_trace_create()</i>	<i>readdir()</i>	<i>syslog()</i>
17900	<i>posix_trace_create_withlog()</i>	<i>readdir_r()</i>	<i>tmpfile()</i>
17901	<i>posix_trace_eventtypelist_getnext_id()</i>	<i>readlink()</i>	<i>tmpnam()</i>
17902	<i>posix_trace_eventtypelist_rewind()</i>	<i>readlinkat()</i>	<i>ttyname()</i>
17903	<i>posix_trace_flush()</i>	<i>remove()</i>	<i>ttyname_r()</i>
17904	<i>posix_trace_get_attr()</i>	<i>rename()</i>	<i>tzset()</i>
17905	<i>posix_trace_get_filter()</i>	<i>renameat()</i>	<i>ungetc()</i>
17906	<i>posix_trace_get_status()</i>	<i>rewind()</i>	<i>ungetwc()</i>
17907	<i>posix_trace_getnext_event()</i>	<i>rewinddir()</i>	<i>unlink()</i>
17908	<i>posix_trace_open()</i>	<i>scandir()</i>	<i>unlinkat()</i>
17909	<i>posix_trace_rewind()</i>	<i>scanf()</i>	<i>utime()</i>
17910	<i>posix_trace_set_filter()</i>	<i>seekdir()</i>	<i>utimensat()</i>
17911	<i>posix_trace_shutdown()</i>	<i>semop()</i>	<i>utimes()</i>
17912	<i>posix_trace_timedgetnext_event()</i>	<i>setgrent()</i>	<i>vdprintf()</i>
17913	<i>posix_typed_mem_open()</i>	<i>sethostent()</i>	<i>vfprintf()</i>
17914	<i>printf()</i>	<i>setnetent()</i>	<i>vfscanf()</i>
17915	<i>psiginfo()</i>	<i>setprotoent()</i>	<i>vprintf()</i>
17916	<i>psignal()</i>	<i>setpwent()</i>	<i>vwprintf()</i>
17917	<i>pthread_rwlock_rdlock()</i>	<i>setservent()</i>	<i>wcsftime()</i>
17918	<i>pthread_rwlock_timedrdlock()</i>	<i>setutxent()</i>	<i>wordexp()</i>
17919	<i>pthread_rwlock_timedwrlock()</i>	<i>sigpause()</i>	<i>wprintf()</i>
17920	<i>pthread_rwlock_wrlock()</i>	<i>stat()</i>	<i>wscanf()</i>
17921	<i>putc()</i>	<i>strerror()</i>	

An implementation shall not introduce cancellation points into any other functions specified in this volume of POSIX.1-2008.

The side-effects of acting upon a cancellation request while suspended during a call of a function are the same as the side-effects that may be seen in a single-threaded program when a call to a function is interrupted by a signal and the given function returns [EINTR]. Any such side-effects occur before any cancellation cleanup handlers are called.

Whenever a thread has cancelability enabled and a cancellation request has been made with that thread as the target, and the thread then calls any function that is a cancellation point (such as *pthread_testcancel()* or *read()*), the cancellation request shall be acted upon before the function returns. If a thread has cancelability enabled and a cancellation request is made with the thread as a target while the thread is suspended at a cancellation point, the thread shall be awakened and the cancellation request shall be acted upon. It is unspecified whether the cancellation request is acted upon or whether the cancellation request remains pending and the thread resumes normal execution if:

- The thread is suspended at a cancellation point and the event for which it is waiting occurs

† For any value of the *cmd* argument.

†† If *opterr* is non-zero.

- A specified timeout expired

before the cancellation request is acted upon.

2.9.5.3 Thread Cancellation Cleanup Handlers

Each thread maintains a list of cancellation cleanup handlers. The programmer uses the *pthread_cleanup_push()* and *pthread_cleanup_pop()* functions to place routines on and remove routines from this list.

When a cancellation request is acted upon, or when a thread calls *pthread_exit()*, the thread first disables cancellation by setting its cancelability state to `PTHREAD_CANCEL_DISABLE` and its cancelability type to `PTHREAD_CANCEL_DEFERRED`. The cancelability state shall remain set to `PTHREAD_CANCEL_DISABLE` until the thread has terminated. The behavior is undefined if a cancellation cleanup handler or thread-specific data destructor routine changes the cancelability state to `PTHREAD_CANCEL_ENABLE`.

The routines in the thread's list of cancellation cleanup handlers are invoked one by one in LIFO sequence; that is, the last routine pushed onto the list (Last In) is the first to be invoked (First Out). When the cancellation cleanup handler for a scope is invoked, the storage for that scope remains valid. If the last cancellation cleanup handler returns, thread-specific data destructors (if any) associated with thread-specific data keys for which the thread has non-NULL values will be run, in unspecified order, as described for *pthread_key_create()*.

After all cancellation cleanup handlers and thread-specific data destructors have returned, thread execution is terminated. If the thread has terminated because of a call to *pthread_exit()*, the *value_ptr* argument is made available to any threads joining with the target. If the thread has terminated by acting on a cancellation request, a status of `PTHREAD_CANCELED` is made available to any threads joining with the target. The symbolic constant `PTHREAD_CANCELED` expands to a constant expression of type `(void *)` whose value matches no pointer to an object in memory nor the value `NULL`.

A side-effect of acting upon a cancellation request while in a condition variable wait is that the mutex is re-acquired before calling the first cancellation cleanup handler. In addition, the thread is no longer considered to be waiting for the condition and the thread shall not have consumed any pending condition signals on the condition.

A cancellation cleanup handler cannot exit via *longjmp()* or *siglongjmp()*.

2.9.5.4 Async-Cancel Safety

The *pthread_cancel()*, *pthread_setcancelstate()*, and *pthread_setcanceltype()* functions are defined to be async-cancel safe.

No other functions in this volume of POSIX.1-2008 are required to be async-cancel-safe.

2.9.6 Thread Read-Write Locks

Multiple readers, single writer (read-write) locks allow many threads to have simultaneous read-only access to data while allowing only one thread to have exclusive write access at any given time. They are typically used to protect data that is read more frequently than it is changed.

One or more readers acquire read access to the resource by performing a read lock operation on the associated read-write lock. A writer acquires exclusive write access by performing a write lock operation. Basically, all readers exclude any writers and a writer excludes all readers and

17981 any other writers.

17982 A thread that has blocked on a read-write lock (for example, has not yet returned from a
17983 *pthread_rwlock_rdlock()* or *pthread_rwlock_wrlock()* call) shall not prevent any unblocked thread
17984 that is eligible to use the same processing resources from eventually making forward progress in
17985 its execution. Eligibility for processing resources shall be determined by the scheduling policy.

17986 Read-write locks can be used to synchronize threads in the current process and other processes if
17987 they are allocated in memory that is writable and shared among the cooperating processes and
17988 have been initialized for this behavior.

17989 2.9.7 Thread Interactions with Regular File Operations

17990 All of the following functions shall be atomic with respect to each other in the effects specified in
17991 POSIX.1-2008 when they operate on regular files or symbolic links:

17992	<i>chmod()</i>	<i>fchownat()</i>	<i>lseek()</i>	<i>readv()</i>	<i>unlink()</i>
17993	<i>chown()</i>	<i>fcntl()</i>	<i>lstat()</i>	<i>pwrite()</i>	<i>unlinkat()</i>
17994	<i>close()</i>	<i>fstat()</i>	<i>open()</i>	<i>rename()</i>	<i>utime()</i>
17995	<i>creat()</i>	<i>fstatat()</i>	<i>openat()</i>	<i>renameat()</i>	<i>utimensat()</i>
17996	<i>dup2()</i>	<i>ftruncate()</i>	<i>pread()</i>	<i>stat()</i>	<i>utimes()</i>
17997	<i>fchmod()</i>	<i>lchown()</i>	<i>read()</i>	<i>symlink()</i>	<i>write()</i>
17998	<i>fchmodat()</i>	<i>link()</i>	<i>readlink()</i>	<i>symlinkat()</i>	<i>writev()</i>
17999	<i>fchown()</i>	<i>linkat()</i>	<i>readlinkat()</i>	<i>truncate()</i>	

18000 If two threads each call one of these functions, each call shall either see all of the specified effects
18001 of the other call, or none of them.

18002 2.9.8 Use of Application-Managed Thread Stacks

18003 An “application-managed thread stack” is a region of memory allocated by the application—for
18004 example, memory returned by the *malloc()* or *mmap()* functions—and designated as a stack
18005 through the act of passing the address and size of the stack, respectively, as the *stackaddr* and
18006 *stacksize* arguments to *pthread_attr_setstack()*. Application-managed stacks allow the application
18007 to precisely control the placement and size of a stack.

18008 The application grants to the implementation permanent ownership of and control over the
18009 application-managed stack when the attributes object in which the *stack* or *stackaddr* attribute has
18010 been set is used, either by presenting that attribute’s object as the *attr* argument in a call to
18011 *pthread_create()* that completes successfully, or by storing a pointer to the attributes object in the
18012 *sigev_notify_attributes* member of a **struct sigevent** and passing that **struct sigevent** to a function
18013 accepting such argument that completes successfully. The application may thereafter utilize the
18014 memory within the stack only within the normal context of stack usage within or properly
18015 synchronized with a thread that has been scheduled by the implementation with stack pointer
18016 value(s) that are within the range of that stack. In particular, the region of memory cannot be
18017 freed, nor can it be later specified as the stack for another thread.

18018 When specifying an attributes object with an application-managed stack through the
18019 *sigev_notify_attributes* member of a **struct sigevent**, the results are undefined if the requested
18020 signal is generated multiple times (as for a repeating timer).

18021 Until an attributes object in which the *stack* or *stackaddr* attribute has been set is used, the
18022 application retains ownership of and control over the memory allocated to the stack. It may free
18023 or reuse the memory as long as it either deletes the attributes object, or before using the

attributes object replaces the stack by making an additional call to `pthread_attr_setstack()`, that was used originally to designate the stack. There is no mechanism to retract the reference to an application-managed stack by an existing attributes object.

Once an attributes object with an application-managed stack has been used, that attributes object cannot be used again by a subsequent call to `pthread_create()` or any function accepting a **struct sigevent** with `sigev_notify_attributes` containing a pointer to the attributes object, without designating an unused application-managed stack by making an additional call to `pthread_attr_setstack()`.

2.10 Sockets

A socket is an endpoint for communication using the facilities described in this section. A socket is created with a specific socket type, described in [Section 2.10.6](#) (on page 523), and is associated with a specific protocol, detailed in [Section 2.10.3](#). A socket is accessed via a file descriptor obtained when the socket is created.

2.10.1 Address Families

All network protocols are associated with a specific address family. An address family provides basic services to the protocol implementation to allow it to function within a specific network environment. These services may include packet fragmentation and reassembly, routing, addressing, and basic transport. An address family is normally comprised of a number of protocols, one per socket type. Each protocol is characterized by an abstract socket type. It is not required that an address family support all socket types. An address family may contain multiple protocols supporting the same socket abstraction.

[Section 2.10.17](#) (on page 530), [Section 2.10.19](#) (on page 531), and [Section 2.10.20](#) (on page 531), respectively, describe the use of sockets for local UNIX connections, for Internet protocols based on IPv4, and for Internet protocols based on IPv6.

2.10.2 Addressing

An address family defines the format of a socket address. All network addresses are described using a general structure, called a **sockaddr**, as defined in the Base Definitions volume of POSIX.1-2008, `<sys/socket.h>`. However, each address family imposes finer and more specific structure, generally defining a structure with fields specific to the address family. The field `sa_family` in the **sockaddr** structure contains the address family identifier, specifying the format of the `sa_data` area. The size of the `sa_data` area is unspecified.

2.10.3 Protocols

A protocol supports one of the socket abstractions detailed in [Section 2.10.6](#) (on page 523). Selecting a protocol involves specifying the address family, socket type, and protocol number to the `socket()` function. Certain semantics of the basic socket abstractions are protocol-specific. All protocols are expected to support the basic model for their particular socket type, but may, in addition, provide non-standard facilities or extensions to a mechanism.

2.10.4 Routing

Sockets provides packet routing facilities. A routing information database is maintained, which is used in selecting the appropriate network interface when transmitting packets.

2.10.5 Interfaces

Each network interface in a system corresponds to a path through which messages can be sent and received. A network interface usually has a hardware device associated with it, though certain interfaces such as the loopback interface, do not.

2.10.6 Socket Types

A socket is created with a specific type, which defines the communication semantics and which allows the selection of an appropriate communication protocol. Four types are defined: SOCK_DGRAM, SOCK_RAW, SOCK_SEQPACKET, and SOCK_STREAM. Implementations may specify additional socket types.

The SOCK_STREAM socket type provides reliable, sequenced, full-duplex octet streams between the socket and a peer to which the socket is connected. A socket of type SOCK_STREAM must be in a connected state before any data may be sent or received. Record boundaries are not maintained; data sent on a stream socket using output operations of one size may be received using input operations of smaller or larger sizes without loss of data. Data may be buffered; successful return from an output function does not imply that the data has been delivered to the peer or even transmitted from the local system. If data cannot be successfully transmitted within a given time then the connection is considered broken, and subsequent operations shall fail. A SIGPIPE signal is raised if a thread attempts to send data on a broken stream (one that is no longer connected), except that the signal is suppressed if the MSG_NOSIGNAL flag is used in calls to *send()*, *sendto()*, and *sendmsg()*. Support for an out-of-band data transmission facility is protocol-specific.

The SOCK_SEQPACKET socket type is similar to the SOCK_STREAM type, and is also connection-oriented. The only difference between these types is that record boundaries are maintained using the SOCK_SEQPACKET type. A record can be sent using one or more output operations and received using one or more input operations, but a single operation never transfers parts of more than one record. Record boundaries are visible to the receiver via the MSG_EOR flag in the received message flags returned by the *recvmsg()* function. It is protocol-specific whether a maximum record size is imposed.

The SOCK_DGRAM socket type supports connectionless data transfer which is not necessarily acknowledged or reliable. Datagrams may be sent to the address specified (possibly multicast or broadcast) in each output operation, and incoming datagrams may be received from multiple sources. The source address of each datagram is available when receiving the datagram. An application may also pre-specify a peer address, in which case calls to output functions that do not specify a peer address shall send to the pre-specified peer. If a peer has been specified, only datagrams from that peer shall be received. A datagram must be sent in a single output operation, and must be received in a single input operation. The maximum size of a datagram is protocol-specific; with some protocols, the limit is implementation-defined. Output datagrams may be buffered within the system; thus, a successful return from an output function does not guarantee that a datagram is actually sent or received. However, implementations should attempt to detect any errors possible before the return of an output function, reporting any error by an unsuccessful return value.

18105 RS The SOCK_RAW socket type is similar to the SOCK_DGRAM type. It differs in that it is
18106 normally used with communication providers that underlie those used for the other socket
18107 types. For this reason, the creation of a socket with type SOCK_RAW shall require appropriate
18108 privileges. The format of datagrams sent and received with this socket type generally include
18109 specific protocol headers, and the formats are protocol-specific and implementation-defined.

18110 2.10.7 Socket I/O Mode

18111 The I/O mode of a socket is described by the O_NONBLOCK file status flag which pertains to
18112 the open file description for the socket. This flag is initially off when a socket is created, but may
18113 be set and cleared by the use of the F_SETFL command of the *fcntl()* function.

18114 When the O_NONBLOCK flag is set, certain functions that would normally block until they are
18115 complete shall return immediately.

18116 The *bind()* function initiates an address assignment and shall return without blocking when
18117 O_NONBLOCK is set; if the socket address cannot be assigned immediately, *bind()* shall return
18118 the [EINPROGRESS] error to indicate that the assignment was initiated successfully, but that it
18119 has not yet completed.

18120 The *connect()* function initiates a connection and shall return without blocking when
18121 O_NONBLOCK is set; it shall return the error [EINPROGRESS] to indicate that the connection
18122 was initiated successfully, but that it has not yet completed.

18123 Data transfer operations (the *read()*, *write()*, *send()*, and *recv()* functions) shall complete
18124 immediately, transfer only as much as is available, and then return without blocking, or return
18125 an error indicating that no transfer could be made without blocking.

18126 2.10.8 Socket Owner

18127 The owner of a socket is unset when a socket is created. The owner may be set to a process ID or
18128 process group ID using the F_SETOWN command of the *fcntl()* function.

18129 2.10.9 Socket Queue Limits

18130 The transmit and receive queue sizes for a socket are set when the socket is created. The default
18131 sizes used are both protocol-specific and implementation-defined. The sizes may be changed
18132 using the *setsockopt()* function.

18133 2.10.10 Pending Error

18134 Errors may occur asynchronously, and be reported to the socket in response to input from the
18135 network protocol. The socket stores the pending error to be reported to a user of the socket at the
18136 next opportunity. The error is returned in response to a subsequent *send()*, *recv()*, or *getsockopt()*
18137 operation on the socket, and the pending error is then cleared.

2.10.11 Socket Receive Queue

A socket has a receive queue that buffers data when it is received by the system until it is removed by a receive call. Depending on the type of the socket and the communication provider, the receive queue may also contain ancillary data such as the addressing and other protocol data associated with the normal data in the queue, and may contain out-of-band or expedited data. The limit on the queue size includes any normal, out-of-band data, datagram source addresses, and ancillary data in the queue. The description in this section applies to all sockets, even though some elements cannot be present in some instances.

The contents of a receive buffer are logically structured as a series of data segments with associated ancillary data and other information. A data segment may contain normal data or out-of-band data, but never both. A data segment may complete a record if the protocol supports records (always true for types SOCK_SEQPACKET and SOCK_DGRAM). A record may be stored as more than one segment; the complete record might never be present in the receive buffer at one time, as a portion might already have been returned to the application, and another portion might not yet have been received from the communications provider. A data segment may contain ancillary protocol data, which is logically associated with the segment. Ancillary data is received as if it were queued along with the first normal data octet in the segment (if any). A segment may contain ancillary data only, with no normal or out-of-band data. For the purposes of this section, a datagram is considered to be a data segment that terminates a record, and that includes a source address as a special type of ancillary data. Data segments are placed into the queue as data is delivered to the socket by the protocol. Normal data segments are placed at the end of the queue as they are delivered. If a new segment contains the same type of data as the preceding segment and includes no ancillary data, and if the preceding segment does not terminate a record, the segments are logically merged into a single segment.

The receive queue is logically terminated if an end-of-file indication has been received or a connection has been terminated. A segment shall be considered to be terminated if another segment follows it in the queue, if the segment completes a record, or if an end-of-file or other connection termination has been reported. The last segment in the receive queue shall also be considered to be terminated while the socket has a pending error to be reported.

A receive operation shall never return data or ancillary data from more than one segment.

2.10.12 Socket Out-of-Band Data State

The handling of received out-of-band data is protocol-specific. Out-of-band data may be placed in the socket receive queue, either at the end of the queue or before all normal data in the queue. In this case, out-of-band data is returned to an application program by a normal receive call. Out-of-band data may also be queued separately rather than being placed in the socket receive queue, in which case it shall be returned only in response to a receive call that requests out-of-band data. It is protocol-specific whether an out-of-band data mark is placed in the receive queue to demarcate data preceding the out-of-band data and following the out-of-band data. An out-of-band data mark is logically an empty data segment that cannot be merged with other segments in the queue. An out-of-band data mark is never returned in response to an input operation. The *socketatmark()* function can be used to test whether an out-of-band data mark is the first element in the queue. If an out-of-band data mark is the first element in the queue when an input function is called without the MSG_PEEK option, the mark is removed from the queue and the following data (if any) is processed as if the mark had not been present.

2.10.13 Connection Indication Queue

Sockets that are used to accept incoming connections maintain a queue of outstanding connection indications. This queue is a list of connections that are awaiting acceptance by the application; see *listen()*.

2.10.14 Signals

One category of event at the socket interface is the generation of signals. These signals report protocol events or process errors relating to the state of the socket. The generation or delivery of a signal does not change the state of the socket, although the generation of the signal may have been caused by a state change.

The SIGPIPE signal shall be sent to a thread that attempts to send data on a socket that is no longer able to send (one that is no longer connected), except that the signal is suppressed if the MSG_NOSIGNAL flag is used in calls to *send()*, *sendto()*, and *sendmsg()*. Regardless of whether the generation of the signal is suppressed, the send operation shall fail with the [EPIPE] error.

If a socket has an owner, the SIGURG signal is sent to the owner of the socket when it is notified of expedited or out-of-band data. The socket state at this time is protocol-dependent, and the status of the socket is specified in [Section 2.10.17](#) (on page 530), [Section 2.10.19](#) (on page 531), and [Section 2.10.20](#) (on page 531). Depending on the protocol, the expedited data may or may not have arrived at the time of signal generation.

2.10.15 Asynchronous Errors

If any of the following conditions occur asynchronously for a socket, the corresponding value listed below shall become the pending error for the socket:

[ECONNABORTED]

The connection was aborted locally.

[ECONNREFUSED]

For a connection-mode socket attempting a non-blocking connection, the attempt to connect was forcefully rejected. For a connectionless-mode socket, an attempt to deliver a datagram was forcefully rejected.

[ECONNRESET]

The peer has aborted the connection.

[EHOSTDOWN]

The destination host has been determined to be down or disconnected.

[EHOSTUNREACH]

The destination host is not reachable.

[EMSGSIZE]

For a connectionless-mode socket, the size of a previously sent datagram prevented delivery.

[ENETDOWN]

The local network connection is not operational.

[ENETRESET]

The connection was aborted by the network.

18223 [ENETUNREACH]
18224 The destination network is not reachable.

18225 **2.10.16 Use of Options**

18226 There are a number of socket options which either specialize the behavior of a socket or provide
18227 useful information. These options may be set at different protocol levels and are always present
18228 at the uppermost “socket” level.

18229 Socket options are manipulated by two functions, *getsockopt()* and *setsockopt()*. These functions
18230 allow an application program to customize the behavior and characteristics of a socket to
18231 provide the desired effect.

18232 All of the options have default values. The type and meaning of these values is defined by the
18233 protocol level to which they apply. Instead of using the default values, an application program
18234 may choose to customize one or more of the options. However, in the bulk of cases, the default
18235 values are sufficient for the application.

18236 Some of the options are used to enable or disable certain behavior within the protocol modules
18237 (for example, turn on debugging) while others may be used to set protocol-specific information
18238 (for example, IP time-to-live on all the application’s outgoing packets). As each of the options is
18239 introduced, its effect on the underlying protocol modules is described.

18240 [Table 2-1](#) shows the value for the socket level.

18241 **Table 2-1** Value of Level for Socket Options

18242	Name	Description
18243	SOL_SOCKET	Options are intended for the sockets level.

18244 [Table 2-2](#) (on page 528) lists those options present at the socket level; that is, when the *level*
18245 parameter of the *getsockopt()* or *setsockopt()* function is SOL_SOCKET, the types of the option
18246 value parameters associated with each option, and a brief synopsis of the meaning of the option
18247 value parameter. Unless otherwise noted, each may be examined with *getsockopt()* and set with
18248 *setsockopt()* on all types of socket. Options at other protocol levels vary in format and name.

Table 2-2 Socket-Level Options

Option	Parameter Type	Parameter Meaning
SO_ACCEPTCONN	int	Non-zero indicates that socket listening is enabled (<i>getsockopt()</i> only).
SO_BROADCAST	int	Non-zero requests permission to transmit broadcast datagrams (SOCK_DGRAM sockets only).
SO_DEBUG	int	Non-zero requests debugging in underlying protocol modules.
SO_DONTROUTE	int	Non-zero requests bypass of normal routing; route based on destination address only.
SO_ERROR	int	Requests and clears pending error information on the socket (<i>getsockopt()</i> only).
SO_KEEPAIVE	int	Non-zero requests periodic transmission of keepalive messages (protocol-specific).
SO_LINGER	struct linger	Specify actions to be taken for queued, unsent data on <i>close()</i> : linger on/off and linger time in seconds.
SO_OOINLINE	int	Non-zero requests that out-of-band data be placed into normal data input queue as received.
SO_RCVBUF	int	Size of receive buffer (in bytes).
SO_RCVLOWAT	int	Minimum amount of data to return to application for input operations (in bytes).
SO_RCVTIMEO	struct timeval	Timeout value for a socket receive operation.
SO_REUSEADDR	int	Non-zero requests reuse of local addresses in <i>bind()</i> (protocol-specific).
SO_SNDBUF	int	Size of send buffer (in bytes).
SO_SNDLOWAT	int	Minimum amount of data to send for output operations (in bytes).
SO_SNDTIMEO	struct timeval	Timeout value for a socket send operation.
SO_TYPE	int	Identify socket type (<i>getsockopt()</i> only).

The SO_ACCEPTCONN option is used only on *getsockopt()*. When this option is specified, *getsockopt()* shall report whether socket listening is enabled for the socket. A value of zero shall indicate that socket listening is disabled; non-zero that it is enabled. SO_ACCEPTCONN has no default value.

The SO_BROADCAST option requests permission to send broadcast datagrams on the socket. Support for SO_BROADCAST is protocol-specific. The default for SO_BROADCAST is that the ability to send broadcast datagrams on a socket is disabled.

The SO_DEBUG option enables debugging in the underlying protocol modules. This can be useful for tracing the behavior of the underlying protocol modules during normal system operation. The semantics of the debug reports are implementation-defined. The default value for SO_DEBUG is for debugging to be turned off.

The SO_DONTROUTE option requests that outgoing messages bypass the standard routing facilities. The destination must be on a directly-connected network, and messages are directed to the appropriate network interface according to the destination address. It is protocol-specific whether this option has any effect and how the outgoing network interface is chosen. Support for this option with each protocol is implementation-defined.

The SO_ERROR option is used only on *getsockopt()*. When this option is specified, *getsockopt()*

shall return any pending error on the socket and clear the error status. It shall return a value of 0 if there is no pending error. `SO_ERROR` may be used to check for asynchronous errors on connected connectionless-mode sockets or for other types of asynchronous errors. `SO_ERROR` has no default value.

The `SO_KEEPALIVE` option enables the periodic transmission of messages on a connected socket. The behavior of this option is protocol-specific. On a connection-mode socket for which a connection has been established, if `SO_KEEPALIVE` is enabled and the connected socket fails to respond to the keep-alive messages, the connection shall be broken. The default value for `SO_KEEPALIVE` is zero, specifying that this capability is turned off.

The `SO_LINGER` option controls the action of the interface when unsent messages are queued on a socket and a `close()` is performed. The details of this option are protocol-specific. If `SO_LINGER` is enabled, the system shall block the calling thread during `close()` until it can transmit the data or until the end of the interval indicated by the `l_linger` member, whichever comes first. If `SO_LINGER` is not specified, and `close()` is issued, the system handles the call in a way that allows the calling thread to continue as quickly as possible. The default value for `SO_LINGER` is zero, or off, for the `l_onoff` element of the option value and zero seconds for the linger time specified by the `l_linger` element.

The `SO_OOBINLINE` option is valid only on protocols that support out-of-band data. The `SO_OOBINLINE` option requests that out-of-band data be placed in the normal data input queue as received; it is then accessible using the `read()` or `recv()` functions without the `MSG_OOB` flag set. The default for `SO_OOBINLINE` is off; that is, for out-of-band data not to be placed in the normal data input queue.

The `SO_RCVBUF` option requests that the buffer space allocated for receive operations on this socket be set to the value, in bytes, of the option value. Applications may wish to increase buffer size for high volume connections, or may decrease buffer size to limit the possible backlog of incoming data. The default value for the `SO_RCVBUF` option value is implementation-defined, and may vary by protocol.

The `SO_RCVLOWAT` option sets the minimum number of bytes to process for socket input operations. In general, receive calls block until any (non-zero) amount of data is received, then return the smaller of the amount available or the amount requested. The default value for `SO_RCVLOWAT` is 1, and does not affect the general case. If `SO_RCVLOWAT` is set to a larger value, blocking receive calls normally wait until they have received the smaller of the low water mark value or the requested amount. Receive calls may still return less than the low water mark if an error occurs, a signal is caught, or the type of data next in the receive queue is different from that returned (for example, out-of-band data). As mentioned previously, the default value for `SO_RCVLOWAT` is 1 byte. It is implementation-defined whether the `SO_RCVLOWAT` option can be set.

The `SO_RCVTIMEO` option is an option to set a timeout value for input operations. It accepts a **timeval** structure with the number of seconds and microseconds specifying the limit on how long to wait for an input operation to complete. If a receive operation has blocked for this much time without receiving additional data, it shall return with a partial count or `errno` shall be set to `[EAGAIN]` or `[EWOULDBLOCK]` if no data were received. The default for this option is the value zero, which indicates that a receive operation will not time out. It is implementation-defined whether the `SO_RCVTIMEO` option can be set.

The `SO_REUSEADDR` option indicates that the rules used in validating addresses supplied in a `bind()` should allow reuse of local addresses. Operation of this option is protocol-specific. The default value for `SO_REUSEADDR` is off; that is, reuse of local addresses is not permitted.

The `SO_SNDBUF` option requests that the buffer space allocated for send operations on this socket be set to the value, in bytes, of the option value. The default value for the `SO_SNDBUF`

option value is implementation-defined, and may vary by protocol.

The `SO_SNDLOWAT` option sets the minimum number of bytes to process for socket output operations. Most output operations process all of the data supplied by the call, delivering data to the protocol for transmission and blocking as necessary for flow control. Non-blocking output operations process as much data as permitted subject to flow control without blocking, but process no data if flow control does not allow the smaller of the send low water mark value or the entire request to be processed. A `select()` operation testing the ability to write to a socket shall return true only if the send low water mark could be processed. The default value for `SO_SNDLOWAT` is implementation-defined and protocol-specific. It is implementation-defined whether the `SO_SNDLOWAT` option can be set.

The `SO_SNDTIMEO` option is an option to set a timeout value for the amount of time that an output function shall block because flow control prevents data from being sent. As noted in [Table 2-2](#) (on page 528), the option value is a **timeval** structure with the number of seconds and microseconds specifying the limit on how long to wait for an output operation to complete. If a send operation has blocked for this much time, it shall return with a partial count or *errno* set to `[EAGAIN]` or `[EWOULDBLOCK]` if no data were sent. The default for this option is the value zero, which indicates that a send operation will not time out. It is implementation-defined whether the `SO_SNDTIMEO` option can be set.

The `SO_TYPE` option is used only on `getsockopt()`. When this option is specified, `getsockopt()` shall return the type of the socket (for example, `SOCK_STREAM`). This option is useful to servers that inherit sockets on start-up. `SO_TYPE` has no default value.

2.10.17 Use of Sockets for Local UNIX Connections

Support for UNIX domain sockets is mandatory.

UNIX domain sockets provide process-to-process communication in a single system.

2.10.17.1 Headers

The symbolic constant `AF_UNIX` defined in the `<sys/socket.h>` header is used to identify the UNIX domain address family. The `<sys/un.h>` header contains other definitions used in connection with UNIX domain sockets. See [XBD Chapter 13](#) (on page 219).

The **sockaddr_storage** structure defined in `<sys/socket.h>` shall be large enough to accommodate a **sockaddr_un** structure (see the `<sys/un.h>` header defined in [XBD Chapter 13](#), on page 219) and shall be aligned at an appropriate boundary so that pointers to it can be cast as pointers to **sockaddr_un** structures and used to access the fields of those structures without alignment problems. When a **sockaddr_storage** structure is cast as a **sockaddr_un** structure, the *ss_family* field maps onto the *sun_family* field.

2.10.18 Use of Sockets over Internet Protocols

When a socket is created in the Internet family with a protocol value of zero, the implementation shall use the protocol listed below for the type of socket created.

`SOCK_STREAM` `IPPROTO_TCP`.

`SOCK_DGRAM` `IPPROTO_UDP`.

18385 RS SOCK_RAW IPPROTO_RAW.

18386 SOCK_SEQPACKET Unspecified.

18387 RS A raw interface to IP is available by creating an Internet socket of type SOCK_RAW. The default
 18388 protocol for type SOCK_RAW shall be identified in the IP header with the value
 18389 IPPROTO_RAW. Applications should not use the default protocol when creating a socket with
 18390 type SOCK_RAW, but should identify a specific protocol by value. The ICMP control protocol is
 18391 accessible from a raw socket by specifying a value of IPPROTO_ICMP for protocol.

18392 2.10.19 Use of Sockets over Internet Protocols Based on IPv4

18393 Support for sockets over Internet protocols based on IPv4 is mandatory.

18394 2.10.19.1 Headers

18395 The symbolic constant AF_INET defined in the `<sys/socket.h>` header is used to identify the
 18396 IPv4 Internet address family. The `<netinet/in.h>` header contains other definitions used in
 18397 connection with IPv4 Internet sockets. See XBD [Chapter 13](#) (on page 219).

18398 The `sockaddr_storage` structure defined in `<sys/socket.h>` shall be large enough to
 18399 accommodate a `sockaddr_in` structure (see the `<netinet/in.h>` header defined in XBD [Chapter](#)
 18400 [13](#), on page 219) and shall be aligned at an appropriate boundary so that pointers to it can be
 18401 cast as pointers to `sockaddr_in` structures and used to access the fields of those structures
 18402 without alignment problems. When a `sockaddr_storage` structure is cast as a `sockaddr_in`
 18403 structure, the `ss_family` field maps onto the `sin_family` field.

18404 2.10.20 Use of Sockets over Internet Protocols Based on IPv6

18405 IP6 This section describes extensions to support sockets over Internet protocols based on IPv6. The
 18406 functionality described in this section shall be provided on implementations that support the
 18407 IPV6 option (and the rest of this section is not further shaded for this option).

18408 To enable smooth transition from IPv4 to IPv6, the features defined in this section may, in certain
 18409 circumstances, also be used in connection with IPv4; see [Section 2.10.20.2](#) (on page 532).

18410 2.10.20.1 Addressing

18411 IPv6 overcomes the addressing limitations of earlier versions by using 128-bit addresses instead
 18412 of 32-bit addresses. The IPv6 address architecture is described in RFC 2373.

18413 There are three kinds of IPv6 address:

18414 Unicast

18415 Identifies a single interface.

18416 A unicast address can be global, link-local (designed for use on a single link), or site-local
 18417 (designed for systems not connected to the Internet). Link-local and site-local addresses
 18418 need not be globally unique.

18419 Anycast

18420 Identifies a set of interfaces such that a packet sent to the address can be delivered to any
 18421 member of the set.

18422 An anycast address is similar to a unicast address; the nodes to which an anycast address is

assigned must be explicitly configured to know that it is an anycast address.

Multicast

Identifies a set of interfaces such that a packet sent to the address should be delivered to every member of the set.

An application can send multicast datagrams by simply specifying an IPv6 multicast address in the *address* argument of *sendto()*. To receive multicast datagrams, an application must join the multicast group (using *setsockopt()* with `IPV6_JOIN_GROUP`) and must bind to the socket the UDP port on which datagrams will be received. Some applications should also bind the multicast group address to the socket, to prevent other datagrams destined to that port from being delivered to the socket.

A multicast address can be global, node-local, link-local, site-local, or organization-local.

The following special IPv6 addresses are defined:

Unspecified

An address that is not assigned to any interface and is used to indicate the absence of an address.

Loopback

A unicast address that is not assigned to any interface and can be used by a node to send packets to itself.

Two sets of IPv6 addresses are defined to correspond to IPv4 addresses:

IPv4-compatible addresses

These are assigned to nodes that support IPv6 and can be used when traffic is “tunneled” through IPv4.

IPv4-mapped addresses

These are used to represent IPv4 addresses in IPv6 address format; see [Section 2.10.20.2](#).

Note that the unspecified address and the loopback address must not be treated as IPv4-compatible addresses.

2.10.20.2 Compatibility with IPv4

The API provides the ability for IPv6 applications to interoperate with applications using IPv4, by using IPv4-mapped IPv6 addresses. These addresses can be generated automatically by the *getaddrinfo()* function when the specified host has only IPv4 addresses.

Applications can use `AF_INET6` sockets to open TCP connections to IPv4 nodes, or send UDP packets to IPv4 nodes, by simply encoding the destination’s IPv4 address as an IPv4-mapped IPv6 address, and passing that address, within a **sockaddr_in6** structure, in the *connect()*, *sendto()*, or *sendmsg()* function. When applications use `AF_INET6` sockets to accept TCP connections from IPv4 nodes, or receive UDP packets from IPv4 nodes, the system shall return the peer’s address to the application in the *accept()*, *recvfrom()*, *recvmsg()*, or *getpeername()* function using a **sockaddr_in6** structure encoded this way. If a node has an IPv4 address, then the implementation shall allow applications to communicate using that address via an `AF_INET6` socket. In such a case, the address will be represented at the API by the corresponding IPv4-mapped IPv6 address. Also, the implementation may allow an `AF_INET6` socket bound to **in6addr_any** to receive inbound connections and packets destined to one of the node’s IPv4 addresses.

An application can use `AF_INET6` sockets to bind to a node’s IPv4 address by specifying the address as an IPv4-mapped IPv6 address in a **sockaddr_in6** structure in the *bind()* function. For an `AF_INET6` socket bound to a node’s IPv4 address, the system shall return the address in the

18468 `getsockname()` function as an IPv4-mapped IPv6 address in a **sockaddr_in6** structure.

18469 2.10.20.3 Interface Identification

18470 Each local interface is assigned a unique positive integer as a numeric index. Indexes start at 1;
 18471 zero is not used. There may be gaps so that there is no current interface for a particular positive
 18472 index. Each interface also has a unique implementation-defined name.

18473 2.10.20.4 Options

18474 The following options apply at the IPPROTO_IPV6 level:

18475 IPV6_JOIN_GROUP

18476 When set via `setsockopt()`, it joins the application to a multicast group on an interface
 18477 (identified by its index) and addressed by a given multicast address, enabling packets sent
 18478 to that address to be read via the socket. If the interface index is specified as zero, the
 18479 system selects the interface (for example, by looking up the address in a routing table and
 18480 using the resulting interface).

18481 An attempt to read this option using `getsockopt()` shall result in an [EOPNOTSUPP] error.

18482 The parameter type of this option is a pointer to an **ipv6_mreq** structure.

18483 IPV6_LEAVE_GROUP

18484 When set via `setsockopt()`, it removes the application from the multicast group on an
 18485 interface (identified by its index) and addressed by a given multicast address.

18486 An attempt to read this option using `getsockopt()` shall result in an [EOPNOTSUPP] error.

18487 The parameter type of this option is a pointer to an **ipv6_mreq** structure.

18488 IPV6_MULTICAST_HOPS

18489 The value of this option is the hop limit for outgoing multicast IPv6 packets sent via the
 18490 socket. Its possible values are the same as those of IPV6_UNICAST_HOPS. If the
 18491 IPV6_MULTICAST_HOPS option is not set, a value of 1 is assumed. This option can be set
 18492 via `setsockopt()` and read via `getsockopt()`.

18493 The parameter type of this option is a pointer to an **int**. (Default value: 1)

18494 IPV6_MULTICAST_IF

18495 The index of the interface to be used for outgoing multicast packets. It can be set via
 18496 `setsockopt()` and read via `getsockopt()`. If the interface index is specified as zero, the system
 18497 selects the interface (for example, by looking up the address in a routing table and using the
 18498 resulting interface).

18499 The parameter type of this option is a pointer to an **unsigned int**. (Default value: 0)

18500 IPV6_MULTICAST_LOOP

18501 This option controls whether outgoing multicast packets should be delivered back to the
 18502 local application when the sending interface is itself a member of the destination multicast
 18503 group. If it is set to 1 they are delivered. If it is set to 0 they are not. Other values result in an
 18504 [EINVAL] error. This option can be set via `setsockopt()` and read via `getsockopt()`.

18505 The parameter type of this option is a pointer to an **unsigned int** which is used as a Boolean
 18506 value. (Default value: 1)

18507 IPV6_UNICAST_HOPS

18508 The value of this option is the hop limit for outgoing unicast IPv6 packets sent via the
 18509 socket. If the option is not set, or is set to -1, the system selects a default value. Attempts to

18510 set a value less than -1 or greater than 255 shall result in an [EINVAL] error. This option can
 18511 be set via *setsockopt()* and read via *getsockopt()*.

18512 The parameter type of this option is a pointer to an **int**. (Default value: Unspecified)

18513 IPV6_V6ONLY

18514 This socket option restricts AF_INET6 sockets to IPv6 communications only. AF_INET6
 18515 sockets may be used for both IPv4 and IPv6 communications. Some applications may want
 18516 to restrict their use of an AF_INET6 socket to IPv6 communications only. For these
 18517 applications, the IPV6_V6ONLY socket option is defined. When this option is turned on, the
 18518 socket can be used to send and receive IPv6 packets only. This is an IPPROTO_IPV6-level
 18519 option.

18520 The parameter type of this option is a pointer to an **int** which is used as a Boolean value.
 18521 (Default value: 0)

18522 An [EOPNOTSUPP] error shall result if IPV6_JOIN_GROUP or IPV6_LEAVE_GROUP is used
 18523 with *getsockopt()*.

18524 2.10.20.5 Headers

18525 The symbolic constant AF_INET6 is defined in the `<sys/socket.h>` header to identify the IPv6
 18526 Internet address family. See XBD Chapter 13 (on page 219).

18527 The **sockaddr_storage** structure defined in `<sys/socket.h>` shall be large enough to
 18528 accommodate a **sockaddr_in6** structure (see the `<netinet/in.h>` header defined in XBD Chapter
 18529 13, on page 219) and shall be aligned at an appropriate boundary so that pointers to it can be
 18530 cast as pointers to **sockaddr_in6** structures and used to access the fields of those structures
 18531 without alignment problems. When a **sockaddr_storage** structure is cast as a **sockaddr_in6**
 18532 structure, the *ss_family* field maps onto the *sin6_family* field.

18533 The `<netinet/in.h>`, `<arpa/inet.h>`, and `<netdb.h>` headers contain other definitions used in
 18534 connection with IPv6 Internet sockets; see XBD Chapter 13 (on page 219).

18535 2.11 Tracing

18536 OB TRC This section describes extensions to support tracing of user applications. The functionality
 18537 described in this section is dependent on support of the Trace option (and the rest of this section
 18538 is not further shaded for this option).

18539 The tracing facilities defined in POSIX.1-2008 allow a process to select a set of trace event types,
 18540 to activate a trace stream of the selected trace events as they occur in the flow of execution, and
 18541 to retrieve the recorded trace events.

18542 The tracing operation relies on three logically different components: the traced process, the
 18543 controller process, and the analyzer process. During the execution of the traced process, when a
 18544 trace point is reached, a trace event is recorded into the trace streams created for that process in
 18545 which the associated trace event type identifier is not being filtered out. The controller process
 18546 controls the operation of recording the trace events into the trace stream. It shall be able to:

- 18547 • Initialize the attributes of a trace stream
- 18548 • Create the trace stream (for a specified traced process) using those attributes

- Start and stop tracing for the trace stream
- Filter the type of trace events to be recorded, if the Trace Event Filter option is supported
- Shut a trace stream down

These operations can be done for an active trace stream. The analyzer process retrieves the traced events either at runtime, when the trace stream has not yet been shut down, but is still recording trace events; or after opening a trace log that had been previously recorded and shut down. These three logically different operations can be performed by the same process, or can be distributed into different processes.

A trace stream identifier can be created by a call to *posix_trace_create()*, *posix_trace_create_withlog()*, or *posix_trace_open()*. The *posix_trace_create()* and *posix_trace_create_withlog()* functions should be used by a controller process. The *posix_trace_open()* should be used by an analyzer process.

The tracing functions can serve different purposes. One purpose is debugging the possibly pre-instrumented code, while another is post-mortem fault analysis. These two potential uses differ in that the first requires pre-filtering capabilities to avoid overwhelming the trace stream and permits focusing on expected information; while the second needs comprehensive trace capabilities in order to be able to record all types of information.

The events to be traced belong to two classes:

1. User trace events (generated by the application instrumentation)
2. System trace events (generated by the operating system)

The trace interface defines several system trace event types associated with control of and operation of the trace stream. This small set of system trace events includes the minimum required to interpret correctly the trace event information present in the stream. Other desirable system trace events for some particular application profile may be implemented and are encouraged; for example, process and thread scheduling, signal occurrence, and so on.

Each traced process shall have a mapping of the trace event names to trace event type identifiers that have been defined for that process. Each active trace stream shall have a mapping that incorporates all the trace event type identifiers predefined by the trace system plus all the mappings of trace event names to trace event type identifiers of the processes that are being traced into that trace stream. These mappings are defined from the instrumented application by calling the *posix_trace_eventid_open()* function and from the controller process by calling the *posix_trace_trid_eventid_open()* function. For a pre-recorded trace stream, the list of trace event types is obtained from the pre-recorded trace log.

The last data modification and file status change timestamps of a file associated with an active trace stream shall be marked for update every time any of the tracing operations modifies that file.

The last data access timestamp of a file associated with a trace stream shall be marked for update every time any of the tracing operations causes data to be read from that file.

Results are undefined if the application performs any operation on a file descriptor associated with an active or pre-recorded trace stream until *posix_trace_shutdown()* or *posix_trace_close()* is called for that trace stream. Results are also undefined if the analyzer process and the traced process do not share the same programming environment (see *c99*, Programming Environments in the Shell and Utilities volume of POSIX.1-2008).

The main purpose of this option is to define a complete set of functions and concepts that allow a conforming application to be traced from creation to termination, whatever its realtime constraints and properties.

2.11.1 Tracing Data Definitions

2.11.1.1 Structures

The **<trace.h>** header shall define the *posix_trace_status_info* and *posix_trace_event_info* structures described below. Implementations may add extensions to these structures.

posix_trace_status_info Structure

To facilitate control of a trace stream, information about the current state of an active trace stream can be obtained dynamically. This structure is returned by a call to the *posix_trace_get_status()* function.

The **posix_trace_status_info** structure defined in **<trace.h>** shall contain at least the following members:

Member Type	Member Name	Description
int	<i>posix_stream_status</i>	The operating mode of the trace stream.
int	<i>posix_stream_full_status</i>	The full status of the trace stream.
int	<i>posix_stream_overrun_status</i>	Indicates whether trace events were lost in the trace stream.

If the Trace Log option is supported in addition to the Trace option, the **posix_trace_status_info** structure defined in **<trace.h>** shall contain at least the following additional members:

Member Type	Member Name	Description
int	<i>posix_stream_flush_status</i>	Indicates whether a flush is in progress.
int	<i>posix_stream_flush_error</i>	Indicates whether any error occurred during the last flush operation.
int	<i>posix_log_overrun_status</i>	Indicates whether trace events were lost in the trace log.
int	<i>posix_log_full_status</i>	The full status of the trace log.

The *posix_stream_status* member indicates the operating mode of the trace stream and shall have one of the following values defined by manifest constants in the **<trace.h>** header:

POSIX_TRACE_RUNNING

Tracing is in progress; that is, the trace stream is accepting trace events.

POSIX_TRACE_SUSPENDED

The trace stream is not accepting trace events. The tracing operation has not yet started or has stopped, either following a *posix_trace_stop()* function call or because the trace resources are exhausted.

The *posix_stream_full_status* member indicates the full status of the trace stream, and it shall have one of the following values defined by manifest constants in the **<trace.h>** header:

POSIX_TRACE_FULL

The space in the trace stream for trace events is exhausted.

POSIX_TRACE_NOT_FULL

There is still space available in the trace stream.

The combination of the *posix_stream_status* and *posix_stream_full_status* members also indicates the actual status of the stream. The status shall be interpreted as follows:

18635 POSIX_TRACE_RUNNING and POSIX_TRACE_NOT_FULL
 18636 This status combination indicates that tracing is in progress, and there is space available for
 18637 recording more trace events.

18638 POSIX_TRACE_RUNNING and POSIX_TRACE_FULL
 18639 This status combination indicates that tracing is in progress and that the trace stream is full
 18640 of trace events. This status combination cannot occur unless the *stream-full-policy* is set to
 18641 POSIX_TRACE_LOOP. The trace stream contains trace events recorded during a moving
 18642 time window of prior trace events, and some older trace events may have been overwritten
 18643 and thus lost.

18644 POSIX_TRACE_SUSPENDED and POSIX_TRACE_NOT_FULL
 18645 This status combination indicates that tracing has not yet been started, has been stopped by
 18646 the *posix_trace_stop()* function, or has been cleared by the *posix_trace_clear()* function.

18647 POSIX_TRACE_SUSPENDED and POSIX_TRACE_FULL
 18648 This status combination indicates that tracing has been stopped by the implementation
 18649 because the *stream-full-policy* attribute was POSIX_TRACE_UNTIL_FULL and trace
 18650 resources were exhausted, or that the trace stream was stopped by the function
 18651 *posix_trace_stop()* at a time when trace resources were exhausted.

18652 The *posix_stream_overrun_status* member indicates whether trace events were lost in the trace
 18653 stream, and shall have one of the following values defined by manifest constants in the
 18654 **<trace.h>** header:

18655 POSIX_TRACE_OVERRUN
 18656 At least one trace event was lost and thus was not recorded in the trace stream.

18657 POSIX_TRACE_NO_OVERRUN
 18658 No trace events were lost.

18659 When the corresponding trace stream is created, the *posix_stream_overrun_status* member shall be
 18660 set to POSIX_TRACE_NO_OVERRUN.

18661 Whenever an overrun occurs, the *posix_stream_overrun_status* member shall be set to
 18662 POSIX_TRACE_OVERRUN.

18663 An overrun occurs when:

- 18664 • The policy is POSIX_TRACE_LOOP and a recorded trace event is overwritten.
- 18665 • The policy is POSIX_TRACE_UNTIL_FULL and the trace stream is full when a trace event
 18666 is generated.
- 18667 • If the Trace Log option is supported, the policy is POSIX_TRACE_FLUSH and at least one
 18668 trace event is lost while flushing the trace stream to the trace log.

18669 The *posix_stream_overrun_status* member is reset to zero after its value is read.

18670 If the Trace Log option is supported in addition to the Trace option, the *posix_stream_flush_status*,
 18671 *posix_stream_flush_error*, *posix_log_overrun_status*, and *posix_log_full_status* members are defined
 18672 as follows; otherwise, they are undefined.

18673 The *posix_stream_flush_status* member indicates whether a flush operation is being performed
 18674 and shall have one of the following values defined by manifest constants in the header
 18675 **<trace.h>**:

18676 POSIX_TRACE_FLUSHING
 18677 The trace stream is currently being flushed to the trace log.

18678 POSIX_TRACE_NOT_FLUSHING
 18679 No flush operation is in progress.

18680 The *posix_stream_flush_status* member shall be set to POSIX_TRACE_FLUSHING if a flush
 18681 operation is in progress either due to a call to the *posix_trace_flush()* function (explicit or caused
 18682 by a trace stream shutdown operation) or because the trace stream has become full with the
 18683 *stream-full-policy* attribute set to POSIX_TRACE_FLUSH. The *posix_stream_flush_status* member
 18684 shall be set to POSIX_TRACE_NOT_FLUSHING if no flush operation is in progress.

18685 The *posix_stream_flush_error* member shall be set to zero if no error occurred during flushing. If
 18686 an error occurred during a previous flushing operation, the *posix_stream_flush_error* member
 18687 shall be set to the value of the first error that occurred. If more than one error occurs while
 18688 flushing, error values after the first shall be discarded. The *posix_stream_flush_error* member is
 18689 reset to zero after its value is read.

18690 The *posix_log_outrun_status* member indicates whether trace events were lost in the trace log,
 18691 and shall have one of the following values defined by manifest constants in the **<trace.h>**
 18692 header:

18693 POSIX_TRACE_OVERRUN
 18694 At least one trace event was lost.

18695 POSIX_TRACE_NO_OVERRUN
 18696 No trace events were lost.

18697 When the corresponding trace stream is created, the *posix_log_outrun_status* member shall be set
 18698 to POSIX_TRACE_NO_OVERRUN. Whenever an overrun occurs, this status shall be set to
 18699 POSIX_TRACE_OVERRUN. The *posix_log_outrun_status* member is reset to zero after its value
 18700 is read.

18701 The *posix_log_full_status* member indicates the full status of the trace log, and it shall have one of
 18702 the following values defined by manifest constants in the **<trace.h>** header:

18703 POSIX_TRACE_FULL
 18704 The space in the trace log is exhausted.

18705 POSIX_TRACE_NOT_FULL
 18706 There is still space available in the trace log.

18707 The *posix_log_full_status* member is only meaningful if the *log-full-policy* attribute is either
 18708 POSIX_TRACE_UNTIL_FULL or POSIX_TRACE_LOOP.

18709 For an active trace stream without log, that is created by the *posix_trace_create()* function, the
 18710 *posix_log_outrun_status* member shall be set to POSIX_TRACE_NO_OVERRUN and the
 18711 *posix_log_full_status* member shall be set to POSIX_TRACE_NOT_FULL.

18712 **posix_trace_event_info Structure**

18713 The trace event structure **posix_trace_event_info** contains the information for one recorded
 18714 trace event. This structure is returned by the set of functions *posix_trace_getnext_event()*,
 18715 *posix_trace_timedgetnext_event()*, and *posix_trace_trygetnext_event()*.

18716 The **posix_trace_event_info** structure defined in **<trace.h>** shall contain at least the following
 18717 members:

Member Type	Member Name	Description
trace_event_id_t	<i>posix_event_id</i>	Trace event type identification.
pid_t	<i>posix_pid</i>	Process ID of the process that generated the trace event.
void *	<i>posix_prog_address</i>	Address at which the trace point was invoked.
int	<i>posix_truncation_status</i>	Status about the truncation of the data associated with this trace event.
struct timespec	<i>posix_timestamp</i>	Time at which the trace event was generated.

In addition, the **posix_trace_event_info** structure defined in **<trace.h>** shall contain the following additional member:

Member Type	Member Name	Description
pthread_t	<i>posix_thread_id</i>	Thread ID of the thread that generated the trace event.

The *posix_event_id* member represents the identification of the trace event type and its value is not directly defined by the user. This identification is returned by a call to one of the following functions: *posix_trace_trid_eventid_open()*, *posix_trace_eventtypelist_getnext_id()*, or *posix_trace_eventid_open()*. The name of the trace event type can be obtained by calling *posix_trace_eventid_get_name()*.

The *posix_pid* is the process identifier of the traced process which generated the trace event. If the *posix_event_id* member is one of the implementation-defined system trace events and that trace event is not associated with any process, the *posix_pid* member shall be set to zero.

For a user trace event, the *posix_prog_address* member is the process mapped address of the point at which the associated call to the *posix_trace_event()* function was made. For a system trace event, if the trace event is caused by a system service explicitly called by the application, the *posix_prog_address* member shall be the address of the process at the point where the call to that system service was made.

The *posix_truncation_status* member defines whether the data associated with a trace event has been truncated at the time the trace event was generated, or at the time the trace event was read from the trace stream, or (if the Trace Log option is supported) from the trace log (see the *event* argument from the *posix_trace_getnext_event()* function). The *posix_truncation_status* member shall have one of the following values defined by manifest constants in the **<trace.h>** header:

POSIX_TRACE_NOT_TRUNCATED

All the traced data is available.

POSIX_TRACE_TRUNCATED_RECORD

Data was truncated at the time the trace event was generated.

POSIX_TRACE_TRUNCATED_READ

Data was truncated at the time the trace event was read from a trace stream or a trace log because the reader's buffer was too small. This truncation status overrides the POSIX_TRACE_TRUNCATED_RECORD status.

The *posix_timestamp* member shall be the time at which the trace event was generated. The clock used is implementation-defined, but the resolution of this clock can be retrieved by a call to the *posix_trace_attr_getclockres()* function.

The *posix_thread_id* member is the identifier of the thread that generated the trace event. If the *posix_event_id* member is one of the implementation-defined system trace events and that trace

event is not associated with any thread, the *posix_thread_id* member shall be set to zero.

2.11.1.2 Trace Stream Attributes

Trace streams have attributes that compose the **posix_trace_attr_t** trace stream attributes object. This object shall contain at least the following attributes:

- The *generation-version* attribute identifies the origin and version of the trace system.
- The *trace-name* attribute is a character string defined by the trace controller, and that identifies the trace stream.
- The *creation-time* attribute represents the time of the creation of the trace stream.
- The *clock-resolution* attribute defines the clock resolution of the clock used to generate timestamps.
- The *stream-min-size* attribute defines the minimum size in bytes of the trace stream strictly reserved for the trace events.
- The *stream-full-policy* attribute defines the policy followed when the trace stream is full; its value is `POSIX_TRACE_LOOP`, `POSIX_TRACE_UNTIL_FULL`, or `POSIX_TRACE_FLUSH`.
- The *max-data-size* attribute defines the maximum record size in bytes of a trace event.

In addition, if the Trace option and the Trace Inherit option are both supported, the **posix_trace_attr_t** trace stream creation attributes object shall contain at least the following attributes:

- The *inheritance* attribute specifies whether a newly created trace stream will inherit tracing in its parent's process trace stream. It is either `POSIX_TRACE_INHERITED` or `POSIX_TRACE_CLOSE_FOR_CHILD`.

In addition, if the Trace option and the Trace Log option are both supported, the **posix_trace_attr_t** trace stream creation attributes object shall contain at least the following attribute:

- If the file type corresponding to the trace log supports the `POSIX_TRACE_LOOP` or the `POSIX_TRACE_UNTIL_FULL` policies, the *log-max-size* attribute defines the maximum size in bytes of the trace log associated with an active trace stream. Other stream data—for example, trace attribute values—shall not be included in this size.
- The *log-full-policy* attribute defines the policy of a trace log associated with an active trace stream to be `POSIX_TRACE_LOOP`, `POSIX_TRACE_UNTIL_FULL`, or `POSIX_TRACE_APPEND`.

2.11.2 Trace Event Type Definitions

2.11.2.1 System Trace Event Type Definitions

The following system trace event types, defined in the **<trace.h>** header, track the invocation of the trace operations:

- `POSIX_TRACE_START` shall be associated with a trace start operation.

- POSIX_TRACE_STOP shall be associated with a trace stop operation.
- If the Trace Event Filter option is supported, POSIX_TRACE_FILTER shall be associated with a trace event type filter change operation.

The following system trace event types, defined in the `<trace.h>` header, report operational trace events:

- POSIX_TRACE_OVERFLOW shall mark the beginning of a trace overflow condition.
- POSIX_TRACE_RESUME shall mark the end of a trace overflow condition.
- If the Trace Log option is supported, POSIX_TRACE_FLUSH_START shall mark the beginning of a flush operation.
- If the Trace Log option is supported, POSIX_TRACE_FLUSH_STOP shall mark the end of a flush operation.
- If an implementation-defined trace error condition is reported, it shall be marked POSIX_TRACE_ERROR.

The interpretation of a trace stream or a trace log by a trace analyzer process relies on the information recorded for each trace event, and also on system trace events that indicate the invocation of trace control operations and trace system operational trace events.

The POSIX_TRACE_START and POSIX_TRACE_STOP trace events specify the time windows during which the trace stream is running.

- The POSIX_TRACE_STOP trace event with an associated data that is equal to zero indicates a call of the function `posix_trace_stop()`.
- The POSIX_TRACE_STOP trace event with an associated data that is different from zero indicates an automatic stop of the trace stream (see the definition of the `posix_trace_attr_getstreamfullpolicy()` function in `posix_trace_attr_getinherited()`).

The POSIX_TRACE_FILTER trace event indicates that a trace event type filter value changed while the trace stream was running.

The POSIX_TRACE_ERROR serves to inform the analyzer process that an implementation-defined internal error of the trace system occurred.

The POSIX_TRACE_OVERFLOW trace event shall be reported with a timestamp equal to the timestamp of the first trace event overwritten. This is an indication that some generated trace events have been lost.

The POSIX_TRACE_RESUME trace event shall be reported with a timestamp equal to the timestamp of the first valid trace event reported after the overflow condition ends and shall be reported before this first valid trace event. This is an indication that the trace system is reliably recording trace events after an overflow condition.

Each of these trace event types shall be defined by a constant trace event name and a **trace_event_id_t** constant; trace event data is associated with some of these trace events.

If the Trace option is supported and the Trace Event Filter option and the Trace Log option are not supported, the following predefined system trace events in [Table 2-3](#) (on page 542) shall be defined:

18839

Table 2-3 Trace Option: System Trace Events

18840

18841

18842

18843

18844

18845

18846

18847

18848

Event Name	Constant	Associated Data
		Data Type
posix_trace_error	POSIX_TRACE_ERROR	error int
posix_trace_start	POSIX_TRACE_START	None.
posix_trace_stop	POSIX_TRACE_STOP	auto int
posix_trace_overflow	POSIX_TRACE_OVERFLOW	None.
posix_trace_resume	POSIX_TRACE_RESUME	None.

18849

18850

18851

If the Trace option and the Trace Event Filter option are both supported, and if the Trace Log option is not supported, the following predefined system trace events in [Table 2-4](#) shall be defined:

18852

Table 2-4 Trace and Trace Event Filter Options: System Trace Events

18853

18854

18855

18856

18857

18858

18859

18860

18861

18862

18863

18864

18865

Event Name	Constant	Associated Data
		Data Type
posix_trace_error	POSIX_TRACE_ERROR	error int
posix_trace_start	POSIX_TRACE_START	event_filter trace_event_set_t
posix_trace_stop	POSIX_TRACE_STOP	auto int
posix_trace_filter	POSIX_TRACE_FILTER	old_event_filter new_event_filter trace_event_set_t
posix_trace_overflow	POSIX_TRACE_OVERFLOW	None.
posix_trace_resume	POSIX_TRACE_RESUME	None.

18866

18867

18868

If the Trace option and the Trace Log option are both supported, and if the Trace Event Filter option is not supported, the following predefined system trace events in [Table 2-5](#) (on page 543) shall be defined:

Table 2-5 Trace and Trace Log Options: System Trace Events

Event Name	Constant	Associated Data
		Data Type
posix_trace_error	POSIX_TRACE_ERROR	error int
posix_trace_start	POSIX_TRACE_START	None.
posix_trace_stop	POSIX_TRACE_STOP	auto int
posix_trace_overflow	POSIX_TRACE_OVERFLOW	None.
posix_trace_resume	POSIX_TRACE_RESUME	None.
posix_trace_flush_start	POSIX_TRACE_FLUSH_START	None.
posix_trace_flush_stop	POSIX_TRACE_FLUSH_STOP	None.

If the Trace option, the Trace Event Filter option, and the Trace Log option are all supported, the following predefined system trace events in [Table 2-6](#) shall be defined:

Table 2-6 Trace, Trace Log, and Trace Event Filter Options: System Trace Events

Event Name	Constant	Associated Data
		Data Type
posix_trace_error	POSIX_TRACE_ERROR	error int
posix_trace_start	POSIX_TRACE_START	event_filter trace_event_set_t
posix_trace_stop	POSIX_TRACE_STOP	auto int
posix_trace_filter	POSIX_TRACE_FILTER	old_event_filter new_event_filter trace_event_set_t
posix_trace_overflow	POSIX_TRACE_OVERFLOW	None.
posix_trace_resume	POSIX_TRACE_RESUME	None.
posix_trace_flush_start	POSIX_TRACE_FLUSH_START	None.
posix_trace_flush_stop	POSIX_TRACE_FLUSH_STOP	None.

2.11.2.2 User Trace Event Type Definitions

The user trace event `POSIX_TRACE_UNNAMED_USEREVENT` is defined in the `<trace.h>` header. If the limit of per-process user trace event names represented by `{TRACE_USER_EVENT_MAX}` has already been reached, this predefined user event shall be returned when the application tries to register more events than allowed. The data associated with this trace event is application-defined.

The following predefined user trace event in [Table 2-7](#) (on page 544) shall be defined:

Table 2-7 Trace Option: User Trace Event

Event Name	Constant
posix_trace_unnamed_userevent	POSIX_TRACE_UNNAMED_USEREVENT

2.11.3 Trace Functions

The trace interface is built and structured to improve portability through use of trace data of opaque type. The object-oriented approach for the manipulation of trace attributes and trace event type identifiers requires definition of many constructor and selector functions which operate on these opaque types. Also, the trace interface must support several different tracing roles. To facilitate reading the trace interface, the trace functions are grouped into small functional sets supporting the three different roles:

- A trace controller process requires functions to set up and customize all the resources needed to run a trace stream, including:
 - Attribute initialization and destruction (*posix_trace_attr_init()*)
 - Identification information manipulation (*posix_trace_attr_getgenversion()*)
 - Trace system behavior modification (*posix_trace_attr_getinherited()*)
 - Trace stream and trace log size set (*posix_trace_attr_getmaxusereventsize()*)
 - Trace stream creation, flush, and shutdown (*posix_trace_create()*)
 - Trace stream and trace log clear (*posix_trace_clear()*)
 - Trace event type identifier manipulation (*posix_trace_trid_eventid_open()*)
 - Trace event type identifier list exploration (*posix_trace_eventtypelist_getnext_id()*)
 - Trace event type set manipulation (*posix_trace_eventset_empty()*)
 - Trace event type filter set (*posix_trace_set_filter()*)
 - Trace stream start and stop (*posix_trace_start()*)
 - Trace stream information and status read (*posix_trace_get_attr()*)
- A traced process requires functions to instrument trace points:
 - Trace event type identifiers definition and trace points insertion (*posix_trace_event()*)
- A trace analyzer process requires functions to retrieve information from a trace stream and trace log:
 - Identification information read (*posix_trace_attr_getgenversion()*)
 - Trace system behavior information read (*posix_trace_attr_getinherited()*)
 - Trace stream and trace log size get (*posix_trace_attr_getmaxusereventsize()*)
 - Trace event type identifier manipulation (*posix_trace_trid_eventid_open()*)
 - Trace event type identifier list exploration (*posix_trace_eventtypelist_getnext_id()*)
 - Trace log open, rewind, and close (*posix_trace_open()*)

- 18940 — Trace stream information and status read (*posix_trace_get_attr()*)
- 18941 — Trace event read (*posix_trace_getnext_event()*)

18942 2.12 Data Types

18943 2.12.1 Defined Types

18944 All of the data types used by various functions are defined by the implementation. The
 18945 following table describes some of these types. Other types referenced in the description of a
 18946 function, not mentioned here, can be found in the appropriate header for that function.

Defined Type	Description
cc_t	Type used for terminal special characters.
clock_t	Integer or real-floating type used for processor times, as defined in the ISO C standard.
clockid_t	Used for clock ID type in some timer functions.
dev_t	Integer type used for device numbers.
DIR	Type representing a directory stream.
div_t	Structure type returned by the <i>div()</i> function.
FILE	Structure containing information about a file.
glob_t	Structure type used in pathname pattern matching.
fpos_t	Type containing all information needed to specify uniquely every position within a file.
gid_t	Integer type used for group IDs.
iconv_t	Type used for conversion descriptors.
id_t	Integer type used as a general identifier; can be used to contain at least the largest of a pid_t , uid_t , or gid_t .
ino_t	Unsigned integer type used for file serial numbers.
key_t	Arithmetic type used for XSI interprocess communication.
ldiv_t	Structure type returned by the <i>ldiv()</i> function.
mode_t	Integer type used for file attributes.
mqd_t	Used for message queue descriptors.
nfds_t	Integer type used for the number of file descriptors.
nlink_t	Integer type used for link counts.
off_t	Signed integer type used for file sizes.
pid_t	Signed integer type used for process and process group IDs.
pthread_attr_t	Used to identify a thread attribute object.
pthread_cond_t	Used for condition variables.
pthread_condattr_t	Used to identify a condition attribute object.
pthread_key_t	Used for thread-specific data keys.
pthread_mutex_t	Used for mutexes.
pthread_mutexattr_t	Used to identify a mutex attribute object.
pthread_once_t	Used for dynamic package initialization.
pthread_rwlock_t	Used for read-write locks.
pthread_rwlockattr_t	Used for read-write lock attributes.
pthread_t	Used to identify a thread.

	Defined Type	Description
18982		
18983	ptrdiff_t	Signed integer type of the result of subtracting two pointers.
18984	regex_t	Structure type used in regular expression matching.
18985	regmatch_t	Structure type used in regular expression matching.
18986	rlim_t	Unsigned integer type used for limit values, to which objects of type int and off_t can be cast without loss of value.
18987		
18988	sem_t	Type used in performing semaphore operations.
18989	sig_atomic_t	Integer type of an object that can be accessed as an atomic entity, even in the presence of asynchronous interrupts.
18990		
18991	sigset_t	Integer or structure type of an object used to represent sets of signals.
18992		
18993	size_t	Unsigned integer type used for size of objects.
18994	speed_t	Type used for terminal baud rates.
18995	ssize_t	Signed integer type used for a count of bytes or an error indication.
18996		
18997	suseconds_t	Signed integer type used for time in microseconds.
18998	tcflag_t	Type used for terminal modes.
18999	time_t	Integer type used for time in seconds, as defined in the ISO C standard.
19000		
19001	timer_t	Used for timer ID returned by the <i>timer_create()</i> function.
19002	uid_t	Integer type used for user IDs.
19003	va_list	Type used for traversing variable argument lists.
19004	wchar_t	Integer type whose range of values can represent distinct codes for all members of the largest extended character set specified by the supported locales.
19005		
19006		
19007	wctype_t	Scalar type which represents a character class descriptor.
19008	wint_t	Integer type capable of storing any valid value of wchar_t or WEOF.
19009		
19010	wordexp_t	Structure type used in word expansion.

19011 2.12.2 The char Type

19012 The type **char** is defined as a single byte; see XBD [Chapter 3](#) (on page 33) (Byte and Character). -

19013

Chapter 3

19014



System Interfaces

19015

This chapter describes the functions, macros, and external variables to support applications portability at the C-language source level.

19016

19017 **NAME**

19018 FD_CLR — macros for synchronous I/O multiplexing

19019 **SYNOPSIS**

```
19020       #include <sys/select.h>
19021       void FD_CLR(int fd, fd_set *fdset);
19022       int FD_ISSET(int fd, fd_set *fdset);
19023       void FD_SET(int fd, fd_set *fdset);
19024       void FD_ZERO(fd_set *fdset);
```

19025 **DESCRIPTION**

19026 Refer to *pselect()*.

19027 **NAME**

19028 _exit, _exit — terminate a process

19029 **SYNOPSIS**

19030 #include <stdlib.h>

19031 void _Exit(int *status*);

19032 #include <unistd.h>

19033 void _exit(int *status*);19034 **DESCRIPTION**

19035 CX For `_Exit()`: The functionality described on this reference page is aligned with the ISO C
 19036 standard. Any conflict between the requirements described here and the ISO C standard is
 19037 unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

19038 CX The value of *status* may be 0, EXIT_SUCCESS, EXIT_FAILURE, or any other value, though only
 19039 the least significant 8 bits (that is, *status* & 0377) shall be available to a waiting parent process.

19040 CX The `_Exit()` and `_exit()` functions shall be functionally equivalent.

19041 CX The `_Exit()` and `_exit()` functions shall not call functions registered with `atexit()` nor any
 19042 CX registered signal handlers. Open streams shall not be flushed. Whether open streams are
 19043 closed (without flushing) is implementation-defined. Finally, the calling process shall be
 19044 terminated with the consequences described below.

19045 **Consequences of Process Termination**

19046 CX Process termination caused by any reason shall have the following consequences:

19047 **Note:** These consequences are all extensions to the ISO C standard and are not further CX shaded.
 19048 However, functionality relating to the XSI option is shaded.

19049 • All of the file descriptors, directory streams, conversion descriptors, and message catalog
 19050 descriptors open in the calling process shall be closed.

19051 XSI • If the parent process of the calling process is executing a `wait()`, `waitid()`, or `waitpid()`, and
 19052 has neither set its SA_NOCLDWAIT flag nor set SIGCHLD to SIG_IGN, it shall be notified
 19053 of termination of the calling process and the low-order eight bits (that is, bits 0377) of *status*
 19054 shall be made available to it. If the parent is not waiting, the child's status shall be made
 19055 available to it when the parent subsequently executes `wait()`, `waitid()`, or `waitpid()`.

19056 The semantics of the `waitid()` function shall be equivalent to `wait()`.

19057 XSI • If the parent process of the calling process is not executing a `wait()`, `waitid()`, or `waitpid()`,
 19058 and has neither set its SA_NOCLDWAIT flag nor set SIGCHLD to SIG_IGN, the calling
 19059 process shall be transformed into a *zombie process*. A *zombie process* is an inactive process
 19060 and it shall be deleted at some later time when its parent process executes `wait()`, `waitid()`,
 19061 or `waitpid()`.

19062 XSI The semantics of the `waitid()` function shall be equivalent to `wait()`.

19063 • Termination of a process does not directly terminate its children. The sending of a SIGHUP
 19064 signal as described below indirectly terminates children in some circumstances.

19065 • Either:

19066 If the implementation supports the SIGCHLD signal, a SIGCHLD shall be sent to the
 19067 parent process.

19068 Or:

- 19069 XSI If the parent process has set its SA_NOCLDWAIT flag, or set SIGCHLD to SIG_IGN, the
 19070 status shall be discarded, and the lifetime of the calling process shall end immediately. If
 19071 SA_NOCLDWAIT is set, it is implementation-defined whether a SIGCHLD signal is sent to
 19072 the parent process.
- 19073 • The parent process ID of all of the existing child processes and zombie processes of the
 19074 calling process shall be set to the process ID of an implementation-defined system process.
 19075 That is, these processes shall be inherited by a special system process.
- 19076 XSI • Each attached shared-memory segment is detached and the value of *shm_nattch* (see
 19077 *shmget()*) in the data structure associated with its shared memory ID shall be decremented
 19078 by 1.
- 19079 XSI • For each semaphore for which the calling process has set a *semadj* value (see *semop()*), that
 19080 value shall be added to the *semval* of the specified semaphore.
- 19081 • If the process is a controlling process, the SIGHUP signal shall be sent to each process in
 19082 the foreground process group of the controlling terminal belonging to the calling process.
- 19083 • If the process is a controlling process, the controlling terminal associated with the session
 19084 shall be disassociated from the session, allowing it to be acquired by a new controlling
 19085 process.
- 19086 • If the exit of the process causes a process group to become orphaned, and if any member of
 19087 the newly-orphaned process group is stopped, then a SIGHUP signal followed by a
 19088 SIGCONT signal shall be sent to each process in the newly-orphaned process group.
- 19089 • All open named semaphores in the calling process shall be closed as if by appropriate calls
 19090 to *sem_close()*.
- 19091 ML • Any memory locks established by the process via calls to *mlockall()* or *mlock()* shall be
 19092 removed. If locked pages in the address space of the calling process are also mapped into
 19093 the address spaces of other processes and are locked by those processes, the locks
 19094 established by the other processes shall be unaffected by the call by this process to *_Exit()*
 19095 or *_exit()*.
- 19096 • Memory mappings that were created in the process shall be unmapped before the process
 19097 is destroyed.
- 19098 TYM • Any blocks of typed memory that were mapped in the calling process shall be unmapped,
 19099 as if *munmap()* was implicitly called to unmap them.
- 19100 MSG • All open message queue descriptors in the calling process shall be closed as if by
 19101 appropriate calls to *mq_close()*.
- 19102 • Any outstanding cancelable asynchronous I/O operations may be canceled. Those
 19103 asynchronous I/O operations that are not canceled shall complete as if the *_Exit()* or
 19104 *_exit()* operation had not yet occurred, but any associated signal notifications shall be
 19105 suppressed. The *_Exit()* or *_exit()* operation may block awaiting such I/O completion.
 19106 Whether any I/O is canceled, and which I/O may be canceled upon *_Exit()* or *_exit()*, is
 19107 implementation-defined.
- 19108 • Threads terminated by a call to *_Exit()* or *_exit()* shall not invoke their cancellation
 19109 cleanup handlers or per-thread data destructors.
- 19110 OB TRC • If the calling process is a trace controller process, any trace streams that were created by
 19111 the calling process shall be shut down as described by the *posix_trace_shutdown()* function,
 19112 and mapping of trace event names to trace event type identifiers of any process built for
 19113 these trace streams may be deallocated.

19114 RETURN VALUE

19115 These functions do not return.

19116 ERRORS

19117 No errors are defined.

19118 EXAMPLES

19119 None.

19120 APPLICATION USAGE

19121 Normally applications should use *exit()* rather than *_Exit()* or *_exit()*.

19122 RATIONALE**19123 Process Termination**

19124 Early proposals drew a distinction between normal and abnormal process termination.
19125 Abnormal termination was caused only by certain signals and resulted in implementation-
19126 defined “actions”, as discussed below. Subsequent proposals distinguished three types of
19127 termination: *normal termination* (as in the current specification), *simple abnormal termination*, and
19128 *abnormal termination with actions*. Again the distinction between the two types of abnormal
19129 termination was that they were caused by different signals and that implementation-defined
19130 actions would result in the latter case. Given that these actions were completely implementation-
19131 defined, the early proposals were only saying when the actions could occur and how their
19132 occurrence could be detected, but not what they were. This was of little or no use to conforming
19133 applications, and thus the distinction is not made in this volume of POSIX.1-2008.

19134 The implementation-defined actions usually include, in most historical implementations, the
19135 creation of a file named **core** in the current working directory of the process. This file contains an
19136 image of the memory of the process, together with descriptive information about the process,
19137 perhaps sufficient to reconstruct the state of the process at the receipt of the signal.

19138 There is a potential security problem in creating a **core** file if the process was set-user-ID and the
19139 current user is not the owner of the program, if the process was set-group-ID and none of the
19140 user’s groups match the group of the program, or if the user does not have permission to write
19141 in the current directory. In this situation, an implementation either should not create a **core** file
19142 or should make it unreadable by the user.

19143 Despite the silence of this volume of POSIX.1-2008 on this feature, applications are advised not
19144 to create files named **core** because of potential conflicts in many implementations. Some
19145 implementations use a name other than **core** for the file; for example, by appending the process
19146 ID to the filename.

19147 Terminating a Process

19148 It is important that the consequences of process termination as described occur regardless of
19149 whether the process called *_exit()* (perhaps indirectly through *exit()*) or instead was terminated
19150 due to a signal or for some other reason. Note that in the specific case of *exit()* this means that
19151 the *status* argument to *exit()* is treated in the same way as the *status* argument to *_exit()*.

19152 A language other than C may have other termination primitives than the C-language *exit()*
19153 function, and programs written in such a language should use its native termination primitives,
19154 but those should have as part of their function the behavior of *_exit()* as described.
19155 Implementations in languages other than C are outside the scope of this version of this volume
19156 of POSIX.1-2008, however.

19157 As required by the ISO C standard, using **return** from *main()* has the same behavior (other than

with respect to language scope issues) as calling *exit()* with the returned value. Reaching the end of the *main()* function has the same behavior as calling *exit(0)*.

A value of zero (or `EXIT_SUCCESS`, which is required to be zero) for the argument *status* conventionally indicates successful termination. This corresponds to the specification for *exit()* in the ISO C standard. The convention is followed by utilities such as *make* and various shells, which interpret a zero status from a child process as success. For this reason, applications should not call *exit(0)* or *_exit(0)* when they terminate unsuccessfully; for example, in signal-catching functions.

Historically, the implementation-defined process that inherits children whose parents have terminated without waiting on them is called *init* and has a process ID of 1.

The sending of a `SIGHUP` to the foreground process group when a controlling process terminates corresponds to somewhat different historical implementations. In System V, the kernel sends a `SIGHUP` on termination of (essentially) a controlling process. In 4.2 BSD, the kernel does not send `SIGHUP` in a case like this, but the termination of a controlling process is usually noticed by a system daemon, which arranges to send a `SIGHUP` to the foreground process group with the *vhangup()* function. However, in 4.2 BSD, due to the behavior of the shells that support job control, the controlling process is usually a shell with no other processes in its process group. Thus, a change to make *_exit()* behave this way in such systems should not cause problems with existing applications.

The termination of a process may cause a process group to become orphaned in either of two ways. The connection of a process group to its parent(s) outside of the group depends on both the parents and their children. Thus, a process group may be orphaned by the termination of the last connecting parent process outside of the group or by the termination of the last direct descendant of the parent process(es). In either case, if the termination of a process causes a process group to become orphaned, processes within the group are disconnected from their job control shell, which no longer has any information on the existence of the process group. Stopped processes within the group would languish forever. In order to avoid this problem, newly orphaned process groups that contain stopped processes are sent a `SIGHUP` signal and a `SIGCONT` signal to indicate that they have been disconnected from their session. The `SIGHUP` signal causes the process group members to terminate unless they are catching or ignoring `SIGHUP`. Under most circumstances, all of the members of the process group are stopped if any of them are stopped.

The action of sending a `SIGHUP` and a `SIGCONT` signal to members of a newly orphaned process group is similar to the action of 4.2 BSD, which sends `SIGHUP` and `SIGCONT` to each stopped child of an exiting process. If such children exit in response to the `SIGHUP`, any additional descendants receive similar treatment at that time. In this volume of POSIX.1-2008, the signals are sent to the entire process group at the same time. Also, in this volume of POSIX.1-2008, but not in 4.2 BSD, stopped processes may be orphaned, but may be members of a process group that is not orphaned; therefore, the action taken at *_exit()* must consider processes other than child processes.

It is possible for a process group to be orphaned by a call to *setpgid()* or *setsid()*, as well as by process termination. This volume of POSIX.1-2008 does not require sending `SIGHUP` and `SIGCONT` in those cases, because, unlike process termination, those cases are not caused accidentally by applications that are unaware of job control. An implementation can choose to send `SIGHUP` and `SIGCONT` in those cases as an extension; such an extension must be documented as required in **<signal.h>**.

The ISO/IEC 9899:1999 standard adds the *_Exit()* function that results in immediate program termination without triggering signals or *atexit()*-registered functions. In POSIX.1-2008, this is

19206 equivalent to the `_exit()` function.

19207 FUTURE DIRECTIONS

19208 None.

19209 SEE ALSO

19210 `atexit()`, `exit()`, `mlock()`, `mlockall()`, `mq_close()`, `munmap()`, `posix_trace_create()`, `sem_close()`,
19211 `semop()`, `setpgid()`, `setsid()`, `shmget()`, `wait()`, `waitid()`

19212 XBD `<stdlib.h>`, `<unistd.h>`

19213 CHANGE HISTORY

19214 First released in Issue 1. Derived from Issue 1 of the SVID.

19215 Issue 5

19216 The DESCRIPTION is updated for alignment with the POSIX Realtime Extension and the POSIX
19217 Threads Extension.

19218 Interactions with the SA_NOCLDWAIT flag and SIGCHLD signal are further clarified.

19219 The values of *status* from `exit()` are better described.

19220 Issue 6

19221 Extensions beyond the ISO C standard are marked.

19222 The DESCRIPTION is updated for alignment with IEEE Std 1003.1j-2000 by adding semantics
19223 for typed memory.

19224 The following changes are made for alignment with the ISO/IEC 9899:1999 standard:

- 19225 • The `_Exit()` function is included.
- 19226 • The DESCRIPTION is updated.

19227 The description of tracing semantics is added for alignment with IEEE Std 1003.1q-2000.

19228 References to the `wait3()` function are removed.

19229 IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/16 is applied, correcting grammar in the
19230 DESCRIPTION.

19231 Issue 7

19232 Austin Group Interpretation 1003.1-2001 #031 is applied, separating these functions from the
19233 `exit()` function.

19234 Austin Group Interpretation 1003.1-2001 #085 is applied, clarifying the text regarding flushing of
19235 streams and closing of temporary files.

19236 Functionality relating to the Asynchronous Input and Output, Memory Mapped Files, and
19237 Semaphores options is moved to the Base.

19238 NAME

19239 _longjmp, _setjmp — non-local goto

19240 SYNOPSIS

```
19241 OB XSI  #include <setjmp.h>
19242         void _longjmp(jmp_buf env, int val);
19243         int _setjmp(jmp_buf env);
```

19244 DESCRIPTION

19245 The *_longjmp()* and *_setjmp()* functions shall be equivalent to *longjmp()* and *setjmp()*,
19246 respectively, with the additional restriction that *_longjmp()* and *_setjmp()* shall not manipulate
19247 the signal mask.

19248 If *_longjmp()* is called even though *env* was never initialized by a call to *_setjmp()*, or when the
19249 last such call was in a function that has since returned, the results are undefined.

19250 RETURN VALUE

19251 Refer to *longjmp()* and *setjmp()*.

19252 ERRORS

19253 No errors are defined.

19254 EXAMPLES

19255 None.

19256 APPLICATION USAGE

19257 If *_longjmp()* is executed and the environment in which *_setjmp()* was executed no longer exists,
19258 errors can occur. The conditions under which the environment of the *_setjmp()* no longer exists
19259 include exiting the function that contains the *_setjmp()* call, and exiting an inner block with
19260 temporary storage. This condition might not be detectable, in which case the *_longjmp()* occurs
19261 and, if the environment no longer exists, the contents of the temporary storage of an inner block
19262 are unpredictable. This condition might also cause unexpected process termination. If the
19263 function has returned, the results are undefined.

19264 Passing *longjmp()* a pointer to a buffer not created by *setjmp()*, passing *_longjmp()* a pointer to a
19265 buffer not created by *_setjmp()*, passing *siglongjmp()* a pointer to a buffer not created by
19266 *sigsetjmp()*, or passing any of these three functions a buffer that has been modified by the user
19267 can cause all the problems listed above, and more.

19268 The *_longjmp()* and *_setjmp()* functions are included to support programs written to historical
19269 system interfaces. New applications should use *siglongjmp()* and *sigsetjmp()* respectively.

19270 RATIONALE

19271 None.

19272 FUTURE DIRECTIONS

19273 The *_longjmp()* and *_setjmp()* functions may be removed in a future version.

19274 SEE ALSO

19275 *longjmp()*, *setjmp()*, *siglongjmp()*, *sigsetjmp()*

19276 XBD *<setjmp.h>*

19277 CHANGE HISTORY

19278 First released in Issue 4, Version 2.

19279 **Issue 5**

19280 Moved from X/OPEN UNIX extension to BASE.

19281 **Issue 7**19282 The *_longjmp()* and *_setjmp()* functions are marked obsolescent.

19283 NAME

19284 _toupper — transliterate uppercase characters to lowercase

19285 SYNOPSIS

```
19286 OB XSI  #include <ctype.h>
19287         int _toupper(int c);
```

19288 DESCRIPTION

19289 The *_toupper()* macro shall be equivalent to *toupper(c)* except that the application shall ensure
19290 that the argument *c* is an uppercase letter.

19291 RETURN VALUE

19292 Upon successful completion, *_toupper()* shall return the lowercase letter corresponding to the
19293 argument passed.

19294 ERRORS

19295 No errors are defined.

19296 EXAMPLES

19297 None.

19298 APPLICATION USAGE

19299 Applications should use the *toupper()* function instead of the obsolescent *_toupper()* function.

19300 RATIONALE

19301 None.

19302 FUTURE DIRECTIONS

19303 The *_toupper()* function may be removed in a future version.

19304 SEE ALSO

19305 *toupper()*, *isupper()*

19306 XBD [Chapter 7](#) (on page 135), [<ctype.h>](#)

19307 CHANGE HISTORY

19308 First released in Issue 1. Derived from Issue 1 of the SVID.

19309 Issue 6

19310 The normative text is updated to avoid use of the term “must” for application requirements.

19311 Issue 7

19312 The *_toupper()* function is marked obsolescent.

19313 NAME

19314 _toupper — transliterate lowercase characters to uppercase

19315 SYNOPSIS

```
19316 OB XSI  #include <ctype.h>
19317         int _toupper(int c);
```

19318 DESCRIPTION

19319 The *_toupper()* macro shall be equivalent to *toupper()* except that the application shall ensure
19320 that the argument *c* is a lowercase letter.

19321 RETURN VALUE

19322 Upon successful completion, *_toupper()* shall return the uppercase letter corresponding to the
19323 argument passed.

19324 ERRORS

19325 No errors are defined.

19326 EXAMPLES

19327 None.

19328 APPLICATION USAGE

19329 Applications should use the *toupper()* function instead of the obsolescent *_toupper()* function.

19330 RATIONALE

19331 None.

19332 FUTURE DIRECTIONS

19333 The *_toupper()* function may be removed in a future version.

19334 SEE ALSO

19335 *islower()*, *toupper()*

19336 XBD [Chapter 7](#) (on page 135), [<ctype.h>](#)

19337 CHANGE HISTORY

19338 First released in Issue 1. Derived from Issue 1 of the SVID.

19339 Issue 6

19340 The normative text is updated to avoid use of the term “must” for application requirements.

19341 Issue 7

19342 The *_toupper()* function is marked obsolescent.

19343 **NAME**

19344 a64l, l64a — convert between a 32-bit integer and a radix-64 ASCII string

19345 **SYNOPSIS**

```
19346 XSI      #include <stdlib.h>
19347          long a64l(const char *s);
19348          char *l64a(long value);
```

19349 **DESCRIPTION**

19350 These functions maintain numbers stored in radix-64 ASCII characters. This is a notation by
 19351 which 32-bit integers can be represented by up to six characters; each character represents a digit
 19352 in radix-64 notation. If the type **long** contains more than 32 bits, only the low-order 32 bits shall
 19353 be used for these operations.

19354 The characters used to represent digits are ' .' (dot) for 0, ' / ' for 1, ' 0 ' through ' 9 ' for [2,11],
 19355 ' A ' through ' Z ' for [12,37], and ' a ' through ' z ' for [38,63].

19356 The *a64l()* function shall take a pointer to a radix-64 representation, in which the first digit is the
 19357 least significant, and return the corresponding **long** value. If the string pointed to by *s* contains
 19358 more than six characters, *a64l()* shall use the first six. If the first six characters of the string
 19359 contain a null terminator, *a64l()* shall use only characters preceding the null terminator. The
 19360 *a64l()* function shall scan the character string from left to right with the least significant digit on
 19361 the left, decoding each character as a 6-bit radix-64 number. If the type **long** contains more than
 19362 32 bits, the resulting value is sign-extended. The behavior of *a64l()* is unspecified if *s* is a null
 19363 pointer or the string pointed to by *s* was not generated by a previous call to *l64a()*.

19364 The *l64a()* function shall take a **long** argument and return a pointer to the corresponding
 19365 radix-64 representation. The behavior of *l64a()* is unspecified if *value* is negative.

19366 The value returned by *l64a()* may be a pointer into a static buffer. Subsequent calls to *l64a()* may
 19367 overwrite the buffer.

19368 The *l64a()* function need not be thread-safe.

19369 **RETURN VALUE**

19370 Upon successful completion, *a64l()* shall return the **long** value resulting from conversion of the
 19371 input string. If a string pointed to by *s* is an empty string, *a64l()* shall return 0L.

19372 The *l64a()* function shall return a pointer to the radix-64 representation. If *value* is 0L, *l64a()* shall
 19373 return a pointer to an empty string.

19374 **ERRORS**

19375 No errors are defined.

19376 **EXAMPLES**

19377 None.

19378 **APPLICATION USAGE**

19379 If the type **long** contains more than 32 bits, the result of *a64l(l64a(x))* is *x* in the low-order 32 bits.

19380 **RATIONALE**

19381 This is not the same encoding as used by either encoding variant of the *uuencode* utility.

19382 **FUTURE DIRECTIONS**

19383 None.

19384 **SEE ALSO**19385 [strtoul\(\)](#)19386 XBD [<stdlib.h>](#)19387 XCU [uuencode](#)19388 **CHANGE HISTORY**

19389 First released in Issue 4, Version 2.

19390 **Issue 5**

19391 Moved from X/OPEN UNIX extension to BASE.

19392 Normative text previously in the APPLICATION USAGE section is moved to the
19393 DESCRIPTION.19394 A note indicating that the *l64a()* function need not be reentrant is added to the DESCRIPTION.19395 **Issue 7**

19396 Austin Group Interpretation 1003.1-2001 #156 is applied.

19397 **NAME**

19398 abort — generate an abnormal process abort

19399 **SYNOPSIS**

19400 #include <stdlib.h>

19401 void abort(void);

19402 **DESCRIPTION**

19403 CX The functionality described on this reference page is aligned with the ISO C standard. Any
 19404 conflict between the requirements described here and the ISO C standard is unintentional. This
 19405 volume of POSIX.1-2008 defers to the ISO C standard.

19406 The *abort()* function shall cause abnormal process termination to occur, unless the signal
 19407 SIGABRT is being caught and the signal handler does not return.

19408 CX The abnormal termination processing shall include the default actions defined for SIGABRT and
 19409 may include an attempt to effect *fclose()* on all open streams.

19410 The SIGABRT signal shall be sent to the calling process as if by means of *raise()* with the
 19411 argument SIGABRT.

19412 CX The status made available to *wait()*, *waitid()*, or *waitpid()* by *abort()* shall be that of a process
 19413 terminated by the SIGABRT signal. The *abort()* function shall override blocking or ignoring the
 19414 SIGABRT signal.

19415 **RETURN VALUE**19416 The *abort()* function shall not return.19417 **ERRORS**

19418 No errors are defined.

19419 **EXAMPLES**

19420 None.

19421 **APPLICATION USAGE**

19422 Catching the signal is intended to provide the application developer with a portable means to
 19423 abort processing, free from possible interference from any implementation-supplied functions.

19424 **RATIONALE**

19425 The ISO/IEC 9899:1999 standard requires the *abort()* function to be async-signal-safe. Since
 19426 POSIX.1-2008 defers to the ISO C standard, this required a change to the DESCRIPTION from
 19427 “shall include the effect of *fclose()*” to “may include an attempt to effect *fclose()*.”

19428 The revised wording permits some backwards-compatibility and avoids a potential deadlock
 19429 situation.

19430 The Open Group Base Resolution bwg2002-003 is applied, removing the following XSI shaded
 19431 paragraph from the DESCRIPTION:

19432 “On XSI-conformant systems, in addition the abnormal termination processing shall include the
 19433 effect of *fclose()* on message catalog descriptors.”

19434 There were several reasons to remove this paragraph:

- 19435 • No special processing of open message catalogs needs to be performed prior to abnormal
 19436 process termination.
- 19437 • The main reason to specifically mention that *abort()* includes the effect of *fclose()* on open
 19438 streams is to flush output queued on the stream. Message catalogs in this context are read-
 19439 only and, therefore, do not need to be flushed.

- The effect of *fclose()* on a message catalog descriptor is unspecified. Message catalog descriptors are allowed, but not required to be implemented using a file descriptor, but there is no mention in POSIX.1-2008 of a message catalog descriptor using a standard I/O stream FILE object as would be expected by *fclose()*.

FUTURE DIRECTIONS

None.

SEE ALSO

exit(), *kill()*, *raise()*, *signal()*, *wait()*, *waitid()*

XBD <stdlib.h>

CHANGE HISTORY

First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6

Extensions beyond the ISO C standard are marked.

Changes are made to the DESCRIPTION for alignment with the ISO/IEC 9899:1999 standard.

The Open Group Base Resolution bwg2002-003 is applied.

IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/10 is applied, changing the DESCRIPTION of abnormal termination processing and adding to the RATIONALE section.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/9 is applied, changing “implementation-defined functions” to “implementation-supplied functions” in the APPLICATION USAGE section.

19460 **NAME**

19461 abs — return an integer absolute value

19462 **SYNOPSIS**

19463 #include <stdlib.h>

19464 int abs(int i);

19465 **DESCRIPTION**

19466 CX The functionality described on this reference page is aligned with the ISO C standard. Any
19467 conflict between the requirements described here and the ISO C standard is unintentional. This
19468 volume of POSIX.1-2008 defers to the ISO C standard.

19469 The *abs()* function shall compute the absolute value of its integer operand, *i*. If the result cannot
19470 be represented, the behavior is undefined.

19471 **RETURN VALUE**19472 The *abs()* function shall return the absolute value of its integer operand.19473 **ERRORS**

19474 No errors are defined.

19475 **EXAMPLES**

19476 None.

19477 **APPLICATION USAGE**

19478 In two's-complement representation, the absolute value of the negative integer with largest
19479 magnitude {INT_MIN} might not be representable.

19480 **RATIONALE**

19481 None.

19482 **FUTURE DIRECTIONS**

19483 None.

19484 **SEE ALSO**19485 *fabs()*, *labs()*

19486 XBD <stdlib.h>

19487 **CHANGE HISTORY**

19488 First released in Issue 1. Derived from Issue 1 of the SVID.

19489 **Issue 6**

19490 Extensions beyond the ISO C standard are marked.

19491 **NAME**

19492 accept — accept a new connection on a socket

19493 **SYNOPSIS**

```
19494       #include <sys/socket.h>
19495       int accept(int socket, struct sockaddr *restrict address,
19496                 socklen_t *restrict address_len);
```

19497 **DESCRIPTION**

19498 The *accept()* function shall extract the first connection on the queue of pending connections,
 19499 create a new socket with the same socket type protocol and address family as the specified
 19500 socket, and allocate a new file descriptor for that socket.

19501 The *accept()* function takes the following arguments:

19502 *socket* Specifies a socket that was created with *socket()*, has been bound to an address
 19503 with *bind()*, and has issued a successful call to *listen()*.

19504 *address* Either a null pointer, or a pointer to a **sockaddr** structure where the address of
 19505 the connecting socket shall be returned.

19506 *address_len* Either a null pointer, if *address* is a null pointer, or a pointer to a **socklen_t**
 19507 object which on input specifies the length of the supplied **sockaddr** structure,
 19508 and on output specifies the length of the stored address.

19509 If *address* is not a null pointer, the address of the peer for the accepted connection shall be stored
 19510 in the **sockaddr** structure pointed to by *address*, and the length of this address shall be stored in
 19511 the object pointed to by *address_len*.

19512 If the actual length of the address is greater than the length of the supplied **sockaddr** structure,
 19513 the stored address shall be truncated.

19514 If the protocol permits connections by unbound clients, and the peer is not bound, then the
 19515 value stored in the object pointed to by *address* is unspecified.

19516 If the listen queue is empty of connection requests and O_NONBLOCK is not set on the file
 19517 descriptor for the socket, *accept()* shall block until a connection is present. If the *listen()* queue is
 19518 empty of connection requests and O_NONBLOCK is set on the file descriptor for the socket,
 19519 *accept()* shall fail and set *errno* to [EAGAIN] or [EWOULDBLOCK].

19520 The accepted socket cannot itself accept more connections. The original socket remains open and
 19521 can accept more connections.

19522 **RETURN VALUE**

19523 Upon successful completion, *accept()* shall return the non-negative file descriptor of the accepted
 19524 socket. Otherwise, -1 shall be returned and *errno* set to indicate the error.

19525 **ERRORS**

19526 The *accept()* function shall fail if:

19527 [EAGAIN] or [EWOULDBLOCK]

19528 O_NONBLOCK is set for the socket file descriptor and no connections are
 19529 present to be accepted.

19530 [EBADF] The *socket* argument is not a valid file descriptor.

19531 [ECONNABORTED]

19532 A connection has been aborted.

19533	[EINTR]	The <i>accept()</i> function was interrupted by a signal that was caught before a valid connection arrived.
19534		
19535	[EINVAL]	The <i>socket</i> is not accepting connections.
19536	[EMFILE]	All file descriptors available to the process are currently open.
19537	[ENFILE]	The maximum number of file descriptors in the system are already open.
19538	[ENOBUFS]	No buffer space is available.
19539	[ENOMEM]	There was insufficient memory available to complete the operation.
19540	[ENOTSOCK]	The <i>socket</i> argument does not refer to a socket.
19541	[EOPNOTSUPP]	The socket type of the specified socket does not support accepting connections.
19542		
19543		The <i>accept()</i> function may fail if:
19544	OB XSR [EPROTO]	A protocol error has occurred; for example, the STREAMS protocol stack has not been initialized.
19545		

19546 EXAMPLES

19547 None.

19548 APPLICATION USAGE

19549 When a connection is available, *select()* indicates that the file descriptor for the socket is ready
19550 for reading.

19551 RATIONALE

19552 None.

19553 FUTURE DIRECTIONS

19554 None.

19555 SEE ALSO

19556 *bind()*, *connect()*, *listen()*, *socket()*

19557 XBD <sys/socket.h>

19558 CHANGE HISTORY

19559 First released in Issue 6. Derived from the XNS, Issue 5.2 specification.

19560 The **restrict** keyword is added to the *accept()* prototype for alignment with the
19561 ISO/IEC 9899:1999 standard.

19562 Issue 7

19563 SD5-XBD-ERN-4 is applied, changing the definition of the [EMFILE] error.

19564 Austin Group Interpretation 1003.1-2001 #044 is applied, changing the “may fail” [ENOBUFS]
19565 and [ENOMEM] errors to become “shall fail” errors.

19566 Functionality relating to XSI STREAMS is marked obsolescent.

19567 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0018 [464] is applied.

+

NAME

access, faccessat — determine accessibility of a file relative to directory file descriptor

SYNOPSIS

```
#include <unistd.h>

int access(const char *path, int amode);
int faccessat(int fd, const char *path, int amode, int flag);
```

DESCRIPTION

The *access()* function shall check the file named by the pathname pointed to by the *path* argument for accessibility according to the bit pattern contained in *amode*, using the real user ID in place of the effective user ID and the real group ID in place of the effective group ID.

The value of *amode* is either the bitwise-inclusive OR of the access permissions to be checked (R_OK, W_OK, X_OK) or the existence test (F_OK).

If any access permissions are checked, each shall be checked individually, as described in XBD [Section 4.4](#) (on page 108), except that where that description refers to execute permission for a process with appropriate privileges, an implementation may indicate success for X_OK even if execute permission is not granted to any user.

The *faccessat()* function shall be equivalent to the *access()* function, except in the case where *path* specifies a relative path. In this case the file whose accessibility is to be determined shall be located relative to the directory associated with the file descriptor *fd* instead of the current working directory. If the file descriptor was opened without O_SEARCH, the function shall check whether directory searches are permitted using the current permissions of the directory underlying the file descriptor. If the file descriptor was opened with O_SEARCH, the function shall not perform the check.

If *faccessat()* is passed the special value AT_FDCWD in the *fd* parameter, the current working directory shall be used and the behavior shall be identical to a call to *access()*.

Values for *flag* are constructed by a bitwise-inclusive OR of flags from the following list, defined in *<fcntl.h>*:

AT_EACCESS	The checks for accessibility are performed using the effective user and group IDs instead of the real user and group ID as required in a call to <i>access()</i> .

RETURN VALUE

Upon successful completion, these functions shall return 0. Otherwise, these functions shall return -1 and set *errno* to indicate the error.

ERRORS

These functions shall fail if:

[EACCES]	Permission bits of the file mode do not permit the requested access, or search permission is denied on a component of the path prefix.
[ELOOP]	A loop exists in symbolic links encountered during resolution of the <i>path</i> argument.
[ENAMETOOLONG]	The length of a component of a pathname is longer than {NAME_MAX}.
[ENOENT]	A component of <i>path</i> does not name an existing file or <i>path</i> is an empty string.
[ENOTDIR]	A component of the path prefix names an existing file that is neither a directory nor a symbolic link to a directory, or the <i>path</i> argument contains at least one non- <i><slash></i> character and ends with one or more trailing <i><slash></i>

19612 characters and the last pathname component names an existing file that is
 19613 neither a directory nor a symbolic link to a directory.

19614 [EROFS] Write access is requested for a file on a read-only file system.

19615 The *faccessat()* function shall fail if:

19616 [EACCES] *fd* was not opened with O_SEARCH and the permissions of the directory
 19617 underlying *fd* do not permit directory searches.

19618 [EBADF] The *path* argument does not specify an absolute path and the *fd* argument is
 19619 neither AT_FDCWD nor a valid file descriptor open for reading or searching.

19620 [ENOTDIR] The *path* argument is not an absolute path and *fd* is a file descriptor associated +
 19621 with a non-directory file.

19622 These functions may fail if:

19623 [EINVAL] The value of the *amode* argument is invalid.

19624 [ELOOP] More than {SYMLOOP_MAX} symbolic links were encountered during
 19625 resolution of the *path* argument.

19626 [ENAMETOOLONG]
 19627 The length of a pathname exceeds {PATH_MAX}, or pathname resolution of a
 19628 symbolic link produced an intermediate result with a length that exceeds
 19629 {PATH_MAX}.

19630 [ETXTBSY] Write access is requested for a pure procedure (shared text) file that is being
 19631 executed.

19632 The *faccessat()* function may fail if:

19633 [EINVAL] The value of the *flag* argument is not valid. -

19634 EXAMPLES

19635 Testing for the Existence of a File

19636 The following example tests whether a file named **myfile** exists in the **/tmp** directory.

```
19637 #include <unistd.h>
19638 ...
19639 int result;
19640 const char *pathname = "/tmp/myfile";
19641 result = access (pathname, F_OK);
```

19642 APPLICATION USAGE

19643 Additional values of *amode* other than the set defined in the description may be valid; for
 19644 example, if a system has extended access controls.

19645 The use of the AT_EACCESS value for *flag* enables functionality not available in *access()*.

19646 RATIONALE

19647 In early proposals, some inadequacies in the *access()* function led to the creation of an *eaccess()*
 19648 function because:

- 19649 1. Historical implementations of *access()* do not test file access correctly when the process'
 19650 real user ID is superuser. In particular, they always return zero when testing execute
 19651 permissions without regard to whether the file is executable.

2. The superuser has complete access to all files on a system. As a consequence, programs started by the superuser and switched to the effective user ID with lesser privileges cannot use *access()* to test their file access permissions.

However, the historical model of *eaccess()* does not resolve problem (1), so this volume of POSIX.1-2008 now allows *access()* to behave in the desired way because several implementations have corrected the problem. It was also argued that problem (2) is more easily solved by using *open()*, *chdir()*, or one of the *exec* functions as appropriate and responding to the error, rather than creating a new function that would not be as reliable. Therefore, *eaccess()* is not included in this volume of POSIX.1-2008.

The sentence concerning appropriate privileges and execute permission bits reflects the two possibilities implemented by historical implementations when checking superuser access for *X_OK*.

New implementations are discouraged from returning *X_OK* unless at least one execution permission bit is set.

The purpose of the *faccessat()* function is to enable the checking of the accessibility of files in directories other than the current working directory without exposure to race conditions. Any part of the path of a file could be changed in parallel to a call to *access()*, resulting in unspecified behavior. By opening a file descriptor for the target directory and using the *faccessat()* function it can be guaranteed that the file tested for accessibility is located relative to the desired directory.

FUTURE DIRECTIONS

None.

SEE ALSO

chmod(), *fstatat()*

XBD Section 4.4 (on page 108), *<fcntl.h>*, *<unistd.h>*

CHANGE HISTORY

First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The [ELOOP] mandatory error condition is added.
- A second [ENAMETOOLONG] is added as an optional error condition.
- The [ETXTBSY] optional error condition is added.

The following changes were made to align with the IEEE P1003.1a draft standard:

- The [ELOOP] optional error condition is added.

Issue 7

Austin Group Interpretations 1003.1-2001 #046 and #143 are applied.

The *faccessat()* function is added from The Open Group Technical Standard, 2006, Extended API Set Part 2.

Changes are made to allow a directory to be opened for searching.

The [ENOTDIR] error condition is clarified to cover the condition where the last component of a pathname exists but is not a directory or a symbolic link to a directory.

19693

19694

19695

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0019 [461], XSH/TC1-2008/0020 [324], +
XSH/TC1-2008/0021 [278], XSH/TC1-2008/0022 [278], and XSH/TC1-2008/0023 [291] are +
applied.

19696 **NAME**

19697 acos, acosf, acosl — arc cosine functions

19698 **SYNOPSIS**

```
19699       #include <math.h>
19700       double acos(double x);
19701       float acosf(float x);
19702       long double acosl(long double x);
```

19703 **DESCRIPTION**

19704 **CX** The functionality described on this reference page is aligned with the ISO C standard. Any
 19705 conflict between the requirements described here and the ISO C standard is unintentional. This
 19706 volume of POSIX.1-2008 defers to the ISO C standard.

19707 These functions shall compute the principal value of the arc cosine of their argument *x*. The
 19708 value of *x* should be in the range $[-1,1]$.

19709 An application wishing to check for error situations should set *errno* to zero and call
 19710 *feclearexcept*(FE_ALL_EXCEPT) before calling these functions. On return, if *errno* is non-zero or
 19711 *fetestexcept*(FE_INVALID | FE_DIVBYZERO | FE_OVERFLOW | FE_UNDERFLOW) is non-
 19712 zero, an error has occurred.

19713 **RETURN VALUE**

19714 Upon successful completion, these functions shall return the arc cosine of *x*, in the range $[0,\pi]$
 19715 radians.

19716 **MX** For finite values of *x* not in the range $[-1,1]$, a domain error shall occur, and either a NaN (if
 19717 supported), or an implementation-defined value shall be returned.

19718 **MX** If *x* is NaN, a NaN shall be returned.

19719 If *x* is +1, +0 shall be returned.

19720 If *x* is $\pm\text{Inf}$, a domain error shall occur, and a NaN shall be returned.

19721 **ERRORS**

19722 These functions shall fail if:

19723 **MX** Domain Error The *x* argument is finite and is not in the range $[-1,1]$, or is $\pm\text{Inf}$.

19724 If the integer expression (*math_errhandling* & MATH_ERRNO) is non-zero,
 19725 then *errno* shall be set to [EDOM]. If the integer expression (*math_errhandling*
 19726 & MATH_ERREXCEPT) is non-zero, then the invalid floating-point exception
 19727 shall be raised.

19728 **EXAMPLES**

19729 None.

19730 **APPLICATION USAGE**

19731 On error, the expressions (*math_errhandling* & MATH_ERRNO) and (*math_errhandling* &
 19732 MATH_ERREXCEPT) are independent of each other, but at least one of them must be non-zero.

19733 **RATIONALE**

19734 None.

19735 **FUTURE DIRECTIONS**

19736 None.

19737 **SEE ALSO**19738 *cos()*, *feclearexcept()*, *fetestexcept()*, *isnan()*19739 XBD Section 4.19 (on page 116), **<math.h>**19740 **CHANGE HISTORY**

19741 First released in Issue 1. Derived from Issue 1 of the SVID.

19742 **Issue 5**19743 The DESCRIPTION is updated to indicate how an application should check for an error. This
19744 text was previously published in the APPLICATION USAGE section.19745 **Issue 6**19746 The *acosf()* and *acosl()* functions are added for alignment with the ISO/IEC 9899:1999 standard.19747 The DESCRIPTION, RETURN VALUE, ERRORS, and APPLICATION USAGE sections are
19748 revised to align with the ISO/IEC 9899:1999 standard.19749 IEC 60559:1989 standard floating-point extensions over the ISO/IEC 9899:1999 standard are
19750 marked.19751 **Issue 7**

19752 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0024 [320] is applied.

+

19753 **NAME**19754 `acosh, acoshf, acoshl` — inverse hyperbolic cosine functions19755 **SYNOPSIS**

```
19756 #include <math.h>
19757 double acosh(double x);
19758 float acoshf(float x);
19759 long double acoshl(long double x);
```

19760 **DESCRIPTION**

19761 CX The functionality described on this reference page is aligned with the ISO C standard. Any
 19762 conflict between the requirements described here and the ISO C standard is unintentional. This
 19763 volume of POSIX.1-2008 defers to the ISO C standard.

19764 These functions shall compute the inverse hyperbolic cosine of their argument x .

19765 An application wishing to check for error situations should set *errno* to zero and call
 19766 *feclearexcept*(FE_ALL_EXCEPT) before calling these functions. On return, if *errno* is non-zero or
 19767 *fetestexcept*(FE_INVALID | FE_DIVBYZERO | FE_OVERFLOW | FE_UNDERFLOW) is non-
 19768 zero, an error has occurred.

19769 **RETURN VALUE**

19770 Upon successful completion, these functions shall return the inverse hyperbolic cosine of their
 19771 argument.

19772 MX For finite values of $x < 1$, a domain error shall occur, and either a NaN (if supported), or an
 19773 implementation-defined value shall be returned.

19774 MX If x is NaN, a NaN shall be returned.

19775 If x is +1, +0 shall be returned.

19776 If x is +Inf, +Inf shall be returned.

19777 If x is -Inf, a domain error shall occur, and a NaN shall be returned. |

19778 **ERRORS**

19779 These functions shall fail if:

19780 MX Domain Error The x argument is finite and less than +1.0, or is -Inf.

19781 If the integer expression (*math_errhandling* & MATH_ERRNO) is non-zero,
 19782 then *errno* shall be set to [EDOM]. If the integer expression (*math_errhandling*
 19783 & MATH_ERREXCEPT) is non-zero, then the invalid floating-point exception
 19784 shall be raised.

19785 **EXAMPLES**

19786 None.

19787 **APPLICATION USAGE**

19788 On error, the expressions (*math_errhandling* & MATH_ERRNO) and (*math_errhandling* &
 19789 MATH_ERREXCEPT) are independent of each other, but at least one of them must be non-zero.

19790 **RATIONALE**

19791 None.

19792 **FUTURE DIRECTIONS**

19793 None.

19794 **SEE ALSO**19795 *cosh()*, *feclearexcept()*, *fetestexcept()*19796 XBD Section 4.19 (on page 116), **<math.h>**19797 **CHANGE HISTORY**

19798 First released in Issue 4, Version 2.

19799 **Issue 5**

19800 Moved from X/OPEN UNIX extension to BASE.

19801 **Issue 6**19802 The *acosh()* function is no longer marked as an extension.19803 The *acoshf()* and *acoshl()* functions are added for alignment with the ISO/IEC 9899:1999
19804 standard.19805 The DESCRIPTION, RETURN VALUE, ERRORS, and APPLICATION USAGE sections are
19806 revised to align with the ISO/IEC 9899:1999 standard.19807 IEC 60559:1989 standard floating-point extensions over the ISO/IEC 9899:1999 standard are
19808 marked.19809 **Issue 7**

19810 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0025 [320] is applied.

+

19811 **NAME**

19812 acosl — arc cosine functions

19813 **SYNOPSIS**

19814 #include <math.h>

19815 long double acosl(long double *x*);19816 **DESCRIPTION**19817 Refer to *acos()*.

NAME

aio_cancel — cancel an asynchronous I/O request

SYNOPSIS

```
#include <aio.h>
```

```
int aio_cancel(int fildev, struct aiocb *aiocbp);
```

DESCRIPTION

The *aio_cancel()* function shall attempt to cancel one or more asynchronous I/O requests currently outstanding against file descriptor *fildev*. The *aiocbp* argument points to the asynchronous I/O control block for a particular request to be canceled. If *aiocbp* is NULL, then all outstanding cancelable asynchronous I/O requests against *fildev* shall be canceled.

Normal asynchronous notification shall occur for asynchronous I/O operations that are successfully canceled. If there are requests that cannot be canceled, then the normal asynchronous completion process shall take place for those requests when they are completed.

For requested operations that are successfully canceled, the associated error status shall be set to [ECANCELED] and the return status shall be -1. For requested operations that are not successfully canceled, the *aiocbp* shall not be modified by *aio_cancel()*.

If *aiocbp* is not NULL, then if *fildev* does not have the same value as the file descriptor with which the asynchronous operation was initiated, unspecified results occur.

Which operations are cancelable is implementation-defined.

RETURN VALUE

The *aio_cancel()* function shall return the value AIO_CANCELED if the requested operation(s) were canceled. The value AIO_NOTCANCELED shall be returned if at least one of the requested operation(s) cannot be canceled because it is in progress. In this case, the state of the other operations, if any, referenced in the call to *aio_cancel()* is not indicated by the return value of *aio_cancel()*. The application may determine the state of affairs for these operations by using *aio_error()*. The value AIO_ALLDONE is returned if all of the operations have already completed. Otherwise, the function shall return -1 and set *errno* to indicate the error.

ERRORS

The *aio_cancel()* function shall fail if:

[EBADF] The *fildev* argument is not a valid file descriptor.

EXAMPLES

None.

APPLICATION USAGE

None.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

aio_read(), *aio_write()*

XBD [<aio.h>](#)

19859 **CHANGE HISTORY**

19860 First released in Issue 5. Included for alignment with the POSIX Realtime Extension.

19861 **Issue 6**

19862 The [ENOSYS] error condition has been removed as stubs need not be provided if an
19863 implementation does not support the Asynchronous Input and Output option.

19864 The APPLICATION USAGE section is added.

19865 IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/10 is applied, removing the words “to the
19866 calling process” in the RETURN VALUE section. The term was unnecessary and precluded
19867 threads.

19868 **Issue 7**

19869 The *aio_cancel()* function is moved from the Asynchronous Input and Output option to the Base.

19870 **NAME**

19871 **aiio_error** — retrieve errors status for an asynchronous I/O operation

19872 **SYNOPSIS**

19873 `#include <aiio.h>`

19874 `int aiio_error(const struct aiocb *aiocbp);`

19875 **DESCRIPTION**

19876 The *aiio_error()* function shall return the error status associated with the **aiocb** structure
 19877 referenced by the *aiocbp* argument. The error status for an asynchronous I/O operation is the
 19878 SIO *errno* value that would be set by the corresponding *read()*, *write()*, *fdatasync()*, or *fsync()*
 19879 operation. If the operation has not yet completed, then the error status shall be equal to
 19880 [EINPROGRESS].

19881 If the **aiocb** structure pointed to by *aiocbp* is not associated with an operation that has been
 19882 scheduled, the results are undefined.

19883 **RETURN VALUE**

19884 If the asynchronous I/O operation has completed successfully, then 0 shall be returned. If the
 19885 asynchronous operation has completed unsuccessfully, then the error status, as described for
 19886 SIO *read()*, *write()*, *fdatasync()*, and *fsync()*, shall be returned. If the asynchronous I/O operation has
 19887 not yet completed, then [EINPROGRESS] shall be returned.

19888 If the *aiio_error()* function fails, it shall return -1 and set *errno* to indicate the error.

19889 **ERRORS**

19890 The *aiio_error()* function may fail if:

19891 [EINVAL] The *aiocbp* argument does not refer to an asynchronous operation whose
 19892 return status has not yet been retrieved.

19893 **EXAMPLES**

19894 None.

19895 **APPLICATION USAGE**

19896 None.

19897 **RATIONALE**

19898 None.

19899 **FUTURE DIRECTIONS**

19900 None.

19901 **SEE ALSO**

19902 *aiio_cancel()*, *aiio_fsync()*, *aiio_read()*, *aiio_return()*, *aiio_write()*, *close()*, *exec*, *exit()*, *fork()*, *lio_listio()*,
 19903 *lseek()*, *read()*

19904 XBD **<aiio.h>**

19905 **CHANGE HISTORY**

19906 First released in Issue 5. Included for alignment with the POSIX Realtime Extension.

19907 **Issue 6**

19908 The [ENOSYS] error condition has been removed as stubs need not be provided if an
 19909 implementation does not support the Asynchronous Input and Output option.

19910 The APPLICATION USAGE section is added.

19911 **Issue 7**

19912 Austin Group Interpretation 1003.1-2001 #045 is applied.

19913 SD5-XSH-ERN-148 is applied.

19914 The *aio_error()* function is moved from the Asynchronous Input and Output option to the Base.

19915 **NAME**

19916 aio_fsync — asynchronous file synchronization

19917 **SYNOPSIS**

```
19918 #include <aio.h>
19919 int aio_fsync(int op, struct aiocb *aiocbp);
```

19920 **DESCRIPTION**

19921 The *aio_fsync()* function shall asynchronously perform a file synchronization operation, as
 19922 specified by the *op* argument, for I/O operations associated with the file indicated by the file
 19923 descriptor *aio_fildes* member of the **aiocb** structure referenced by the *aiocbp* argument and
 19924 queued at the time of the call to *aio_fsync()*. The function call shall return when the
 19925 synchronization request has been initiated or queued to the file or device (even when the data
 19926 cannot be synchronized immediately).

19927 SIO If *op* is O_DSYNC, all currently queued I/O operations shall be completed as if by a call to
 19928 *fdatsync()*; that is, as defined for synchronized I/O data integrity completion.

19929 FSC If *op* is O_SYNC, all currently queued I/O operations shall be completed as if by a call to *fsync()*;
 19930 FSC SIO that is, as defined for synchronized I/O file integrity completion. If the *aio_fsync()* function
 19931 fails, or if the operation queued by *aio_fsync()* fails, then outstanding I/O operations are not
 19932 guaranteed to have been completed.

19933 If *aio_fsync()* succeeds, then it is only the I/O that was queued at the time of the call to
 19934 *aio_fsync()* that is guaranteed to be forced to the relevant completion state. The completion of
 19935 subsequent I/O on the file descriptor is not guaranteed to be completed in a synchronized
 19936 fashion.

19937 The *aiocbp* argument refers to an asynchronous I/O control block. The *aiocbp* value may be used
 19938 as an argument to *aio_error()* and *aio_return()* in order to determine the error status and return
 19939 status, respectively, of the asynchronous operation while it is proceeding. When the request is
 19940 queued, the error status for the operation is [EINPROGRESS]. When all data has been
 19941 successfully transferred, the error status shall be reset to reflect the success or failure of the
 19942 operation. If the operation does not complete successfully, the error status for the operation shall
 19943 be set to indicate the error. The *aio_sigevent* member determines the asynchronous notification to
 19944 occur as specified in [Section 2.4.1](#) (on page 488) when all operations have achieved synchronized
 19945 I/O completion. All other members of the structure referenced by *aiocbp* are ignored. If the
 19946 control block referenced by *aiocbp* becomes an illegal address prior to asynchronous I/O
 19947 completion, then the behavior is undefined.

19948 If the *aio_fsync()* function fails or *aiocbp* indicates an error condition, data is not guaranteed to
 19949 have been successfully transferred.

19950 **RETURN VALUE**

19951 The *aio_fsync()* function shall return the value 0 if the I/O operation is successfully queued;
 19952 otherwise, the function shall return the value -1 and set *errno* to indicate the error.

19953 **ERRORS**

19954 The *aio_fsync()* function shall fail if:

19955 19956	[EAGAIN]	The requested asynchronous operation was not queued due to temporary resource limitations.
19957 19958	[EBADF]	The <i>aio_fildes</i> member of the aiocb structure referenced by the <i>aiocbp</i> argument is not a valid file descriptor open for writing.

19959	SIO	[EINVAL]	This implementation does not support synchronized I/O for this file.	
19960	FSC	[EINVAL]	The <i>aio_fildes</i> member of the aio_cb structure refers to a file on which an <i>fsync()</i> operation is not possible.	+
19961				
19962		[EINVAL]	A value of <i>op</i> other than O_DSYNC or O_SYNC was specified, or O_DSYNC was specified and the implementation does not provide runtime support for the Synchronized Input and Output option, or O_SYNC was specified and the implementation does not provide runtime support for the File Synchronization option.	
19963				
19964				
19965				
19966				
19967			In the event that any of the queued I/O operations fail, <i>aio_fsync()</i> shall return the error condition defined for <i>read()</i> and <i>write()</i> . The error is returned in the error status for the asynchronous operation, which can be retrieved using <i>aio_error()</i> .	
19968				
19969				
19970			EXAMPLES	
19971			None.	
19972			APPLICATION USAGE	
19973			None.	
19974			RATIONALE	
19975			None.	
19976			FUTURE DIRECTIONS	
19977			None.	
19978			SEE ALSO	
19979			<i>fcntl()</i> , <i>fdatasync()</i> , <i>fsync()</i> , <i>open()</i> , <i>read()</i> , <i>write()</i>	
19980			XBD <aio.h>	
19981			CHANGE HISTORY	
19982			First released in Issue 5. Included for alignment with the POSIX Realtime Extension.	
19983			Issue 6	
19984			The [ENOSYS] error condition has been removed as stubs need not be provided if an implementation does not support the Asynchronous Input and Output option.	
19985				
19986			The APPLICATION USAGE section is added.	
19987			IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/11 is applied, removing the words “to the calling process” in the RETURN VALUE section. The term was unnecessary and precluded threads.	
19988				
19989				
19990			Issue 7	
19991			The <i>aio_fsync()</i> function is moved from the Asynchronous Input and Output option to the Base.	
19992			POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0026 [98] and XSH/TC1-2008/0027	+
19993			[98] are applied.	

19994 **NAME**

19995 aio_read — asynchronous read from a file

19996 **SYNOPSIS**

19997 #include <aio.h>

19998 int aio_read(struct aiocb *aiocbp);

19999 **DESCRIPTION**

20000 The *aio_read()* function shall read *aiocbp->aio_nbytes* from the file associated with
 20001 *aiocbp->aio_fildes* into the buffer pointed to by *aiocbp->aio_buf*. The function call shall return
 20002 when the read request has been initiated or queued to the file or device (even when the data
 20003 cannot be delivered immediately).

20004 **PIO** If prioritized I/O is supported for this file, then the asynchronous operation shall be submitted
 20005 at a priority equal to a base scheduling priority minus *aiocbp->aio_reqprio*. If Thread Execution
 20006 Scheduling is not supported, then the base scheduling priority is that of the calling process;
 20007 **PIO TPS** otherwise, the base scheduling priority is that of the calling thread.

20008 The *aiocbp* value may be used as an argument to *aio_error()* and *aio_return()* in order to
 20009 determine the error status and return status, respectively, of the asynchronous operation while it
 20010 is proceeding. If an error condition is encountered during queuing, the function call shall return
 20011 without having initiated or queued the request. The requested operation takes place at the
 20012 absolute position in the file as given by *aio_offset*, as if *lseek()* were called immediately prior to
 20013 the operation with an *offset* equal to *aio_offset* and a *whence* equal to *SEEK_SET*. After a
 20014 successful call to enqueue an asynchronous I/O operation, the value of the file offset for the file
 20015 is unspecified.

20016 The *aio_sigevent* member specifies the notification which occurs when the request is completed.

20017 The *aiocbp->aio_lio_opcode* field shall be ignored by *aio_read()*.

20018 The *aiocbp* argument points to an **aiocb** structure. If the buffer pointed to by *aiocbp->aio_buf* or
 20019 the control block pointed to by *aiocbp* becomes an illegal address prior to asynchronous I/O
 20020 completion, then the behavior is undefined.

20021 Simultaneous asynchronous operations using the same *aiocbp* produce undefined results.

20022 **SIO** If synchronized I/O is enabled on the file associated with *aiocbp->aio_fildes*, the behavior of this
 20023 function shall be according to the definitions of synchronized I/O data integrity completion and
 20024 synchronized I/O file integrity completion.

20025 For any system action that changes the process memory space while an asynchronous I/O is
 20026 outstanding to the address range being changed, the result of that action is undefined.

20027 For regular files, no data transfer shall occur past the offset maximum established in the open
 20028 file description associated with *aiocbp->aio_fildes*.

20029 **RETURN VALUE**

20030 The *aio_read()* function shall return the value zero if the I/O operation is successfully queued;
 20031 otherwise, the function shall return the value -1 and set *errno* to indicate the error.

20032 **ERRORS**

20033 The *aio_read()* function shall fail if:

20034 [EAGAIN] The requested asynchronous I/O operation was not queued due to system
 20035 resource limitations.

20036 Each of the following conditions may be detected synchronously at the time of the call to
 20037 *aio_read()*, or asynchronously. If any of the conditions below are detected synchronously, the

20038 *aio_read()* function shall return `-1` and set *errno* to the corresponding value. If any of the
 20039 conditions below are detected asynchronously, the return status of the asynchronous operation
 20040 is set to `-1`, and the error status of the asynchronous operation is set to the corresponding value.

20041 [EBADF] The *aiocbp*→*aio_fildes* argument is not a valid file descriptor open for reading.

20042 [EINVAL] The file offset value implied by *aiocbp*→*aio_offset* would be invalid,
 20043 PIO *aiocbp*→*aio_reqprio* is not a valid value, or *aiocbp*→*aio_nbytes* is an invalid
 20044 value.

20045 In the case that the *aio_read()* successfully queues the I/O operation but the operation is
 20046 subsequently canceled or encounters an error, the return status of the asynchronous operation is
 20047 one of the values normally returned by the *read()* function call. In addition, the error status of
 20048 the asynchronous operation is set to one of the error statuses normally set by the *read()* function
 20049 call, or one of the following values:

20050 [EBADF] The *aiocbp*→*aio_fildes* argument is not a valid file descriptor open for reading.

20051 [ECANCELED] The requested I/O was canceled before the I/O completed due to an explicit
 20052 *aio_cancel()* request.

20053 [EINVAL] The file offset value implied by *aiocbp*→*aio_offset* would be invalid.

20054 The following condition may be detected synchronously or asynchronously:

20055 [EOVERFLOW] The file is a regular file, *aiocbp*→*aio_nbytes* is greater than 0, and the starting
 20056 offset in *aiocbp*→*aio_offset* is before the end-of-file and is at or beyond the
 20057 offset maximum in the open file description associated with *aiocbp*→*aio_fildes*.

20058 EXAMPLES

20059 None.

20060 APPLICATION USAGE

20061 None.

20062 RATIONALE

20063 None.

20064 FUTURE DIRECTIONS

20065 None.

20066 SEE ALSO

20067 *aio_cancel()*, *aio_error()*, *lio_listio()*, *aio_return()*, *aio_write()*, *close()*, *exec*, *exit()*, *fork()*, *lseek()*,
 20068 *read()*

20069 XBD <[aio.h](#)>

20070 CHANGE HISTORY

20071 First released in Issue 5. Included for alignment with the POSIX Realtime Extension.

20072 Large File Summit extensions are added.

20073 Issue 6

20074 The [ENOSYS] error condition has been removed as stubs need not be provided if an
 20075 implementation does not support the Asynchronous Input and Output option.

20076 The APPLICATION USAGE section is added.

20077	The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:
20078	
20079	• In the DESCRIPTION, text is added to indicate setting of the offset maximum in the open file description. This change is to support large files.
20080	
20081	• In the ERRORS section, the [EOVERFLOW] condition is added. This change is to support large files.
20082	
20083	IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/12 is applied, rewording the DESCRIPTION when prioritized I/O is supported to account for threads, and removing the words “to the calling process” in the RETURN VALUE section.
20084	
20085	
20086	IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/13 is applied, updating the [EINVAL] error, so that detection of an [EINVAL] error for an invalid value of <i>aiocbp</i> → <i>aio_reqprio</i> is only required if the Prioritized Input and Output option is supported.
20087	
20088	
20089	Issue 7
20090	Austin Group Interpretation 1003.1-2001 #082 is applied.
20091	The <i>aio_read()</i> function is moved from the Asynchronous Input and Output option to the Base.

20092 **NAME**

20093 aio_return — retrieve return status of an asynchronous I/O operation

20094 **SYNOPSIS**

```
20095 #include <aio.h>
20096 ssize_t aio_return(struct aiocb *aiocbp);
```

20097 **DESCRIPTION**

20098 The *aio_return()* function shall return the return status associated with the **aiocb** structure
 20099 referenced by the *aiocbp* argument. The return status for an asynchronous I/O operation is the
 20100 value that would be returned by the corresponding *read()*, *write()*, or *fsync()* function call. If the
 20101 error status for the operation is equal to [EINPROGRESS], then the return status for the
 20102 operation is undefined. The *aio_return()* function may be called exactly once to retrieve the
 20103 return status of a given asynchronous operation; thereafter, if the same **aiocb** structure is used in
 20104 a call to *aio_return()* or *aio_error()*, an error may be returned. When the **aiocb** structure referred
 20105 to by *aiocbp* is used to submit another asynchronous operation, then *aio_return()* may be
 20106 successfully used to retrieve the return status of that operation.

20107 **RETURN VALUE**

20108 If the asynchronous I/O operation has completed, then the return status, as described for *read()*,
 20109 *write()*, and *fsync()*, shall be returned. If the asynchronous I/O operation has not yet completed,
 20110 the results of *aio_return()* are undefined.

20111 If the *aio_return()* function fails, it shall return -1 and set *errno* to indicate the error.

20112 **ERRORS**

20113 The *aio_return()* function may fail if:

20114 [EINVAL] The *aiocbp* argument does not refer to an asynchronous operation whose
 20115 return status has not yet been retrieved.

20116 **EXAMPLES**

20117 None.

20118 **APPLICATION USAGE**

20119 None.

20120 **RATIONALE**

20121 None.

20122 **FUTURE DIRECTIONS**

20123 None.

20124 **SEE ALSO**

20125 *aio_cancel()*, *aio_error()*, *aio_fsync()*, *aio_read()*, *aio_write()*, *close()*, *exec*, *exit()*, *fork()*, *lio_listio()*,
 20126 *lseek()*, *read()*

20127 XBD [<aio.h>](#)

20128 **CHANGE HISTORY**

20129 First released in Issue 5. Included for alignment with the POSIX Realtime Extension.

20130 **Issue 6**

20131 The [ENOSYS] error condition has been removed as stubs need not be provided if an
 20132 implementation does not support the Asynchronous Input and Output option.

20133 The APPLICATION USAGE section is added.

20134 The [EINVAL] error condition is made optional. This is for consistency with the DESCRIPTION.

20135 **Issue 7**

20136 SD5-XSH-ERN-148 is applied.

20137 The *aio_return()* function is moved from the Asynchronous Input and Output option to the Base.

NAME

aio_suspend — wait for an asynchronous I/O request

SYNOPSIS

```
#include <aio.h>

int aio_suspend(const struct aiocb *const list[], int nent,
               const struct timespec *timeout);
```

DESCRIPTION

The *aio_suspend()* function shall suspend the calling thread until at least one of the asynchronous I/O operations referenced by the *list* argument has completed, until a signal interrupts the function, or, if *timeout* is not NULL, until the time interval specified by *timeout* has passed. If any of the **aiocb** structures in the list correspond to completed asynchronous I/O operations (that is, the error status for the operation is not equal to [EINPROGRESS]) at the time of the call, the function shall return without suspending the calling thread. The *list* argument is an array of pointers to asynchronous I/O control blocks. The *nent* argument indicates the number of elements in the array. Each **aiocb** structure pointed to has been used in initiating an asynchronous I/O request via *aio_read()*, *aio_write()*, or *lio_listio()*. This array may contain null pointers, which are ignored. If this array contains pointers that refer to **aiocb** structures that have not been used in submitting asynchronous I/O, the effect is undefined.

If the time interval indicated in the **timespec** structure pointed to by *timeout* passes before any of the I/O operations referenced by *list* are completed, then *aio_suspend()* shall return with an error.

MON If the Monotonic Clock option is supported, the clock that shall be used to measure this time interval shall be the CLOCK_MONOTONIC clock.

RETURN VALUE

If the *aio_suspend()* function returns after one or more asynchronous I/O operations have completed, the function shall return zero. Otherwise, the function shall return a value of -1 and set *errno* to indicate the error.

The application may determine which asynchronous I/O completed by scanning the associated error and return status using *aio_error()* and *aio_return()*, respectively.

ERRORS

The *aio_suspend()* function shall fail if:

[EAGAIN] No asynchronous I/O indicated in the list referenced by *list* completed in the time interval indicated by *timeout*.

[EINTR] A signal interrupted the *aio_suspend()* function. Note that, since each asynchronous I/O operation may possibly provoke a signal when it completes, this error return may be caused by the completion of one (or more) of the very I/O operations being awaited.

EXAMPLES

None.

APPLICATION USAGE

None.

RATIONALE

None.

20180 **FUTURE DIRECTIONS**

20181 None.

20182 **SEE ALSO**20183 *aio_read()*, *aio_write()*, *lio_listio()*

20184 XBD <aio.h>

20185 **CHANGE HISTORY**

20186 First released in Issue 5. Included for alignment with the POSIX Realtime Extension.

20187 **Issue 6**20188 The [ENOSYS] error condition has been removed as stubs need not be provided if an
20189 implementation does not support the Asynchronous Input and Output option.

20190 The APPLICATION USAGE section is added.

20191 The DESCRIPTION is updated for alignment with IEEE Std 1003.1j-2000 by specifying that the
20192 CLOCK_MONOTONIC clock, if supported, is used.20193 **Issue 7**20194 The *aio_suspend()* function is moved from the Asynchronous Input and Output option to the
20195 Base.

20196 **NAME**

20197 aio_write — asynchronous write to a file

20198 **SYNOPSIS**

20199 #include <aio.h>

20200 int aio_write(struct aiocb *aiocbp);

20201 **DESCRIPTION**

20202 The *aio_write()* function shall write *aiocbp->aio_nbytes* to the file associated with
 20203 *aiocbp->aio_fildes* from the buffer pointed to by *aiocbp->aio_buf*. The function shall return when
 20204 the write request has been initiated or, at a minimum, queued to the file or device.

20205 **PIO** If prioritized I/O is supported for this file, then the asynchronous operation shall be submitted
 20206 at a priority equal to a base scheduling priority minus *aiocbp->aio_reqprio*. If Thread Execution
 20207 Scheduling is not supported, then the base scheduling priority is that of the calling process;
 20208 **PIO TPS** otherwise, the base scheduling priority is that of the calling thread.

20209 The *aiocbp* argument may be used as an argument to *aio_error()* and *aio_return()* in order to
 20210 determine the error status and return status, respectively, of the asynchronous operation while it
 20211 is proceeding.

20212 The *aiocbp* argument points to an **aiocb** structure. If the buffer pointed to by *aiocbp->aio_buf* or
 20213 the control block pointed to by *aiocbp* becomes an illegal address prior to asynchronous I/O
 20214 completion, then the behavior is undefined.

20215 If O_APPEND is not set for the file descriptor *aio_fildes*, then the requested operation shall take
 20216 place at the absolute position in the file as given by *aio_offset*, as if *lseek()* were called
 20217 immediately prior to the operation with an *offset* equal to *aio_offset* and a *whence* equal to
 20218 SEEK_SET. If O_APPEND is set for the file descriptor, or if *aio_fildes* is associated with a device
 20219 that is incapable of seeking, write operations append to the file in the same order as the calls
 20220 were made, except under circumstances described in [Section 2.8.2](#). After a successful call to
 20221 enqueue an asynchronous I/O operation, the value of the file offset for the file is unspecified.

20222 The *aio_sigevent* member specifies the notification which occurs when the request is completed.

20223 The *aiocbp->aio_lio_opcode* field shall be ignored by *aio_write()*.

20224 Simultaneous asynchronous operations using the same *aiocbp* produce undefined results.

20225 **SIO** If synchronized I/O is enabled on the file associated with *aiocbp->aio_fildes*, the behavior of this
 20226 function shall be according to the definitions of synchronized I/O data integrity completion, and
 20227 synchronized I/O file integrity completion.

20228 For any system action that changes the process memory space while an asynchronous I/O is
 20229 outstanding to the address range being changed, the result of that action is undefined.

20230 For regular files, no data transfer shall occur past the offset maximum established in the open
 20231 file description associated with *aiocbp->aio_fildes*.

20232 **RETURN VALUE**

20233 The *aio_write()* function shall return the value zero if the I/O operation is successfully queued;
 20234 otherwise, the function shall return the value -1 and set *errno* to indicate the error.

20235 **ERRORS**

20236 The *aio_write()* function shall fail if:

20237 [EAGAIN] The requested asynchronous I/O operation was not queued due to system
 20238 resource limitations.

20239 Each of the following conditions may be detected synchronously at the time of the call to

20240 *aio_write()*, or asynchronously. If any of the conditions below are detected synchronously, the
 20241 *aio_write()* function shall return `-1` and set *errno* to the corresponding value. If any of the
 20242 conditions below are detected asynchronously, the return status of the asynchronous operation
 20243 shall be set to `-1`, and the error status of the asynchronous operation is set to the corresponding
 20244 value.

20245 [EBADF] The *aiocbp*→*aio_fildes* argument is not a valid file descriptor open for writing.

20246 [EINVAL] The file offset value implied by *aiocbp*→*aio_offset* would be invalid,
 20247 PIO *aiocbp*→*aio_reqprio* is not a valid value, or *aiocbp*→*aio_nbytes* is an invalid
 20248 value.

20249 In the case that the *aio_write()* successfully queues the I/O operation, the return status of the
 20250 asynchronous operation shall be one of the values normally returned by the *write()* function call.
 20251 If the operation is successfully queued but is subsequently canceled or encounters an error, the
 20252 error status for the asynchronous operation contains one of the values normally set by the
 20253 *write()* function call, or one of the following:

20254 [EBADF] The *aiocbp*→*aio_fildes* argument is not a valid file descriptor open for writing.

20255 [EINVAL] The file offset value implied by *aiocbp*→*aio_offset* would be invalid.

20256 [ECANCELED] The requested I/O was canceled before the I/O completed due to an explicit
 20257 *aio_cancel()* request.

20258 The following condition may be detected synchronously or asynchronously:

20259 [EFBIG] The file is a regular file, *aiocbp*→*aio_nbytes* is greater than 0, and the starting
 20260 offset in *aiocbp*→*aio_offset* is at or beyond the offset maximum in the open file
 20261 description associated with *aiocbp*→*aio_fildes*.

20262 EXAMPLES

20263 None.

20264 APPLICATION USAGE

20265 None.

20266 RATIONALE

20267 None.

20268 FUTURE DIRECTIONS

20269 None.

20270 SEE ALSO

20271 Section 2.8.2 (on page 503), *aio_cancel()*, *aio_error()*, *aio_read()*, *aio_return()*, *close()*, *exec*, *exit()*,
 20272 *fork()*, *lio_listio()*, *lseek()*, *write()*

20273 XBD <aio.h>

20274 CHANGE HISTORY

20275 First released in Issue 5. Included for alignment with the POSIX Realtime Extension.

20276 Large File Summit extensions are added.

20277 Issue 6

20278 The [ENOSYS] error condition has been removed as stubs need not be provided if an
 20279 implementation does not support the Asynchronous Input and Output option.

20280 The APPLICATION USAGE section is added.

20281 The following new requirements on POSIX implementations derive from alignment with the

20282	Single UNIX Specification:
20283	• In the DESCRIPTION, text is added to indicate that for regular files no data transfer occurs
20284	past the offset maximum established in the open file description associated with
20285	<i>aiocbp</i> → <i>aio_fildes</i> .
20286	• The [EFBIG] error is added as part of the large file support extensions.
20287	IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/14 is applied, rewording the
20288	DESCRIPTION when prioritized I/O is supported to account for threads, and removing the
20289	words “to the calling process” in the RETURN VALUE section.
20290	IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/15 is applied, updating the [EINVAL]
20291	error, so that detection of an [EINVAL] error for an invalid value of <i>aiocbp</i> → <i>aio_reqprio</i> is only
20292	required if the Prioritized Input and Output option is supported.
20293	Issue 7
20294	Austin Group Interpretation 1003.1-2001 #082 is applied.
20295	The <i>aio_write()</i> function is moved from the Asynchronous Input and Output option to the Base.
20296	POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0028 [317] is applied.

+

20297 **NAME**

20298 alarm — schedule an alarm signal

20299 **SYNOPSIS**

20300 #include <unistd.h>

20301 unsigned alarm(unsigned seconds);

20302 **DESCRIPTION**

20303 The *alarm()* function shall cause the system to generate a SIGALRM signal for the process after
 20304 the number of realtime seconds specified by *seconds* have elapsed. Processor scheduling delays
 20305 may prevent the process from handling the signal as soon as it is generated.

20306 If *seconds* is 0, a pending alarm request, if any, is canceled.

20307 Alarm requests are not stacked; only one SIGALRM generation can be scheduled in this manner.
 20308 If the SIGALRM signal has not yet been generated, the call shall result in rescheduling the time
 20309 at which the SIGALRM signal is generated.

20310 XSI Interactions between *alarm()* and *setitimer()* are unspecified.

20311 **RETURN VALUE**

20312 If there is a previous *alarm()* request with time remaining, *alarm()* shall return a non-zero value
 20313 that is the number of seconds until the previous request would have generated a SIGALRM
 20314 signal. Otherwise, *alarm()* shall return 0.

20315 **ERRORS**

20316 The *alarm()* function is always successful, and no return value is reserved to indicate an error.

20317 **EXAMPLES**

20318 None.

20319 **APPLICATION USAGE**

20320 The *fork()* function clears pending alarms in the child process. A new process image created by
 20321 one of the *exec* functions inherits the time left to an alarm signal in the image of the old process.

20322 Application developers should note that the type of the argument *seconds* and the return value of
 20323 *alarm()* is **unsigned**. That means that a Strictly Conforming POSIX System Interfaces
 20324 Application cannot pass a value greater than the minimum guaranteed value for {UINT_MAX},
 20325 which the ISO C standard sets as 65 535, and any application passing a larger value is restricting
 20326 its portability. A different type was considered, but historical implementations, including those
 20327 with a 16-bit **int** type, consistently use either **unsigned** or **int**.

20328 Application developers should be aware of possible interactions when the same process uses
 20329 both the *alarm()* and *sleep()* functions.

20330 **RATIONALE**

20331 Many historical implementations (including Version 7 and System V) allow an alarm to occur up
 20332 to a second early. Other implementations allow alarms up to half a second or one clock tick
 20333 early or do not allow them to occur early at all. The latter is considered most appropriate, since it
 20334 gives the most predictable behavior, especially since the signal can always be delayed for an
 20335 indefinite amount of time due to scheduling. Applications can thus choose the *seconds* argument
 20336 as the minimum amount of time they wish to have elapse before the signal.

20337 The term “realtime” here and elsewhere (*sleep()*, *times()*) is intended to mean “wall clock” time
 20338 as common English usage, and has nothing to do with “realtime operating systems”. It is in
 20339 contrast to *virtual time*, which could be misinterpreted if just *time* were used.

20340 In some implementations, including 4.3 BSD, very large values of the *seconds* argument are
 20341 silently rounded down to an implementation-specific maximum value. This maximum is large

20342 enough (to the order of several months) that the effect is not noticeable.

20343 There were two possible choices for alarm generation in multi-threaded applications: generation
20344 for the calling thread or generation for the process. The first option would not have been
20345 particularly useful since the alarm state is maintained on a per-process basis and the alarm that
20346 is established by the last invocation of *alarm()* is the only one that would be active.

20347 Furthermore, allowing generation of an asynchronous signal for a thread would have
20348 introduced an exception to the overall signal model. This requires a compelling reason in order
20349 to be justified.

20350 FUTURE DIRECTIONS

20351 None.

20352 SEE ALSO

20353 *alarm()*, *exec*, *fork()*, *getitimer()*, *pause()*, *sigaction()*, *sleep()*

20354 XBD [`<signal.h>`](#), [`<unistd.h>`](#)

20355 CHANGE HISTORY

20356 First released in Issue 1. Derived from Issue 1 of the SVID.

20357 Issue 6

20358 The following new requirements on POSIX implementations derive from alignment with the
20359 Single UNIX Specification:

- 20360 • The DESCRIPTION is updated to indicate that interactions with the *setitimer()*, *ualarm()*,
20361 and *usleep()* functions are unspecified.

20362 IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/16 is applied, replacing “an
20363 implementation-defined maximum value” with “an implementation-specific maximum value”
20364 in the RATIONALE.

20365 **NAME**

20366 alphasort, scandir — scan a directory

20367 **SYNOPSIS**

```
20368     #include <dirent.h>

20369     int alphasort(const struct dirent **d1, const struct dirent **d2);
20370     int scandir(const char *dir, struct dirent ***namelist,
20371                int (*sel)(const struct dirent *),
20372                int (*compar)(const struct dirent **, const struct dirent **));
```

20373 **DESCRIPTION**

20374 The *alphasort()* function can be used as the comparison function for the *scandir()* function to sort
 20375 the directory entries, *d1* and *d2*, into alphabetical order. Sorting happens as if by calling the
 20376 *strcoll()* function on the *d_name* element of the **dirent** structures passed as the two parameters. If
 20377 the *strcoll()* function fails, the return value of *alphasort()* is unspecified.

20378 The *alphasort()* function shall not change the setting of *errno* if successful. Since no return value
 20379 is reserved to indicate an error, an application wishing to check for error situations should set
 20380 *errno* to 0, then call *alphasort()*, then check *errno*.

20381 The *scandir()* function shall scan the directory *dir*, calling the function referenced by *sel* on each
 20382 directory entry. Entries for which the function referenced by *sel* returns non-zero shall be stored
 20383 in strings allocated as if by a call to *malloc()*, and sorted as if by a call to *qsort()* with the
 20384 comparison function *compar*, except that *compar* need not provide total ordering. The strings are
 20385 collected in array *namelist* which shall be allocated as if by a call to *malloc()*. If *sel* is a null
 20386 pointer, all entries shall be selected. If the comparison function *compar* does not provide total
 20387 ordering, the order in which the directory entries are stored is unspecified.

20388 **RETURN VALUE**

20389 Upon successful completion, the *alphasort()* function shall return an integer greater than, equal
 20390 to, or less than 0, according to whether the name of the directory entry pointed to by *d1* is
 20391 lexically greater than, equal to, or less than the directory pointed to by *d2* when both are
 20392 interpreted as appropriate to the current locale. There is no return value reserved to indicate an
 20393 error.

20394 Upon successful completion, the *scandir()* function shall return the number of entries in the
 20395 array and a pointer to the array through the parameter *namelist*. Otherwise, the *scandir()*
 20396 function shall return -1.

20397 **ERRORS**

20398 The *scandir()* function shall fail if:

20399 20400	[EACCES]	Search permission is denied for the component of the path prefix of <i>dir</i> or read permission is denied for <i>dir</i> .
20401 20402	[ELOOP]	A loop exists in symbolic links encountered during resolution of the <i>dir</i> argument.
20403 20404	[ENAMETOOLONG]	The length of a component of a pathname is longer than {NAME_MAX}.
20405 20406	[ENOENT]	A component of <i>dir</i> does not name an existing directory or <i>dir</i> is an empty string.
20407	[ENOMEM]	Insufficient storage space is available.

20408	[ENOTDIR]	A component of <i>dir</i> names an existing file that is neither a directory nor a	
20409		symbolic link to a directory.	
20410	[EOVERFLOW]	One of the values to be returned or passed to a callback function cannot be	
20411		represented correctly.	
20412	The <i>scandir()</i> function may fail if:		
20413	[ELOOP]	More than {SYMLOOP_MAX} symbolic links were encountered during	
20414		resolution of the <i>dir</i> argument.	
20415	[EMFILE]	All file descriptors available to the process are currently open.	
20416	[ENAMETOOLONG]		
20417		The length of a pathname exceeds {PATH_MAX}, or pathname resolution of a	
20418		symbolic link produced an intermediate result with a length that exceeds	
20419		{PATH_MAX}.	
20420	[ENFILE]	Too many files are currently open in the system.	

EXAMPLES

An example to print the files in the current directory:

```
#include <dirent.h>
#include <stdio.h>
#include <stdlib.h>
...
struct dirent **namelist;
int i, n;

n = scandir(".", &namelist, 0, alphasort);
if (n < 0)
    perror("scandir");
else {
    for (i = 0; i < n; i++) {
        printf("%s\n", namelist[i]->d_name);
        free(namelist[i]);
    }
    free(namelist);
...

```

APPLICATION USAGE

If *dir* contains filenames that do not form character strings, or which contain characters outside the domain of the collating sequence of the current locale, the *alphasort()* function need not provide a total ordering. This condition is not possible if all filenames within the directory consist only of characters from the portable filename character set.

The *scandir()* function may allocate dynamic storage during its operation. If *scandir()* is forcibly terminated, such as by *longjmp()* or *siglongjmp()* being executed by the function pointed to by *sel* or *compar*, or by an interrupt routine, *scandir()* does not have a chance to free that storage, so it remains permanently allocated. A safe way to handle interrupts is to store the fact that an interrupt has occurred, then wait until *scandir()* returns to act on the interrupt.

For functions that allocate memory as if by *malloc()*, the application should release such memory when it is no longer required by a call to *free()*. For *scandir()*, this is *namelist* (including all of the individual strings in *namelist*).

20453 **RATIONALE**

20454 None.

20455 **FUTURE DIRECTIONS**

20456 None.

20457 **SEE ALSO**20458 *qsort()*, *strcoll()*

20459 XBD <dirent.h>

20460 **CHANGE HISTORY**

20461 First released in Issue 7.

20462 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0029 [324], XSH/TC1-2008/0030 [404], +
20463 XSH/TC1-2008/0031 [393], and XSH/TC1-2008/0032 [291] are applied.

20464 **NAME**

20465 asctime, asctime_r — convert date and time to a string

20466 **SYNOPSIS**

```

20467 OB      #include <time.h>
20468          char *asctime(const struct tm *timeptr);
20469 OB CX    char *asctime_r(const struct tm *restrict tm, char *restrict buf);

```

20470 **DESCRIPTION**

20471 CX For *asctime()*: The functionality described on this reference page is aligned with the ISO C
 20472 standard. Any conflict between the requirements described here and the ISO C standard is
 20473 unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

20474 The *asctime()* function shall convert the broken-down time in the structure pointed to by *timeptr*
 20475 into a string in the form:

20476 Sun Sep 16 01:03:52 1973\n\0

20477 using the equivalent of the following algorithm:

```

20478 char *asctime(const struct tm *timeptr)
20479 {
20480     static char wday_name[7][3] = {
20481         "Sun", "Mon", "Tue", "Wed", "Thu", "Fri", "Sat"
20482     };
20483     static char mon_name[12][3] = {
20484         "Jan", "Feb", "Mar", "Apr", "May", "Jun",
20485         "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"
20486     };
20487     static char result[26];
20488     sprintf(result, "%.3s %.3s%3d %.2d:%.2d:%.2d %d\n",
20489         wday_name[timeptr->tm_wday],
20490         mon_name[timeptr->tm_mon],
20491         timeptr->tm_mday, timeptr->tm_hour,
20492         timeptr->tm_min, timeptr->tm_sec,
20493         1900 + timeptr->tm_year);
20494     return result;
20495 }

```

20496 However, the behavior is undefined if *timeptr->tm_wday* or *timeptr->tm_mon* are not within the
 20497 normal ranges as defined in **<time.h>**, or if *timeptr->tm_year* exceeds {INT_MAX}-1990, or if the
 20498 above algorithm would attempt to generate more than 26 bytes of output (including the
 20499 terminating null).

20500 The **tm** structure is defined in the **<time.h>** header.

20501 CX The *asctime()*, *ctime()*, *gmtime()*, and *localtime()* functions shall return values in one of two static
 20502 objects: a broken-down time structure and an array of type **char**. Execution of any of the
 20503 functions may overwrite the information returned in either of these objects by any of the other
 20504 functions.

20505 The *asctime()* function need not be thread-safe.

20506 The *asctime_r()* function shall convert the broken-down time in the structure pointed to by *tm*
 20507 into a string (of the same form as that returned by *asctime()*, and with the same undefined
 20508 behavior when input or output is out of range) that is placed in the user-supplied buffer pointed

20509 to by *buf* (which shall contain at least 26 bytes) and then return *buf*.

20510 RETURN VALUE

20511 CX Upon successful completion, *asctime()* shall return a pointer to the string. If the function is
20512 unsuccessful, it shall return NULL.

20513 Upon successful completion, *asctime_r()* shall return a pointer to a character string containing
20514 the date and time. This string is pointed to by the argument *buf*. If the function is unsuccessful,
20515 it shall return NULL.

20516 ERRORS

20517 No errors are defined.

20518 EXAMPLES

20519 None.

20520 APPLICATION USAGE

20521 These functions are included only for compatibility with older implementations. They have
20522 undefined behavior if the resulting string would be too long, so the use of these functions
20523 should be discouraged. On implementations that do not detect output string length overflow, it
20524 is possible to overflow the output buffers in such a way as to cause applications to fail, or
20525 possible system security violations. Also, these functions do not support localized date and time
20526 formats. To avoid these problems, applications should use *strftime()* to generate strings from
20527 broken-down times.

20528 Values for the broken-down time structure can be obtained by calling *gmtime()* or *localtime()*.

20529 The *asctime_r()* function is thread-safe and shall return values in a user-supplied buffer instead
20530 of possibly using a static data area that may be overwritten by each call.

20531 RATIONALE

20532 The standard developers decided to mark the *asctime()* and *asctime_r()* functions obsolescent
20533 even though *asctime()* is in the ISO C standard due to the possibility of buffer overflow. The
20534 ISO C standard also provides the *strftime()* function which can be used to avoid these problems.

20535 FUTURE DIRECTIONS

20536 These functions may be removed in a future version.

20537 SEE ALSO

20538 [*clock\(\)*](#), [*ctime\(\)*](#), [*difftime\(\)*](#), [*gmtime\(\)*](#), [*localtime\(\)*](#), [*mktime\(\)*](#), [*strftime\(\)*](#), [*strptime\(\)*](#), [*time\(\)*](#), [*utime\(\)*](#)

20539 XBD [**<time.h>**](#)

20540 CHANGE HISTORY

20541 First released in Issue 1. Derived from Issue 1 of the SVID.

20542 Issue 5

20543 Normative text previously in the APPLICATION USAGE section is moved to the
20544 DESCRIPTION.

20545 The *asctime_r()* function is included for alignment with the POSIX Threads Extension.

20546 A note indicating that the *asctime()* function need not be reentrant is added to the
20547 DESCRIPTION.

20548 Issue 6

20549 The *asctime_r()* function is marked as part of the Thread-Safe Functions option.

20550 Extensions beyond the ISO C standard are marked.

20551 The APPLICATION USAGE section is updated to include a note on the thread-safe function and

- 20552 its avoidance of possibly using a static data area.
- 20553 The DESCRIPTION of *asctime_r()* is updated to describe the format of the string returned.
- 20554 The **restrict** keyword is added to the *asctime_r()* prototype for alignment with the
20555 ISO/IEC 9899:1999 standard
- 20556 IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/17 is applied, adding the CX extension in
20557 the RETURN VALUE section requiring that if the *asctime()* function is unsuccessful it returns
20558 NULL.
- 20559 **Issue 7**
- 20560 Austin Group Interpretation 1003.1-2001 #053 is applied, marking these functions obsolescent.
- 20561 Austin Group Interpretation 1003.1-2001 #156 is applied.
- 20562 The *asctime_r()* function is moved from the Thread-Safe Functions option to the Base.
- 20563 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0033 [86,429] is applied. +

20564 **NAME**

20565 asin, asinf, asinl — arc sine function

20566 **SYNOPSIS**

```
20567 #include <math.h>
20568 double asin(double x);
20569 float asinf(float x);
20570 long double asinl(long double x);
```

20571 **DESCRIPTION**

20572 CX The functionality described on this reference page is aligned with the ISO C standard. Any
 20573 conflict between the requirements described here and the ISO C standard is unintentional. This
 20574 volume of POSIX.1-2008 defers to the ISO C standard.

20575 These functions shall compute the principal value of the arc sine of their argument x . The value
 20576 of x should be in the range $[-1,1]$.

20577 An application wishing to check for error situations should set *errno* to zero and call
 20578 *feclearexcept*(FE_ALL_EXCEPT) before calling these functions. On return, if *errno* is non-zero or
 20579 *fetestexcept*(FE_INVALID | FE_DIVBYZERO | FE_OVERFLOW | FE_UNDERFLOW) is non-
 20580 zero, an error has occurred.

20581 **RETURN VALUE**

20582 Upon successful completion, these functions shall return the arc sine of x , in the range
 20583 $[-\pi/2, \pi/2]$ radians.

20584 MX For finite values of x not in the range $[-1,1]$, a domain error shall occur, and either a NaN (if
 20585 supported), or an implementation-defined value shall be returned.

20586 MX If x is NaN, a NaN shall be returned.

20587 If x is ± 0 , x shall be returned.

20588 If x is $\pm \text{Inf}$, a domain error shall occur, and a NaN shall be returned. |

20589 If x is subnormal, a range error may occur |

20590 MXX and x should be returned.

20591 MX If x is not returned, *asin()*, *asinf()*, and *asinl()* shall return an implementation-defined value no
 20592 greater in magnitude than DBL_MIN, FLT_MIN, and LDBL_MIN, respectively. +

20593 **ERRORS**

20594 These functions shall fail if:

20595 MX Domain Error The x argument is finite and is not in the range $[-1,1]$, or is $\pm \text{Inf}$.

20596 If the integer expression (*math_errhandling* & MATH_ERRNO) is non-zero,
 20597 then *errno* shall be set to [EDOM]. If the integer expression (*math_errhandling*
 20598 & MATH_ERREXCEPT) is non-zero, then the invalid floating-point exception
 20599 shall be raised.

20600 These functions may fail if:

20601 MX Range Error The value of x is subnormal.

20602 If the integer expression (*math_errhandling* & MATH_ERRNO) is non-zero,
 20603 then *errno* shall be set to [ERANGE]. If the integer expression
 20604 (*math_errhandling* & MATH_ERREXCEPT) is non-zero, then the underflow
 20605 floating-point exception shall be raised.

EXAMPLES

None.

APPLICATION USAGE

On error, the expressions (*math_errhandling* & MATH_ERRNO) and (*math_errhandling* & MATH_ERREXCEPT) are independent of each other, but at least one of them must be non-zero.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

feclearexcept(), *fetestexcept()*, *isnan()*, *sin()*

XBD Section 4.19 (on page 116), <math.h>

CHANGE HISTORY

First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 5

The DESCRIPTION is updated to indicate how an application should check for an error. This text was previously published in the APPLICATION USAGE section.

Issue 6

The *asinf()* and *asinl()* functions are added for alignment with the ISO/IEC 9899:1999 standard.

The DESCRIPTION, RETURN VALUE, ERRORS, and APPLICATION USAGE sections are revised to align with the ISO/IEC 9899:1999 standard.

IEC 60559:1989 standard floating-point extensions over the ISO/IEC 9899:1999 standard are marked.

Issue 7

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0034 [320] and XSH/TC1-2008/0035 + [68] are applied.

20632 **NAME**

20633 asinh, asinhf, asinhl — inverse hyperbolic sine functions

20634 **SYNOPSIS**

```
20635 #include <math.h>
20636 double asinh(double x);
20637 float asinhf(float x);
20638 long double asinhl(long double x);
```

20639 **DESCRIPTION**

20640 **CX** The functionality described on this reference page is aligned with the ISO C standard. Any
 20641 conflict between the requirements described here and the ISO C standard is unintentional. This
 20642 volume of POSIX.1-2008 defers to the ISO C standard.

20643 These functions shall compute the inverse hyperbolic sine of their argument x .

20644 An application wishing to check for error situations should set *errno* to zero and call
 20645 *feclearexcept*(FE_ALL_EXCEPT) before calling these functions. On return, if *errno* is non-zero or
 20646 *fetestexcept*(FE_INVALID | FE_DIVBYZERO | FE_OVERFLOW | FE_UNDERFLOW) is non-
 20647 zero, an error has occurred.

20648 **RETURN VALUE**

20649 Upon successful completion, these functions shall return the inverse hyperbolic sine of their
 20650 argument.

20651 **MX** If x is NaN, a NaN shall be returned.

20652 If x is ± 0 , or $\pm \text{Inf}$, x shall be returned.

20653 If x is subnormal, a range error may occur

20654 **MX** and x should be returned.

20655 **MX** If x is not returned, *asinh*(), *asinhf*(), and *asinhl*() shall return an implementation-defined value
 20656 no greater in magnitude than DBL_MIN, FLT_MIN, and LDBL_MIN, respectively.

20657 **ERRORS**

20658 These functions may fail if:

20659 **MX** **Range Error** The value of x is subnormal.

20660 If the integer expression (*math_errhandling* & MATH_ERRNO) is non-zero,
 20661 then *errno* shall be set to [ERANGE]. If the integer expression
 20662 (*math_errhandling* & MATH_ERREXCEPT) is non-zero, then the underflow
 20663 floating-point exception shall be raised.

20664 **EXAMPLES**

20665 None.

20666 **APPLICATION USAGE**

20667 On error, the expressions (*math_errhandling* & MATH_ERRNO) and (*math_errhandling* &
 20668 MATH_ERREXCEPT) are independent of each other, but at least one of them must be non-zero.

20669 **RATIONALE**

20670 None.

20671 **FUTURE DIRECTIONS**

20672 None.

20673 **SEE ALSO**20674 *feclearexcept()*, *fetestexcept()*, *sinh()*20675 XBD Section 4.19 (on page 116), **<math.h>**20676 **CHANGE HISTORY**

20677 First released in Issue 4, Version 2.

20678 **Issue 5**

20679 Moved from X/OPEN UNIX extension to BASE.

20680 **Issue 6**20681 The *asinh()* function is no longer marked as an extension.20682 The *asinhf()* and *asinhll()* functions are added for alignment with the ISO/IEC 9899:1999
20683 standard.20684 The DESCRIPTION, RETURN VALUE, ERRORS, and APPLICATION USAGE sections are
20685 revised to align with the ISO/IEC 9899:1999 standard.20686 IEC 60559:1989 standard floating-point extensions over the ISO/IEC 9899:1999 standard are
20687 marked.20688 **Issue 7**

20689 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0036 [68] is applied.

+

20690 **NAME**20691 **asinl** — arc sine function20692 **SYNOPSIS**20693 `#include <math.h>`20694 `long double asinl(long double x);`20695 **DESCRIPTION**20696 Refer to *asin()*.

20697 **NAME**

20698 assert — insert program diagnostics

20699 **SYNOPSIS**

20700 #include <assert.h>

20701 void assert(scalar expression);

20702 **DESCRIPTION**

20703 **CX** The functionality described on this reference page is aligned with the ISO C standard. Any
 20704 conflict between the requirements described here and the ISO C standard is unintentional. This
 20705 volume of POSIX.1-2008 defers to the ISO C standard.

20706 The *assert()* macro shall insert diagnostics into programs; it shall expand to a **void** expression.
 20707 When it is executed, if *expression* (which shall have a **scalar** type) is false (that is, compares equal
 20708 to 0), *assert()* shall write information about the particular call that failed on *stderr* and shall call
 20709 *abort()*.

20710 The information written about the call that failed shall include the text of the argument, the
 20711 name of the source file, the source file line number, and the name of the enclosing function; the
 20712 latter are, respectively, the values of the preprocessing macros `__FILE__` and `__LINE__` and of
 20713 the identifier `__func__`.

20714 Forcing a definition of the name `NDEBUG`, either from the compiler command line or with the
 20715 preprocessor control statement **#define** `NDEBUG` ahead of the **#include** `<assert.h>` statement,
 20716 shall stop assertions from being compiled into the program.

20717 **RETURN VALUE**20718 The *assert()* macro shall not return a value.20719 **ERRORS**

20720 No errors are defined.

20721 **EXAMPLES**

20722 None.

20723 **APPLICATION USAGE**

20724 None.

20725 **RATIONALE**

20726 None.

20727 **FUTURE DIRECTIONS**

20728 None.

20729 **SEE ALSO**20730 *abort()*, *stdin*20731 XBD `<assert.h>`20732 **CHANGE HISTORY**

20733 First released in Issue 1. Derived from Issue 1 of the SVID.

20734 **Issue 6**

20735 The prototype for the *expression* argument to *assert()* is changed from **int** to **scalar** for alignment
 20736 with the ISO/IEC 9899:1999 standard.

20737 The DESCRIPTION of *assert()* is updated for alignment with the ISO/IEC 9899:1999 standard.

20738 **NAME**

20739 atan, atanf, atanl — arc tangent function

20740 **SYNOPSIS**

```
20741 #include <math.h>
20742 double atan(double x);
20743 float atanf(float x);
20744 long double atanl(long double x);
```

20745 **DESCRIPTION**

20746 CX The functionality described on this reference page is aligned with the ISO C standard. Any
 20747 conflict between the requirements described here and the ISO C standard is unintentional. This
 20748 volume of POSIX.1-2008 defers to the ISO C standard.

20749 These functions shall compute the principal value of the arc tangent of their argument x .

20750 An application wishing to check for error situations should set *errno* to zero and call
 20751 *feclearexcept*(FE_ALL_EXCEPT) before calling these functions. On return, if *errno* is non-zero or
 20752 *fetestexcept*(FE_INVALID | FE_DIVBYZERO | FE_OVERFLOW | FE_UNDERFLOW) is non-
 20753 zero, an error has occurred.

20754 **RETURN VALUE**

20755 Upon successful completion, these functions shall return the arc tangent of x in the range
 20756 $[-\pi/2, \pi/2]$ radians.

20757 MX If x is NaN, a NaN shall be returned.

20758 If x is ± 0 , x shall be returned.

20759 If x is $\pm \text{Inf}$, $\pm \pi/2$ shall be returned.

20760 If x is subnormal, a range error may occur

20761 MXX and x should be returned.

20762 MX If x is not returned, *atan()*, *atanf()*, and *atanl()* shall return an implementation-defined value no
 20763 greater in magnitude than DBL_MIN, FLT_MIN, and LDBL_MIN, respectively.

20764 **ERRORS**

20765 These functions may fail if:

20766 MX **Range Error** The value of x is subnormal.

20767 If the integer expression (*math_errhandling* & MATH_ERRNO) is non-zero,
 20768 then *errno* shall be set to [ERANGE]. If the integer expression
 20769 (*math_errhandling* & MATH_ERREXCEPT) is non-zero, then the underflow
 20770 floating-point exception shall be raised.

20771 **EXAMPLES**

20772 None.

20773 **APPLICATION USAGE**

20774 On error, the expressions (*math_errhandling* & MATH_ERRNO) and (*math_errhandling* &
 20775 MATH_ERREXCEPT) are independent of each other, but at least one of them must be non-zero.

20776 **RATIONALE**

20777 None.

20778 **FUTURE DIRECTIONS**

20779 None.

20780 **SEE ALSO**20781 *atan2()*, *feclearexcept()*, *fetestexcept()*, *isnan()*, *tan()*20782 XBD [Section 4.19](#) (on page 116), [<math.h>](#)20783 **CHANGE HISTORY**

20784 First released in Issue 1. Derived from Issue 1 of the SVID.

20785 **Issue 5**20786 The DESCRIPTION is updated to indicate how an application should check for an error. This
20787 text was previously published in the APPLICATION USAGE section.20788 **Issue 6**20789 The *atanf()* and *atanl()* functions are added for alignment with the ISO/IEC 9899:1999 standard.20790 The DESCRIPTION, RETURN VALUE, ERRORS, and APPLICATION USAGE sections are
20791 revised to align with the ISO/IEC 9899:1999 standard.20792 IEC 60559:1989 standard floating-point extensions over the ISO/IEC 9899:1999 standard are
20793 marked.20794 **Issue 7**

20795 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0037 [68] is applied.

+

20796 **NAME**

20797 atan2, atan2f, atan2l — arc tangent functions

20798 **SYNOPSIS**

```
20799 #include <math.h>
20800 double atan2(double y, double x);
20801 float atan2f(float y, float x);
20802 long double atan2l(long double y, long double x);
```

20803 **DESCRIPTION**

20804 CX The functionality described on this reference page is aligned with the ISO C standard. Any
 20805 conflict between the requirements described here and the ISO C standard is unintentional. This
 20806 volume of POSIX.1-2008 defers to the ISO C standard.

20807 These functions shall compute the principal value of the arc tangent of y/x , using the signs of
 20808 both arguments to determine the quadrant of the return value.

20809 An application wishing to check for error situations should set *errno* to zero and call
 20810 *feclearexcept*(FE_ALL_EXCEPT) before calling these functions. On return, if *errno* is non-zero or
 20811 *fetestexcept*(FE_INVALID | FE_DIVBYZERO | FE_OVERFLOW | FE_UNDERFLOW) is non-
 20812 zero, an error has occurred.

20813 **RETURN VALUE**

20814 Upon successful completion, these functions shall return the arc tangent of y/x in the range
 20815 $[-\pi, \pi]$ radians.

20816 If y is ± 0 and x is < 0 , $\pm\pi$ shall be returned.

20817 If y is ± 0 and x is > 0 , ± 0 shall be returned.

20818 If y is < 0 and x is ± 0 , $-\pi/2$ shall be returned.

20819 If y is > 0 and x is ± 0 , $\pi/2$ shall be returned.

20820 If x is 0, a pole error shall not occur.

20821 MX If either x or y is NaN, a NaN shall be returned.

20822 If the correct value would cause underflow, a range error may occur, and *atan*(), *atan2f*(), and
 20823 *atan2l*() shall return an implementation-defined value no greater in magnitude than DBL_MIN,
 20824 FLT_MIN, and LDBL_MIN, respectively.

20825 MXX If the IEC 60559 Floating-Point option is supported, y/x should be returned.

20826 MX If y is ± 0 and x is -0 , $\pm\pi$ shall be returned.

20827 If y is ± 0 and x is $+0$, ± 0 shall be returned.

20828 For finite values of $\pm y > 0$, if x is $-\text{Inf}$, $\pm\pi$ shall be returned.

20829 For finite values of $\pm y > 0$, if x is $+\text{Inf}$, ± 0 shall be returned.

20830 For finite values of x , if y is $\pm\text{Inf}$, $\pm\pi/2$ shall be returned.

20831 If y is $\pm\text{Inf}$ and x is $-\text{Inf}$, $\pm 3\pi/4$ shall be returned.

20832 If y is $\pm\text{Inf}$ and x is $+\text{Inf}$, $\pm\pi/4$ shall be returned.

20833 If both arguments are 0, a domain error shall not occur.

ERRORS

These functions may fail if:

MX

Range Error

The result underflows.

If the integer expression *(math_errhandling & MATH_ERRNO)* is non-zero, then *errno* shall be set to [ERANGE]. If the integer expression *(math_errhandling & MATH_ERREXCEPT)* is non-zero, then the underflow floating-point exception shall be raised.

EXAMPLES**Converting Cartesian to Polar Coordinates System**

The function below uses *atan2()* to convert a 2d vector expressed in cartesian coordinates *(x,y)* to the polar coordinates *(rho,theta)*. There are other ways to compute the angle *theta*, using *asin()*, *acos()*, or *atan()*. However, *atan2()* presents here two advantages:

- The angle's quadrant is automatically determined.
- The singular cases *(0,y)* are taken into account.

Finally, this example uses *hypot()* rather than *sqrt()* since it is better for special cases; see *hypot()* for more information.

```
#include <math.h>

void
cartesian_to_polar(const double x, const double y,
                  double *rho, double *theta)
{
    *rho    = hypot (x,y); /* better than sqrt(x*x+y*y) */
    *theta  = atan2 (y,x);
}
```

APPLICATION USAGE

On error, the expressions *(math_errhandling & MATH_ERRNO)* and *(math_errhandling & MATH_ERREXCEPT)* are independent of each other, but at least one of them must be non-zero.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

acos(), *asin()*, *atan()*, *feclearexcept()*, *fetestexcept()*, *hypot()*, *isnan()*, *sqrt()*, *tan()*

XBD Section 4.19 (on page 116), **<math.h>**

CHANGE HISTORY

First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 5

The DESCRIPTION is updated to indicate how an application should check for an error. This text was previously published in the APPLICATION USAGE section.

20874 **Issue 6**

20875 The *atan2f()* and *atan2l()* functions are added for alignment with the ISO/IEC 9899:1999
20876 standard.

20877 The DESCRIPTION, RETURN VALUE, ERRORS, and APPLICATION USAGE sections are
20878 revised to align with the ISO/IEC 9899:1999 standard, and the IEC 60559:1989 standard
20879 floating-point extensions over the ISO/IEC 9899:1999 standard are marked. -

20880 IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/18 is applied, adding to the EXAMPLES
20881 section.

20882 **Issue 7**

20883 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0038 [68,428] is applied. +

20884 **NAME**

20885 atanf — arc tangent function

20886 **SYNOPSIS**

20887 #include <math.h>

20888 float atanf(float x);

20889 **DESCRIPTION**20890 Refer to *atan()*.

20891 **NAME**

20892 atanh, atanhf, atanh1 — inverse hyperbolic tangent functions

20893 **SYNOPSIS**

```
20894       #include <math.h>
20895       double atanh(double x);
20896       float atanhf(float x);
20897       long double atanh1(long double x);
```

20898 **DESCRIPTION**

20899 CX The functionality described on this reference page is aligned with the ISO C standard. Any
 20900 conflict between the requirements described here and the ISO C standard is unintentional. This
 20901 volume of POSIX.1-2008 defers to the ISO C standard.

20902 These functions shall compute the inverse hyperbolic tangent of their argument x .

20903 An application wishing to check for error situations should set *errno* to zero and call
 20904 *feclearexcept*(FE_ALL_EXCEPT) before calling these functions. On return, if *errno* is non-zero or
 20905 *fetestexcept*(FE_INVALID | FE_DIVBYZERO | FE_OVERFLOW | FE_UNDERFLOW) is non-
 20906 zero, an error has occurred.

20907 **RETURN VALUE**

20908 Upon successful completion, these functions shall return the inverse hyperbolic tangent of their
 20909 argument.

20910 If x is ± 1 , a pole error shall occur, and *atanh*(), *atanhf*(), and *atanhl*() shall return the value of the
 20911 macro HUGE_VAL, HUGE_VALF, and HUGE_VALL, respectively, with the same sign as the
 20912 correct value of the function.

20913 MX For finite $|x| > 1$, a domain error shall occur, and either a NaN (if supported), or an
 20914 implementation-defined value shall be returned.

20915 MX If x is NaN, a NaN shall be returned.

20916 If x is ± 0 , x shall be returned.

20917 If x is $\pm \text{Inf}$, a domain error shall occur, and a NaN shall be returned. |

20918 If x is subnormal, a range error may occur |

20919 MXX and x should be returned.

20920 MX If x is not returned, *atanh*(), *atanhf*(), and *atanhl*() shall return an implementation-defined value +
 20921 no greater in magnitude than DBL_MIN, FLT_MIN, and LDBL_MIN, respectively.

20922 **ERRORS**

20923 These functions shall fail if:

20924 MX Domain Error The x argument is finite and not in the range $[-1, 1]$, or is $\pm \text{Inf}$.

20925 If the integer expression (*math_errhandling* & MATH_ERRNO) is non-zero,
 20926 then *errno* shall be set to [EDOM]. If the integer expression (*math_errhandling*
 20927 & MATH_ERREXCEPT) is non-zero, then the invalid floating-point exception
 20928 shall be raised.

20929 Pole Error The x argument is ± 1 .

20930 If the integer expression (*math_errhandling* & MATH_ERRNO) is non-zero,
 20931 then *errno* shall be set to [ERANGE]. If the integer expression
 20932 (*math_errhandling* & MATH_ERREXCEPT) is non-zero, then the divide-by-zero
 20933 floating-point exception shall be raised.

20934 These functions may fail if:

20935	MX	Range Error	The value of x is subnormal.
20936			If the integer expression $(\text{math\_errhandling} \ \& \ \text{MATH\_ERRNO})$ is non-zero,
20937			then <i>errno</i> shall be set to [ERANGE]. If the integer expression
20938			$(\text{math\_errhandling} \ \& \ \text{MATH\_ERREXCEPT})$ is non-zero, then the underflow
20939			floating-point exception shall be raised.

20940 **EXAMPLES**

20941 None.

20942 **APPLICATION USAGE**

20943 On error, the expressions $(\text{math\_errhandling} \ \& \ \text{MATH\_ERRNO})$ and $(\text{math\_errhandling} \ \& \ \text{MATH\_ERREXCEPT})$ are independent of each other, but at least one of them must be non-zero.

20945 **RATIONALE**

20946 None.

20947 **FUTURE DIRECTIONS**

20948 None.

20949 **SEE ALSO**

20950 *feclearexcept()*, *fetestexcept()*, *tanh()*

20951 XBD [Section 4.19](#) (on page 116), `<math.h>`

20952 **CHANGE HISTORY**

20953 First released in Issue 4, Version 2.

20954 **Issue 5**

20955 Moved from X/OPEN UNIX extension to BASE.

20956 **Issue 6**

20957 The *atanh()* function is no longer marked as an extension.

20958 The *atanhf()* and *atanhl()* functions are added for alignment with the ISO/IEC 9899:1999 standard.

20960 The DESCRIPTION, RETURN VALUE, ERRORS, and APPLICATION USAGE sections are revised to align with the ISO/IEC 9899:1999 standard.

20962 IEC 60559:1989 standard floating-point extensions over the ISO/IEC 9899:1999 standard are marked.

20964 **Issue 7**

20965 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0039 [320] and XSH/TC1-2008/0040 +
20966 [680] are applied.

20967 **NAME**

20968 atanl — arc tangent function

20969 **SYNOPSIS**

20970 #include <math.h>

20971 long double atanl(long double x);

20972 **DESCRIPTION**20973 Refer to *atan()*.

20974 **NAME**

20975 atexit — register a function to run at process termination

20976 **SYNOPSIS**

20977 #include <stdlib.h>

20978 int atexit(void (*func)(void));

20979 **DESCRIPTION**

20980 CX The functionality described on this reference page is aligned with the ISO C standard. Any
 20981 conflict between the requirements described here and the ISO C standard is unintentional. This
 20982 volume of POSIX.1-2008 defers to the ISO C standard.

20983 The *atexit()* function shall register the function pointed to by *func*, to be called without
 20984 arguments at normal program termination. At normal program termination, all functions
 20985 registered by the *atexit()* function shall be called, in the reverse order of their registration, except
 20986 that a function is called after any previously registered functions that had already been called at
 20987 the time it was registered. Normal termination occurs either by a call to *exit()* or a return from
 20988 *main()*.

20989 At least 32 functions can be registered with *atexit()*.

20990 CX After a successful call to any of the *exec* functions, any functions previously registered by *atexit()*
 20991 shall no longer be registered.

20992 **RETURN VALUE**20993 Upon successful completion, *atexit()* shall return 0; otherwise, it shall return a non-zero value.20994 **ERRORS**

20995 No errors are defined.

20996 **EXAMPLES**

20997 None.

20998 **APPLICATION USAGE**20999 The functions registered by a call to *atexit()* must return to ensure that all registered functions
21000 are called.

21001 The application should call *sysconf()* to obtain the value of {ATEXIT_MAX}, the number of
 21002 functions that can be registered. There is no way for an application to tell how many functions
 21003 have already been registered with *atexit()*.

21004 Since the behavior is undefined if the *exit()* function is called more than once, portable
 21005 applications calling *atexit()* must ensure that the *exit()* function is not called at normal process
 21006 termination when all functions registered by the *atexit()* function are called.

21007 All functions registered by the *atexit()* function are called at normal process termination, which
 21008 occurs by a call to the *exit()* function or a return from *main()* or on the last thread termination,
 21009 when the behavior is as if the implementation called *exit()* with a zero argument at thread
 21010 termination time.

21011 If, at normal process termination, a function registered by the *atexit()* function is called and a
 21012 portable application needs to stop further *exit()* processing, it must call the *_exit()* function or
 21013 the *_Exit()* function or one of the functions which cause abnormal process termination.

21014 **RATIONALE**

21015 None.

21016 FUTURE DIRECTIONS

21017 None.

21018 SEE ALSO

21019 *exec*, *exit()*, *sysconf()*

21020 XBD <stdlib.h>

21021 CHANGE HISTORY

21022 First released in Issue 4. Derived from the ANSI C standard.

21023 Issue 6

21024 Extensions beyond the ISO C standard are marked.

21025 The DESCRIPTION is updated for alignment with the ISO/IEC 9899:1999 standard.

21026 IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/19 is applied, adding further clarification
21027 to the APPLICATION USAGE section.

21028 **NAME**

21029 atof — convert a string to a double-precision number

21030 **SYNOPSIS**

21031 #include <stdlib.h>

21032 double atof(const char *str);

21033 **DESCRIPTION**

21034 CX The functionality described on this reference page is aligned with the ISO C standard. Any
21035 conflict between the requirements described here and the ISO C standard is unintentional. This
21036 volume of POSIX.1-2008 defers to the ISO C standard.

21037 The call *atof(str)* shall be equivalent to:

21038 strtod(str, (char **) NULL),

21039 except that the handling of errors may differ. If the value cannot be represented, the behavior is
21040 undefined.

21041 **RETURN VALUE**21042 The *atof()* function shall return the converted value if the value can be represented.21043 **ERRORS**

21044 No errors are defined.

21045 **EXAMPLES**

21046 None.

21047 **APPLICATION USAGE**

21048 The *atof()* function is subsumed by *strtod()* but is retained because it is used extensively in
21049 existing code. If the number is not known to be in range, *strtod()* should be used because *atof()*
21050 is not required to perform any error checking.

21051 **RATIONALE**

21052 None.

21053 **FUTURE DIRECTIONS**

21054 None.

21055 **SEE ALSO**21056 *strtod()*

21057 XBD <stdlib.h>

21058 **CHANGE HISTORY**

21059 First released in Issue 1. Derived from Issue 1 of the SVID.

21060 **NAME**

21061 atoi — convert a string to an integer

21062 **SYNOPSIS**

21063 #include <stdlib.h>

21064 int atoi(const char *str);

21065 **DESCRIPTION**

21066 CX The functionality described on this reference page is aligned with the ISO C standard. Any
 21067 conflict between the requirements described here and the ISO C standard is unintentional. This
 21068 volume of POSIX.1-2008 defers to the ISO C standard.

21069 The call *atoi(str)* shall be equivalent to:

21070 (int) strtol(str, (char **)NULL, 10)

21071 except that the handling of errors may differ. If the value cannot be represented, the behavior is
 21072 undefined.

21073 **RETURN VALUE**21074 The *atoi()* function shall return the converted value if the value can be represented.21075 **ERRORS**

21076 No errors are defined.

21077 **EXAMPLES**21078 **Converting an Argument**

21079 The following example checks for proper usage of the program. If there is an argument and the
 21080 decimal conversion of this argument (obtained using *atoi()*) is greater than 0, then the program
 21081 has a valid number of minutes to wait for an event.

```
21082       #include <stdlib.h>
21083       #include <stdio.h>
21084       ...
21085       int minutes_to_event;
21086       ...
21087       if (argc < 2 || ((minutes_to_event = atoi (argv[1]))) <= 0) {
21088           fprintf(stderr, "Usage: %s minutes\n", argv[0]); exit(1);
21089       }
21090       ...
```

21091 **APPLICATION USAGE**

21092 The *atoi()* function is subsumed by *strtol()* but is retained because it is used extensively in
 21093 existing code. If the number is not known to be in range, *strtol()* should be used because *atoi()* is
 21094 not required to perform any error checking.

21095 **RATIONALE**

21096 None.

21097 **FUTURE DIRECTIONS**

21098 None.

21099 **SEE ALSO**21100 *strtol()*

21101 XBD <stdlib.h>

21102	CHANGE HISTORY
21103	First released in Issue 1. Derived from Issue 1 of the SVID.

21104 **NAME**

21105 `atoi, atoll` — convert a string to a long integer

21106 **SYNOPSIS**

21107 `#include <stdlib.h>`
 21108 `long atoi(const char *str);`
 21109 `long long atoll(const char *nptr);`

21110 **DESCRIPTION**

21111 CX The functionality described on this reference page is aligned with the ISO C standard. Any
 21112 conflict between the requirements described here and the ISO C standard is unintentional. This
 21113 volume of POSIX.1-2008 defers to the ISO C standard.

21114 The call `atoi(str)` shall be equivalent to:

21115 `strtol(str, (char **)NULL, 10)`

21116 The call `atoll(nptr)` shall be equivalent to:

21117 `strtoll(nptr, (char **)NULL, 10)`

21118 except that the handling of errors may differ. If the value cannot be represented, the behavior is
 21119 undefined.

21120 **RETURN VALUE**

21121 These functions shall return the converted value if the value can be represented.

21122 **ERRORS**

21123 No errors are defined.

21124 **EXAMPLES**

21125 None.

21126 **APPLICATION USAGE**

21127 The `atoi()` function is subsumed by `strtol()` but is retained because it is used extensively in
 21128 existing code. If the number is not known to be in range, `strtol()` should be used because `atoi()` is
 21129 not required to perform any error checking.

21130 **RATIONALE**

21131 None.

21132 **FUTURE DIRECTIONS**

21133 None.

21134 **SEE ALSO**

21135 [*strtol\(\)*](#)
 21136 XBD [*<stdlib.h>*](#)

21137 **CHANGE HISTORY**

21138 First released in Issue 1. Derived from Issue 1 of the SVID.

21139 **Issue 6**

21140 The `atoll()` function is added for alignment with the ISO/IEC 9899:1999 standard.

21141 **Issue 7**

21142 SD5-XSH-ERN-61 is applied, correcting the DESCRIPTION of `atoll()`.

21143 **NAME**21144 **basename** — return the last component of a pathname21145 **SYNOPSIS**

```
21146 XSI      #include <libgen.h>
21147          char *basename(char *path);
```

21148 **DESCRIPTION**

21149 The *basename()* function shall take the pathname pointed to by *path* and return a pointer to the
 21150 final component of the pathname, deleting any trailing '/' characters.

21151 If the string pointed to by *path* consists entirely of the '/' character, *basename()* shall return a
 21152 pointer to the string "/". If the string pointed to by *path* is exactly "///", it is implementation-
 21153 defined whether '/' or "///" is returned.

21154 If *path* is a null pointer or points to an empty string, *basename()* shall return a pointer to the
 21155 string ".".

21156 The *basename()* function may modify the string pointed to by *path*, and may return a pointer to
 21157 internal storage. The returned pointer might be invalidated or the storage might be overwritten
 21158 by a subsequent call to *basename()*.

21159 The *basename()* function need not be thread-safe.

21160 **RETURN VALUE**

21161 The *basename()* function shall return a pointer to the final component of *path*.

21162 **ERRORS**

21163 No errors are defined.

21164 **EXAMPLES**21165 **Using basename()**

21166 The following program fragment returns a pointer to the value *lib*, which is the base name of
 21167 */usr/lib*.

```
21168     #include <libgen.h>
21169     ...
21170     char *name = "/usr/lib";
21171     char *base;

21172     base = basename(name);
21173     ...
```

21174 **Sample Input and Output Strings for basename()**

21175 In the following table, the input string is the value pointed to by *path*, and the output string is
 21176 the return value of the *basename()* function.

Input String	Output String
"/usr/lib"	"lib"
"/usr/"	"usr"
"/"	"/"
"/" / "	"/"
"/usr//lib//"	"lib"

21183 APPLICATION USAGE

21184 None.

21185 RATIONALE

21186 None.

21187 FUTURE DIRECTIONS

21188 None.

21189 SEE ALSO

21190 *dirname()*

21191 XBD <libgen.h>

21192 XCU *basename*

21193 CHANGE HISTORY

21194 First released in Issue 4, Version 2.

21195 Issue 5

21196 Moved from X/OPEN UNIX extension to BASE.

21197 Normative text previously in the APPLICATION USAGE section is moved to the
21198 DESCRIPTION.

21199 A note indicating that this function need not be reentrant is added to the DESCRIPTION.

21200 Issue 6

21201 In the DESCRIPTION, the note about reentrancy is expanded to cover thread-safety.

21202 IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/20 is applied, changing the
21203 DESCRIPTION to make it clear that the string referenced is the string pointed to by *path*.

21204 Issue 7

21205 Austin Group Interpretation 1003.1-2001 #156 is applied.

21206 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0041 [75] is applied.

+

21207 NAME

21208 bind — bind a name to a socket

21209 SYNOPSIS

```
21210       #include <sys/socket.h>
21211       int bind(int socket, const struct sockaddr *address,
21212               socklen_t address_len);
```

21213 DESCRIPTION

21214 The *bind()* function shall assign a local socket address *address* to a socket identified by descriptor
 21215 *socket* that has no local socket address assigned. Sockets created with the *socket()* function are
 21216 initially unnamed; they are identified only by their address family.

21217 The *bind()* function takes the following arguments:

21218	<i>socket</i>	Specifies the file descriptor of the socket to be bound.
21219	<i>address</i>	Points to a sockaddr structure containing the address to be bound to the
21220		socket. The length and format of the address depend on the address family of
21221		the socket.
21222	<i>address_len</i>	Specifies the length of the sockaddr structure pointed to by the <i>address</i>
21223		argument.

21224 The socket specified by *socket* may require the process to have appropriate privileges to use the
 21225 *bind()* function.

21226 If the address family of the socket is AF_UNIX and the pathname in *address* names a symbolic +
 21227 link, *bind()* shall fail and set *errno* to [EADDRINUSE]. +

21228 If the socket address cannot be assigned immediately and O_NONBLOCK is set for the file
 21229 descriptor for the socket, *bind()* shall fail and set *errno* to [EINPROGRESS], but the assignment
 21230 request shall not be aborted, and the assignment shall be completed asynchronously. Subsequent
 21231 calls to *bind()* for the same socket, before the assignment is completed, shall fail and set *errno* to
 21232 [EALREADY].

21233 When the assignment has been performed asynchronously, *pselect()*, *select()*, and *poll()* shall
 21234 indicate that the file descriptor for the socket is ready for reading and writing.

21235 RETURN VALUE

21236 Upon successful completion, *bind()* shall return 0; otherwise, -1 shall be returned and *errno* set
 21237 to indicate the error.

21238 ERRORS

21239	The <i>bind()</i> function shall fail if:	
21240	[EADDRINUSE]	The specified address is already in use.
21241	[EADDRNOTAVAIL]	
21242		The specified address is not available from the local machine.
21243	[EAFNOSUPPORT]	
21244		The specified address is not a valid address for the address family of the
21245		specified socket.
21246	[EALREADY]	An assignment request is already in progress for the specified socket.
21247	[EBADF]	The <i>socket</i> argument is not a valid file descriptor.

21248	[EINPROGRESS]	O_NONBLOCK is set for the file descriptor for the socket and the assignment cannot be immediately performed; the assignment shall be performed asynchronously.	
21249			
21250			
21251	[EINVAL]	The socket is already bound to an address, and the protocol does not support binding to a new address; or the socket has been shut down.	
21252			
21253	[ENOBUFS]	Insufficient resources were available to complete the call.	
21254	[ENOTSOCK]	The <i>socket</i> argument does not refer to a socket.	
21255	[EOPNOTSUPP]	The socket type of the specified socket does not support binding to an address.	
21256		If the address family of the socket is AF_UNIX, then <i>bind()</i> shall fail if:	
21257	[EACCES]	A component of the path prefix denies search permission, or the requested name requires writing in a directory with a mode that denies write permission.	
21258			
21259			
21260	[EDESTADDRREQ] or [EISDIR]		
21261		The <i>address</i> argument is a null pointer.	
21262	[EIO]	An I/O error occurred.	
21263	[ELOOP]	A loop exists in symbolic links encountered during resolution of the pathname in <i>address</i> .	
21264			
21265	[ENAMETOOLONG]		
21266		The length of a component of a pathname is longer than {NAME_MAX}.	
21267	[ENOENT]	A component of the path prefix of the pathname in <i>address</i> does not name an existing file or the pathname is an empty string.	
21268			
21269	[ENOENT] or [ENOTDIR]		
21270		The pathname in <i>address</i> contains at least one non- <i><slash></i> character and ends with one or more trailing <i><slash></i> characters. If the pathname names an existing file, an [ENOENT] error shall not occur.	
21271			
21272			
21273	[ENOTDIR]	A component of the path prefix of the pathname in <i>address</i> names an existing file that is neither a directory nor a symbolic link to a directory, or the pathname in <i>address</i> contains at least one non- <i><slash></i> character and ends with one or more trailing <i><slash></i> characters and the last pathname component names an existing file that is neither a directory nor a symbolic link to a directory.	
21274			
21275			
21276			
21277			
21278			
21279	[EROFS]	The name would reside on a read-only file system.	
21280		The <i>bind()</i> function may fail if:	
21281	[EACCES]	The specified address is protected and the current user does not have permission to bind to it.	
21282			
21283	[EINVAL]	The <i>address_len</i> argument is not a valid length for the address family.	
21284	[EISCONN]	The socket is already connected.	
21285	[ELOOP]	More than {SYMLOOP_MAX} symbolic links were encountered during resolution of the pathname in <i>address</i> .	
21286			
21287	[ENAMETOOLONG]		
21288		The length of a pathname exceeds {PATH_MAX}, or pathname resolution of a symbolic link produced an intermediate result with a length that exceeds	
21289			

21290 {PATH_MAX}.

21291 EXAMPLES

21292 The following code segment shows how to create a socket and bind it to a name in the AF_UNIX
21293 domain.

```
21294 #define MY_SOCKET_PATH "/somepath"
21295 int sfd;
21296 struct sockaddr_un my_addr;
21297 sfd = socket(AF_UNIX, SOCK_STREAM, 0);
21298 if (sfd == -1)
21299     /* Handle error */;
21300 memset(&my_addr, '\0', sizeof(struct sockaddr_un));
21301     /* Clear structure */
21302 my_addr.sun_family = AF_UNIX;
21303 strncpy(my_addr.sun_path, MY_SOCKET_PATH, sizeof(my_addr.sun_path) - 1);
21304 if (bind(sfd, (struct sockaddr *) &my_addr,
21305         sizeof(struct sockaddr_un)) == -1)
21306     /* Handle error */;
```

21307 APPLICATION USAGE

21308 An application program can retrieve the assigned socket name with the *getsockname()* function.

21309 RATIONALE

21310 None.

21311 FUTURE DIRECTIONS

21312 None.

21313 SEE ALSO

21314 *connect()*, *getsockname()*, *listen()*, *socket()*

21315 XBD <sys/socket.h>

21316 CHANGE HISTORY

21317 First released in Issue 6. Derived from the XNS, Issue 5.2 specification.

21318 Issue 7

21319 Austin Group Interpretation 1003.1-2001 #044 is applied, changing the “may fail” [ENOBUFFS]
21320 error to become a “shall fail” error.

21321 Austin Group Interpretation 1003.1-2001 #143 is applied.

21322 SD5-XSH-ERN-185 is applied.

21323 An example is added.

21324 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0042 [146], XSH/TC1-2008/0043 [146], +
21325 and XSH/TC1-2008/0044 [324] are applied.

NAME

bsearch — binary search a sorted table

SYNOPSIS

```
#include <stdlib.h>
```

```
void *bsearch(const void *key, const void *base, size_t nel,
             size_t width, int (*compar)(const void *, const void *));
```

DESCRIPTION

CX The functionality described on this reference page is aligned with the ISO C standard. Any conflict between the requirements described here and the ISO C standard is unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

The `bsearch()` function shall search an array of *nel* objects, the initial element of which is pointed to by *base*, for an element that matches the object pointed to by *key*. The size of each element in the array is specified by *width*. If the *nel* argument has the value zero, the comparison function pointed to by *compar* shall not be called and no match shall be found.

The comparison function pointed to by *compar* shall be called with two arguments that point to the *key* object and to an array element, in that order.

The application shall ensure that the comparison function pointed to by *compar* does not alter the contents of the array. The implementation may reorder elements of the array between calls to the comparison function, but shall not alter the contents of any individual element.

The implementation shall ensure that the first argument is always a pointer to the key.

When the same objects (consisting of width bytes, irrespective of their current positions in the array) are passed more than once to the comparison function, the results shall be consistent with one another. That is, the same object shall always compare the same way with the key.

The application shall ensure that the function returns an integer less than, equal to, or greater than 0 if the *key* object is considered, respectively, to be less than, to match, or to be greater than the array element. The application shall ensure that the array consists of all the elements that compare less than, all the elements that compare equal to, and all the elements that compare greater than the *key* object, in that order.

RETURN VALUE

The `bsearch()` function shall return a pointer to a matching member of the array, or a null pointer if no match is found. If two or more members compare equal, which member is returned is unspecified.

ERRORS

No errors are defined.

EXAMPLES

The example below searches a table containing pointers to nodes consisting of a string and its length. The table is ordered alphabetically on the string in the node pointed to by each entry.

The code fragment below reads in strings and either finds the corresponding node and prints out the string and its length, or prints an error message.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
```

```
#define TABSIZE 1000
```

[illegible]

```

21370     char *string;
21371     int length;
21372 };
21373 struct node table[TABSIZE];    /* Table to be searched. */
21374     .
21375     .
21376     .
21377 {
21378     struct node *node_ptr, node;
21379     /* Routine to compare 2 nodes. */
21380     int node_compare(const void *, const void *);
21381     .
21382     .
21383     .
21384     while (scanf("%ms", &node.string) != EOF) {
21385         node_ptr = (struct node *)bsearch((void *)(&node),
21386             (void *)table, TABSIZE,
21387             sizeof(struct node), node_compare);
21388         if (node_ptr != NULL) {
21389             (void)printf("string = %20s, length = %d\n",
21390                 node_ptr->string, node_ptr->length);
21391         } else {
21392             (void)printf("not found: %s\n", node.string);
21393         }
21394         free(node.string);
21395     }
21396 }
21397 /*
21398     This routine compares two nodes based on an
21399     alphabetical ordering of the string field.
21400 */
21401 int
21402 node_compare(const void *node1, const void *node2)
21403 {
21404     return strcoll(((const struct node *)node1)->string,
21405         ((const struct node *)node2)->string);
21406 }

```

APPLICATION USAGE

The pointers to the key and the element at the base of the table should be of type pointer-to-element.

The comparison function need not compare every byte, so arbitrary data may be contained in the elements in addition to the values being compared.

In practice, the array is usually sorted according to the comparison function.

RATIONALE

The requirement that the second argument (hereafter referred to as *p*) to the comparison function is a pointer to an element of the array implies that for every call all of the following expressions are non-zero:

```

((char *)p - (char *(base) % width == 0
(char *)p >= (char *)base

```

21419 (char *)p < (char *)base + nel * width

21420 **FUTURE DIRECTIONS**

21421 None.

21422 **SEE ALSO**

21423 *hcreate()*, *lsearch()*, *qsort()*, *tdelete()*

21424 XBD <stdlib.h>

21425 **CHANGE HISTORY**

21426 First released in Issue 1. Derived from Issue 1 of the SVID.

21427 **Issue 6**

21428 The normative text is updated to avoid use of the term “must” for application requirements.

21429 IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/11 is applied, adding to the
21430 DESCRIPTION the last sentence of the first non-shaded paragraph, and the following three
21431 paragraphs. The RATIONALE section is also updated. These changes are for alignment with the
21432 ISO C standard.

21433 **Issue 7**

21434 The EXAMPLES section is revised.

21435 **NAME**

21436 btowc — single byte to wide character conversion

21437 **SYNOPSIS**

21438 #include <stdio.h>

21439 #include <wchar.h>

21440 wint_t btowc(int c);

21441 **DESCRIPTION**

21442 CX The functionality described on this reference page is aligned with the ISO C standard. Any
21443 conflict between the requirements described here and the ISO C standard is unintentional. This
21444 volume of POSIX.1-2008 defers to the ISO C standard.

21445 The *btowc()* function shall determine whether *c* constitutes a valid (one-byte) character in the
21446 initial shift state.

21447 The behavior of this function shall be affected by the *LC_CTYPE* category of the current locale.

21448 **RETURN VALUE**

21449 The *btowc()* function shall return WEOF if *c* has the value EOF or if (**unsigned char**) *c* does not
21450 constitute a valid (one-byte) character in the initial shift state. Otherwise, it shall return the
21451 wide-character representation of that character.

21452 **ERRORS**

21453 No errors are defined.

21454 **EXAMPLES**

21455 None.

21456 **APPLICATION USAGE**

21457 None.

21458 **RATIONALE**

21459 None.

21460 **FUTURE DIRECTIONS**

21461 None.

21462 **SEE ALSO**21463 *wctob()*

21464 XBD <stdio.h>, <wchar.h>

21465 **CHANGE HISTORY**

21466 First released in Issue 5. Included for alignment with ISO/IEC 9899:1990/Amendment 1:1995
21467 (E).

21468 **NAME**

21469 cabs, cabsf, cabsl — return a complex absolute value

21470 **SYNOPSIS**

```
21471       #include <complex.h>
21472       double cabs(double complex z);
21473       float cabsf(float complex z);
21474       long double cabsl(long double complex z);
```

21475 **DESCRIPTION**

21476 CX The functionality described on this reference page is aligned with the ISO C standard. Any
21477 conflict between the requirements described here and the ISO C standard is unintentional. This
21478 volume of POSIX.1-2008 defers to the ISO C standard.

21479 These functions shall compute the complex absolute value (also called norm, modulus, or
21480 magnitude) of *z*.

21481 **RETURN VALUE**

21482 These functions shall return the complex absolute value.

21483 **ERRORS**

21484 No errors are defined.

21485 **EXAMPLES**

21486 None.

21487 **APPLICATION USAGE**

21488 None.

21489 **RATIONALE**

21490 None.

21491 **FUTURE DIRECTIONS**

21492 None.

21493 **SEE ALSO**21494 XBD [<complex.h>](#)21495 **CHANGE HISTORY**

21496 First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

21497 **NAME**

21498 cacos, cacosf, cacosl — complex arc cosine functions

21499 **SYNOPSIS**

21500 #include <complex.h>

21501 double complex cacos(double complex z);

21502 float complex cacosf(float complex z);

21503 long double complex cacosl(long double complex z);

21504 **DESCRIPTION**

21505 CX The functionality described on this reference page is aligned with the ISO C standard. Any
 21506 conflict between the requirements described here and the ISO C standard is unintentional. This
 21507 volume of POSIX.1-2008 defers to the ISO C standard.

21508 These functions shall compute the complex arc cosine of z , with branch cuts outside the interval
 21509 $[-1, +1]$ along the real axis.

21510 **RETURN VALUE**

21511 These functions shall return the complex arc cosine value, in the range of a strip mathematically
 21512 unbounded along the imaginary axis and in the interval $[0, \pi]$ along the real axis.

21513 **ERRORS**

21514 No errors are defined.

21515 **EXAMPLES**

21516 None.

21517 **APPLICATION USAGE**

21518 None.

21519 **RATIONALE**

21520 None.

21521 **FUTURE DIRECTIONS**

21522 None.

21523 **SEE ALSO**

21524 ccos()

21525 XBD <complex.h>

21526 **CHANGE HISTORY**

21527 First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

21528 **NAME**

21529 cacosh, cacoshf, cacoshl — complex arc hyperbolic cosine functions

21530 **SYNOPSIS**

```
21531 #include <complex.h>
21532 double complex cacosh(double complex z);
21533 float complex cacoshf(float complex z);
21534 long double complex cacoshl(long double complex z);
```

21535 **DESCRIPTION**

21536 CX The functionality described on this reference page is aligned with the ISO C standard. Any
21537 conflict between the requirements described here and the ISO C standard is unintentional. This
21538 volume of POSIX.1-2008 defers to the ISO C standard.

21539 These functions shall compute the complex arc hyperbolic cosine of z , with a branch cut at
21540 values less than 1 along the real axis.

21541 **RETURN VALUE**

21542 These functions shall return the complex arc hyperbolic cosine value, in the range of a half-strip
21543 of non-negative values along the real axis and in the interval $[-i\pi, +i\pi]$ along the imaginary axis.

21544 **ERRORS**

21545 No errors are defined.

21546 **EXAMPLES**

21547 None.

21548 **APPLICATION USAGE**

21549 None.

21550 **RATIONALE**

21551 None.

21552 **FUTURE DIRECTIONS**

21553 None.

21554 **SEE ALSO**21555 [ccosh\(\)](#)21556 XBD [<complex.h>](#)21557 **CHANGE HISTORY**

21558 First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

21559 **NAME**

21560 cacosl — complex arc cosine functions

21561 **SYNOPSIS**

21562 #include <complex.h>

21563 long double complex cacosl(long double complex z);

21564 **DESCRIPTION**21565 Refer to *cacos()*.

21566 NAME

21567 `calloc` — a memory allocator

21568 SYNOPSIS

21569 `#include <stdlib.h>`

21570 `void *calloc(size_t nelem, size_t elsize);`

21571 DESCRIPTION

21572 CX The functionality described on this reference page is aligned with the ISO C standard. Any
21573 conflict between the requirements described here and the ISO C standard is unintentional. This
21574 volume of POSIX.1-2008 defers to the ISO C standard.

21575 The `calloc()` function shall allocate unused space for an array of *nelem* elements each of whose
21576 size in bytes is *elsize*. The space shall be initialized to all bits 0.

21577 The order and contiguity of storage allocated by successive calls to `calloc()` is unspecified. The
21578 pointer returned if the allocation succeeds shall be suitably aligned so that it may be assigned to
21579 a pointer to any type of object and then used to access such an object or an array of such objects
21580 in the space allocated (until the space is explicitly freed or reallocated). Each such allocation shall
21581 yield a pointer to an object disjoint from any other object. The pointer returned shall point to the
21582 start (lowest byte address) of the allocated space. If the space cannot be allocated, a null pointer
21583 shall be returned. If the size of the space requested is 0, the behavior is implementation-defined:
21584 the value returned shall be either a null pointer or a unique pointer.

21585 RETURN VALUE

21586 Upon successful completion with both *nelem* and *elsize* non-zero, `calloc()` shall return a pointer to
21587 the allocated space. If either *nelem* or *elsize* is 0, then either a null pointer or a unique pointer
21588 value that can be successfully passed to `free()` shall be returned. Otherwise, it shall return a null
21589 CX pointer and set *errno* to indicate the error.

21590 ERRORS

21591 The `calloc()` function shall fail if:

21592 CX [ENOMEM] Insufficient memory is available.

21593 EXAMPLES

21594 None.

21595 APPLICATION USAGE

21596 There is now no requirement for the implementation to support the inclusion of `<malloc.h>`.

21597 RATIONALE

21598 None.

21599 FUTURE DIRECTIONS

21600 None.

21601 SEE ALSO

21602 [*free\(\)*](#), [*malloc\(\)*](#), [*realloc\(\)*](#)

21603 XBD `<stdlib.h>`

21604 CHANGE HISTORY

21605 First released in Issue 1. Derived from Issue 1 of the SVID.

21606 Issue 6

21607 Extensions beyond the ISO C standard are marked.

21608
21609

21610
21611

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The setting of *errno* and the [ENOMEM] error condition are mandatory if an insufficient memory condition occurs.

21612 **NAME**

21613 carg, cargf, cargl — complex argument functions

21614 **SYNOPSIS**

```
21615       #include <complex.h>

21616       double carg(double complex z);
21617       float cargf(float complex z);
21618       long double cargl(long double complex z);
```

21619 **DESCRIPTION**

21620 CX The functionality described on this reference page is aligned with the ISO C standard. Any
21621 conflict between the requirements described here and the ISO C standard is unintentional. This
21622 volume of POSIX.1-2008 defers to the ISO C standard.

21623 These functions shall compute the argument (also called phase angle) of z , with a branch cut
21624 along the negative real axis.

21625 **RETURN VALUE**21626 These functions shall return the value of the argument in the interval $[-\pi, +\pi]$.21627 **ERRORS**

21628 No errors are defined.

21629 **EXAMPLES**

21630 None.

21631 **APPLICATION USAGE**

21632 None.

21633 **RATIONALE**

21634 None.

21635 **FUTURE DIRECTIONS**

21636 None.

21637 **SEE ALSO**21638 *cimag()*, *conj()*, *cproj()*21639 XBD **<complex.h>**21640 **CHANGE HISTORY**

21641 First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

21642 **NAME**

21643 casin, casinf, casinl — complex arc sine functions

21644 **SYNOPSIS**

```
21645       #include <complex.h>

21646       double complex casin(double complex z);
21647       float complex casinf(float complex z);
21648       long double complex casinl(long double complex z);
```

21649 **DESCRIPTION**

21650 CX The functionality described on this reference page is aligned with the ISO C standard. Any
 21651 conflict between the requirements described here and the ISO C standard is unintentional. This
 21652 volume of POSIX.1-2008 defers to the ISO C standard.

21653 These functions shall compute the complex arc sine of z , with branch cuts outside the interval
 21654 $[-1, +1]$ along the real axis.

21655 **RETURN VALUE**

21656 These functions shall return the complex arc sine value, in the range of a strip mathematically
 21657 unbounded along the imaginary axis and in the interval $[-\pi/2, +\pi/2]$ along the real axis.

21658 **ERRORS**

21659 No errors are defined.

21660 **EXAMPLES**

21661 None.

21662 **APPLICATION USAGE**

21663 None.

21664 **RATIONALE**

21665 None.

21666 **FUTURE DIRECTIONS**

21667 None.

21668 **SEE ALSO**21669 `csin()`21670 XBD `<complex.h>`21671 **CHANGE HISTORY**

21672 First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

21673 **NAME**

21674 c sinh, c sinh f, c sinh l — complex arc hyperbolic sine functions

21675 **SYNOPSIS**

21676 #include <complex.h>

21677 double complex c sinh(double complex z);

21678 float complex c sinh f(float complex z);

21679 long double complex c sinh l(long double complex z);

21680 **DESCRIPTION**

21681 CX The functionality described on this reference page is aligned with the ISO C standard. Any
21682 conflict between the requirements described here and the ISO C standard is unintentional. This
21683 volume of POSIX.1-2008 defers to the ISO C standard.

21684 These functions shall compute the complex arc hyperbolic sine of z , with branch cuts outside the
21685 interval $[-i, +i]$ along the imaginary axis.

21686 **RETURN VALUE**

21687 These functions shall return the complex arc hyperbolic sine value, in the range of a strip
21688 mathematically unbounded along the real axis and in the interval $[-i\pi/2, +i\pi/2]$ along the
21689 imaginary axis.

21690 **ERRORS**

21691 No errors are defined.

21692 **EXAMPLES**

21693 None.

21694 **APPLICATION USAGE**

21695 None.

21696 **RATIONALE**

21697 None.

21698 **FUTURE DIRECTIONS**

21699 None.

21700 **SEE ALSO**

21701 c sinh()

21702 XBD <complex.h>

21703 **CHANGE HISTORY**

21704 First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

21705 **NAME**

21706 casinl — complex arc sine functions

21707 **SYNOPSIS**

21708 #include <complex.h>

21709 long double complex casinl(long double complex z);

21710 **DESCRIPTION**21711 Refer to *casin()*.

21712 **NAME**

21713 catan, catanf, catanl — complex arc tangent functions

21714 **SYNOPSIS**

```
21715       #include <complex.h>

21716       double complex catan(double complex z);
21717       float complex catanf(float complex z);
21718       long double complex catanl(long double complex z);
```

21719 **DESCRIPTION**

21720 CX The functionality described on this reference page is aligned with the ISO C standard. Any
21721 conflict between the requirements described here and the ISO C standard is unintentional. This
21722 volume of POSIX.1-2008 defers to the ISO C standard.

21723 These functions shall compute the complex arc tangent of z , with branch cuts outside the
21724 interval $[-i, +i]$ along the imaginary axis.

21725 **RETURN VALUE**

21726 These functions shall return the complex arc tangent value, in the range of a strip
21727 mathematically unbounded along the imaginary axis and in the interval $[-\pi/2, +\pi/2]$ along the
21728 real axis.

21729 **ERRORS**

21730 No errors are defined.

21731 **EXAMPLES**

21732 None.

21733 **APPLICATION USAGE**

21734 None.

21735 **RATIONALE**

21736 None.

21737 **FUTURE DIRECTIONS**

21738 None.

21739 **SEE ALSO**

21740 ctan()

21741 XBD <complex.h>

21742 **CHANGE HISTORY**

21743 First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

21744 **NAME**

21745 catanh, catanhf, catanhl — complex arc hyperbolic tangent functions

21746 **SYNOPSIS**

21747 #include <complex.h>

21748 double complex catanh(double complex z);

21749 float complex catanhf(float complex z);

21750 long double complex catanhl(long double complex z);

21751 **DESCRIPTION**

21752 CX The functionality described on this reference page is aligned with the ISO C standard. Any
21753 conflict between the requirements described here and the ISO C standard is unintentional. This
21754 volume of POSIX.1-2008 defers to the ISO C standard.

21755 These functions shall compute the complex arc hyperbolic tangent of z , with branch cuts outside
21756 the interval $[-1, +1]$ along the real axis.

21757 **RETURN VALUE**

21758 These functions shall return the complex arc hyperbolic tangent value, in the range of a strip
21759 mathematically unbounded along the real axis and in the interval $[-i\pi/2, +i\pi/2]$ along the
21760 imaginary axis.

21761 **ERRORS**

21762 No errors are defined.

21763 **EXAMPLES**

21764 None.

21765 **APPLICATION USAGE**

21766 None.

21767 **RATIONALE**

21768 None.

21769 **FUTURE DIRECTIONS**

21770 None.

21771 **SEE ALSO**

21772 ctanh()

21773 XBD <complex.h>

21774 **CHANGE HISTORY**

21775 First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

21776 **NAME**

21777 catanl — complex arc tangent functions

21778 **SYNOPSIS**

21779 #include <complex.h>

21780 long double complex catanl(long double complex z);

21781 **DESCRIPTION**21782 Refer to *catan()*.

21783 **NAME**

21784 catclose — close a message catalog descriptor

21785 **SYNOPSIS**

21786 #include <nl_types.h>

21787 int catclose(nl_catd catd);

21788 **DESCRIPTION**

21789 The *catclose()* function shall close the message catalog identified by *catd*. If a file descriptor is
21790 used to implement the type **nl_catd**, that file descriptor shall be closed.

21791 **RETURN VALUE**

21792 Upon successful completion, *catclose()* shall return 0; otherwise, -1 shall be returned, and *errno*
21793 set to indicate the error.

21794 **ERRORS**21795 The *catclose()* function may fail if:

21796 [EBADF] The catalog descriptor is not valid.

21797 [EINTR] The *catclose()* function was interrupted by a signal.21798 **EXAMPLES**

21799 None.

21800 **APPLICATION USAGE**

21801 None.

21802 **RATIONALE**

21803 None.

21804 **FUTURE DIRECTIONS**

21805 None.

21806 **SEE ALSO**21807 *catgets()*, *catopen()*

21808 XBD <nl_types.h>

21809 **CHANGE HISTORY**

21810 First released in Issue 2.

21811 **Issue 7**21812 The *catclose()* function is moved from the XSI option to the Base.

21813 **NAME**

21814 catgets — read a program message

21815 **SYNOPSIS**

21816 #include <nl_types.h>

21817 char *catgets(nl_catd catd, int set_id, int msg_id, const char *s);

21818 **DESCRIPTION**

21819 The *catgets()* function shall attempt to read message *msg_id*, in set *set_id*, from the message
 21820 catalog identified by *catd*. The *catd* argument is a message catalog descriptor returned from an
 21821 earlier call to *catopen()*. The results are undefined if *catd* is not a value returned by *catopen()* for
 21822 a message catalog still open in the process. The *s* argument points to a default message string
 21823 which shall be returned by *catgets()* if it cannot retrieve the identified message.

21824 The *catgets()* function need not be thread-safe.21825 **RETURN VALUE**

21826 If the identified message is retrieved successfully, *catgets()* shall return a pointer to an internal
 21827 buffer area containing the null-terminated message string. If the call is unsuccessful for any
 21828 reason, *s* shall be returned and *errno* shall be set to indicate the error.

21829 **ERRORS**21830 The *catgets()* function shall fail if:

21831 [EINTR] The read operation was terminated due to the receipt of a signal, and no data
 21832 was transferred.

21833 [ENOMSG] The message identified by *set_id* and *msg_id* is not in the message catalog.21834 The *catgets()* function may fail if:21835 [EBADF] The *catd* argument is not a valid message catalog descriptor open for reading.

21836 [EBADMSG] The message identified by *set_id* and *msg_id* in the specified message catalog
 21837 did not satisfy implementation-defined security criteria.

21838 [EINVAL] The message catalog identified by *catd* is corrupted.21839 **EXAMPLES**

21840 None.

21841 **APPLICATION USAGE**

21842 None.

21843 **RATIONALE**

21844 None.

21845 **FUTURE DIRECTIONS**

21846 None.

21847 **SEE ALSO**21848 *catclose()*, *catopen()*

21849 XBD <nl_types.h>

21850 **CHANGE HISTORY**

21851 First released in Issue 2.

21852 **Issue 5**

21853 A note indicating that this function need not be reentrant is added to the DESCRIPTION.

21854 **Issue 6**

21855 In the DESCRIPTION, the note about reentrancy is expanded to cover thread-safety.

21856 **Issue 7**

21857 Austin Group Interpretation 1003.1-2001 #044 is applied, changing the “may fail” [EINTR] and
21858 [ENOMSG] errors to become “shall fail” errors, updating the RETURN VALUE section, and
21859 updating the DESCRIPTION to note that: “The results are undefined if *catd* is not a value
21860 returned by *catopen()* for a message catalog still open in the process.

21861 Austin Group Interpretation 1003.1-2001 #148 is applied, adding

21862 The *catgets()* function is moved from the XSI option to the Base.

21863 **NAME**

21864 catopen — open a message catalog

21865 **SYNOPSIS**

21866 #include <nl_types.h>

21867 nl_catd catopen(const char *name, int oflag);

21868 **DESCRIPTION**

21869 The *catopen()* function shall open a message catalog and return a message catalog descriptor.
 21870 The *name* argument specifies the name of the message catalog to be opened. If *name* contains a
 21871 ' / ', then *name* specifies a complete name for the message catalog. Otherwise, the environment
 21872 variable *NLSPATH* is used with *name* substituted for the %N conversion specification (see XBD
 21873 Chapter 8, on page 173). If *NLSPATH* exists in the environment when the process starts, then if
 21874 the process has appropriate privileges, the behavior of *catopen()* is undefined. If *NLSPATH* does
 21875 not exist in the environment, or if a message catalog cannot be found in any of the components
 21876 specified by *NLSPATH*, then an implementation-defined default path shall be used. This default
 21877 may be affected by the setting of *LC_MESSAGES* if the value of *oflag* is *NL_CAT_LOCALE*, or
 21878 the *LANG* environment variable if *oflag* is 0.

21879 A message catalog descriptor shall remain valid in a process until that process closes it, or a
 21880 successful call to one of the *exec* functions. A change in the setting of the *LC_MESSAGES*
 21881 category may invalidate existing open catalogs.

21882 If a file descriptor is used to implement message catalog descriptors, the *FD_CLOEXEC* flag
 21883 shall be set; see <fcntl.h>.

21884 If the value of the *oflag* argument is 0, the *LANG* environment variable is used to locate the
 21885 catalog without regard to the *LC_MESSAGES* category. If the *oflag* argument is
 21886 *NL_CAT_LOCALE*, the *LC_MESSAGES* category is used to locate the message catalog (see XBD
 21887 Section 8.2, on page 174).

21888 **RETURN VALUE**

21889 Upon successful completion, *catopen()* shall return a message catalog descriptor for use on
 21890 subsequent calls to *catgets()* and *catclose()*. Otherwise, *catopen()* shall return (*nl_catd*) -1 and set
 21891 *errno* to indicate the error.

21892 **ERRORS**21893 The *catopen()* function may fail if:

21894 [EACCES] Search permission is denied for the component of the path prefix of the
 21895 message catalog or read permission is denied for the message catalog.

21896 [EMFILE] All file descriptors available to the process are currently open.

21897 [ENAMETOOLONG]
 21898 The length of a component of a pathname is longer than {NAME_MAX}.

21899 [ENAMETOOLONG]
 21900 The length of a pathname exceeds {PATH_MAX}, or pathname resolution of a
 21901 symbolic link produced an intermediate result with a length that exceeds
 21902 {PATH_MAX}.

21903 [ENFILE] Too many files are currently open in the system.

21904 [ENOENT] The message catalog does not exist or the *name* argument points to an empty
 21905 string.

21906	[ENOMEM]	Insufficient storage space is available.	
21907	[ENOTDIR]	A component of the path prefix of the message catalog names an existing file	
21908		that is neither a directory nor a symbolic link to a directory, or the pathname	
21909		of the message catalog contains at least one non- <code><slash></code> character and ends	
21910		with one or more trailing <code><slash></code> characters and the last pathname	
21911		component names an existing file that is neither a directory nor a symbolic	
21912		link to a directory.	

EXAMPLES

21913
21914 None.

APPLICATION USAGE

21916 Some implementations of *catopen()* use *malloc()* to allocate space for internal buffer areas. The
21917 *catopen()* function may fail if there is insufficient storage space available to accommodate these
21918 buffers.

21919 Conforming applications must assume that message catalog descriptors are not valid after a call
21920 to one of the *exec* functions.

21921 Application developers should be aware that guidelines for the location of message catalogs
21922 have not yet been developed. Therefore they should take care to avoid conflicting with catalogs
21923 used by other applications and the standard utilities.

RATIONALE

21924
21925 None.

FUTURE DIRECTIONS

21926
21927 None.

SEE ALSO

21928 *catclose()*, *catgets()*
21929
21930 XBD Chapter 8 (on page 173), *<fcntl.h>*, *<nl_types.h>*,

CHANGE HISTORY

21931
21932 First released in Issue 2.

Issue 7

21934 Austin Group Interpretation 1003.1-2001 #143 is applied.
21935 SD5-XBD-ERN-4 is applied, changing the definition of the [EMFILE] error.
21936 The *catopen()* function is moved from the XSI option to the Base.
21937 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0045 [324] is applied.

+

21938 **NAME**

21939 cbrt, cbrtf, cbrtl — cube root functions

21940 **SYNOPSIS**

```
21941       #include <math.h>
21942       double cbrt(double x);
21943       float cbrtf(float x);
21944       long double cbrtl(long double x);
```

21945 **DESCRIPTION**

21946 **CX** The functionality described on this reference page is aligned with the ISO C standard. Any
 21947 conflict between the requirements described here and the ISO C standard is unintentional. This
 21948 volume of POSIX.1-2008 defers to the ISO C standard.

21949 These functions shall compute the real cube root of their argument *x*.21950 **RETURN VALUE**21951 Upon successful completion, these functions shall return the cube root of *x*.21952 **MX** If *x* is NaN, a NaN shall be returned.21953 If *x* is ± 0 or $\pm \text{Inf}$, *x* shall be returned.21954 **ERRORS**

21955 No errors are defined.

21956 **EXAMPLES**

21957 None.

21958 **APPLICATION USAGE**

21959 None.

21960 **RATIONALE**

21961 For some applications, a true cube root function, which returns negative results for negative
 21962 arguments, is more appropriate than *pow(x, 1.0/3.0)*, which returns a NaN for *x* less than 0.

21963 **FUTURE DIRECTIONS**

21964 None.

21965 **SEE ALSO**21966 XBD [<math.h>](#)21967 **CHANGE HISTORY**

21968 First released in Issue 4, Version 2.

21969 **Issue 5**

21970 Moved from X/OPEN UNIX extension to BASE.

21971 **Issue 6**21972 The *cbrt()* function is no longer marked as an extension.21973 The *cbrtf()* and *cbrtl()* functions are added for alignment with the ISO/IEC 9899:1999 standard.

21974 The DESCRIPTION, RETURN VALUE, ERRORS, and APPLICATION USAGE sections are
 21975 revised to align with the ISO/IEC 9899:1999 standard.

21976 IEC 60559:1989 standard floating-point extensions over the ISO/IEC 9899:1999 standard are
 21977 marked.

21978 **NAME**

21979 ccos, ccosf, ccosl — complex cosine functions

21980 **SYNOPSIS**

21981 #include <complex.h>

21982 double complex ccos(double complex z);

21983 float complex ccosf(float complex z);

21984 long double complex ccosl(long double complex z);

21985 **DESCRIPTION**

21986 CX The functionality described on this reference page is aligned with the ISO C standard. Any
21987 conflict between the requirements described here and the ISO C standard is unintentional. This
21988 volume of POSIX.1-2008 defers to the ISO C standard.

21989 These functions shall compute the complex cosine of z.

21990 **RETURN VALUE**

21991 These functions shall return the complex cosine value.

21992 **ERRORS**

21993 No errors are defined.

21994 **EXAMPLES**

21995 None.

21996 **APPLICATION USAGE**

21997 None.

21998 **RATIONALE**

21999 None.

22000 **FUTURE DIRECTIONS**

22001 None.

22002 **SEE ALSO**22003 [ccos\(\)](#)22004 XBD [<complex.h>](#)22005 **CHANGE HISTORY**

22006 First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

22007 **NAME**

22008 ccosh, ccoshf, ccoshl — complex hyperbolic cosine functions

22009 **SYNOPSIS**

22010 #include <complex.h>

22011 double complex ccosh(double complex z);

22012 float complex ccoshf(float complex z);

22013 long double complex ccoshl(long double complex z);

22014 **DESCRIPTION**

22015 CX The functionality described on this reference page is aligned with the ISO C standard. Any
22016 conflict between the requirements described here and the ISO C standard is unintentional. This
22017 volume of POSIX.1-2008 defers to the ISO C standard.

22018 These functions shall compute the complex hyperbolic cosine of z.

22019 **RETURN VALUE**

22020 These functions shall return the complex hyperbolic cosine value.

22021 **ERRORS**

22022 No errors are defined.

22023 **EXAMPLES**

22024 None.

22025 **APPLICATION USAGE**

22026 None.

22027 **RATIONALE**

22028 None.

22029 **FUTURE DIRECTIONS**

22030 None.

22031 **SEE ALSO**

22032 cacosh()

22033 XBD <complex.h>

22034 **CHANGE HISTORY**

22035 First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

22036 **NAME**

22037 ccosl — complex cosine functions

22038 **SYNOPSIS**

22039 #include <complex.h>

22040 long double complex ccosl(long double complex z);

22041 **DESCRIPTION**22042 Refer to *ccos()*.

22043 **NAME**

22044 ceil, ceilf, ceill — ceiling value function

22045 **SYNOPSIS**

```
22046       #include <math.h>
22047       double ceil(double x);
22048       float ceilf(float x);
22049       long double ceill(long double x);
```

22050 **DESCRIPTION**

22051 CX The functionality described on this reference page is aligned with the ISO C standard. Any
 22052 conflict between the requirements described here and the ISO C standard is unintentional. This
 22053 volume of POSIX.1-2008 defers to the ISO C standard.

22054 These functions shall compute the smallest integral value not less than *x*. -

22055 **RETURN VALUE**

22056 MX The result shall have the same sign as *x*. +

22057 Upon successful completion, *ceil()*, *ceilf()*, and *ceill()* shall return the smallest integral value not
 22058 less than *x*, expressed as a type **double**, **float**, or **long double**, respectively.

22059 MX If *x* is NaN, a NaN shall be returned.

22060 If *x* is ± 0 or $\pm \text{Inf}$, *x* shall be returned. -

22061 **ERRORS**

22062 No errors are defined. |

22063 **EXAMPLES**

22064 None.

22065 **APPLICATION USAGE**

22066 The integral value returned by these functions need not be expressible as an **intmax_t**. The
 22067 return value should be tested before assigning it to an integer type to avoid the undefined
 22068 results of an integer overflow.

22069 These functions may raise the inexact floating-point exception if the result differs in value from |
 22070 the argument.

22071 **RATIONALE**

22072 None.

22073 **FUTURE DIRECTIONS**

22074 None.

22075 **SEE ALSO**

22076 *feclearexcept()*, *fetestexcept()*, *floor()*, *isnan()*

22077 XBD Section 4.19 (on page 116), **<math.h>**

22078 **CHANGE HISTORY**

22079 First released in Issue 1. Derived from Issue 1 of the SVID.

22080 **Issue 5**

22081 The DESCRIPTION is updated to indicate how an application should check for an error. This
 22082 text was previously published in the APPLICATION USAGE section.

Issue 6

22083 The *ceilf()* and *ceil()* functions are added for alignment with the ISO/IEC 9899:1999 standard.

22084
22085 The DESCRIPTION, RETURN VALUE, ERRORS, and APPLICATION USAGE sections are
22086 revised to align with the ISO/IEC 9899:1999 standard.

22087 IEC 60559:1989 standard floating-point extensions over the ISO/IEC 9899:1999 standard are
22088 marked.

Issue 7

22089 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0046 [346] is applied.

+

22091 **NAME**

22092 cexp, cexpf, cexpl — complex exponential functions

22093 **SYNOPSIS**

22094 #include <complex.h>

22095 double complex cexp(double complex z);

22096 float complex cexpf(float complex z);

22097 long double complex cexpl(long double complex z);

22098 **DESCRIPTION**

22099 CX The functionality described on this reference page is aligned with the ISO C standard. Any
 22100 conflict between the requirements described here and the ISO C standard is unintentional. This
 22101 volume of POSIX.1-2008 defers to the ISO C standard.

22102 These functions shall compute the complex exponent of z , defined as e^z .22103 **RETURN VALUE**22104 These functions shall return the complex exponential value of z .22105 **ERRORS**

22106 No errors are defined.

22107 **EXAMPLES**

22108 None.

22109 **APPLICATION USAGE**

22110 None.

22111 **RATIONALE**

22112 None.

22113 **FUTURE DIRECTIONS**

22114 None.

22115 **SEE ALSO**22116 [clog\(\)](#)22117 XBD [<complex.h>](#)22118 **CHANGE HISTORY**

22119 First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

22120 **NAME**

22121 cfgetispeed — get input baud rate

22122 **SYNOPSIS**

22123 #include <termios.h>

22124 speed_t cfgetispeed(const struct termios *termios_p);

22125 **DESCRIPTION**

22126 The *cfgetispeed()* function shall extract the input baud rate from the **termios** structure to which
 22127 the *termios_p* argument points.

22128 This function shall return exactly the value in the **termios** data structure, without interpretation.

22129 **RETURN VALUE**

22130 Upon successful completion, *cfgetispeed()* shall return a value of type **speed_t** representing the
 22131 input baud rate.

22132 **ERRORS**

22133 No errors are defined.

22134 **EXAMPLES**

22135 None.

22136 **APPLICATION USAGE**

22137 None.

22138 **RATIONALE**

22139 The term “baud” is used historically here, but is not technically correct. This is properly “bits per
 22140 second”, which may not be the same as baud. However, the term is used because of the
 22141 historical usage and understanding.

22142 The *cfgetispeed()*, *cfsetispeed()*, *cfsetospeed()*, and *cfsetispeed()* functions do not take arguments as
 22143 numbers, but rather as symbolic names. There are two reasons for this:

- 22144 1. Historically, numbers were not used because of the way the rate was stored in the data
 22145 structure. This is retained even though a function is now used.
- 22146 2. More importantly, only a limited set of possible rates is at all portable, and this constrains
 22147 the application to that set.

22148 There is nothing to prevent an implementation accepting as an extension a number (such as 126),
 22149 and since the encoding of the Bxxx symbols is not specified, this can be done to avoid
 22150 introducing ambiguity.

22151 Setting the input baud rate to zero was a mechanism to allow for split baud rates. Clarifications
 22152 in this volume of POSIX.1-2008 have made it possible to determine whether split rates are
 22153 supported and to support them without having to treat zero as a special case. Since this
 22154 functionality is also confusing, it has been declared obsolescent. The 0 argument referred to is
 22155 the literal constant 0, not the symbolic constant B0. This volume of POSIX.1-2008 does not
 22156 preclude B0 from being defined as the value 0; in fact, implementations would likely benefit
 22157 from the two being equivalent. This volume of POSIX.1-2008 does not fully specify whether the
 22158 previous *cfsetispeed()* value is retained after a *tcgetattr()* as the actual value or as zero. Therefore,
 22159 conforming applications should always set both the input speed and output speed when setting
 22160 either.

22161 In historical implementations, the baud rate information is traditionally kept in **c_cflag**.
 22162 Applications should be written to presume that this might be the case (and thus not blindly copy
 22163 **c_cflag**), but not to rely on it in case it is in some other field of the structure. Setting the **c_cflag**
 22164 field absolutely after setting a baud rate is a non-portable action because of this. In general, the

22165 unused parts of the flag fields might be used by the implementation and should not be blindly
22166 copied from the descriptions of one terminal device to another.

22167 **FUTURE DIRECTIONS**

22168 None.

22169 **SEE ALSO**

22170 [*cfgetospeed\(\)*](#), [*cfsetispeed\(\)*](#), [*cfsetospeed\(\)*](#), [*tcgetattr\(\)*](#)

22171 XBD [Chapter 11](#) (on page 199), [**<termios.h>**](#)

22172 **CHANGE HISTORY**

22173 First released in Issue 3. Included for alignment with the POSIX.1-1988 standard.

22174 **NAME**

22175 cfgetospeed — get output baud rate

22176 **SYNOPSIS**

22177 #include <termios.h>

22178 speed_t cfgetospeed(const struct termios *termios_p);

22179 **DESCRIPTION**22180 The *cfgetospeed()* function shall extract the output baud rate from the **termios** structure to which
22181 the *termios_p* argument points.22182 This function shall return exactly the value in the **termios** data structure, without interpretation.22183 **RETURN VALUE**22184 Upon successful completion, *cfgetospeed()* shall return a value of type **speed_t** representing the
22185 output baud rate.22186 **ERRORS**

22187 No errors are defined.

22188 **EXAMPLES**

22189 None.

22190 **APPLICATION USAGE**

22191 None.

22192 **RATIONALE**22193 Refer to *cfgetispeed()*.22194 **FUTURE DIRECTIONS**

22195 None.

22196 **SEE ALSO**22197 *cfgetispeed()*, *cfsetispeed()*, *cfsetospeed()*, *tcgetattr()*

22198 XBD Chapter 11 (on page 199), <termios.h>

22199 **CHANGE HISTORY**

22200 First released in Issue 3. Included for alignment with the POSIX.1-1988 standard.

22201 **NAME**

22202 cfsetispeed — set input baud rate

22203 **SYNOPSIS**

22204 #include <termios.h>

22205 int cfsetispeed(struct termios *termios_p, speed_t speed);

22206 **DESCRIPTION**22207 The *cfsetispeed()* function shall set the input baud rate stored in the structure pointed to by
22208 *termios_p* to *speed*.22209 There shall be no effect on the baud rates set in the hardware until a subsequent successful call
22210 to *tcsetattr()* with the same **termios** structure. Similarly, errors resulting from attempts to set
22211 baud rates not supported by the terminal device need not be detected until the *tcsetattr()*
22212 function is called.22213 **RETURN VALUE**22214 Upon successful completion, *cfsetispeed()* shall return 0; otherwise, -1 shall be returned, and
22215 *errno* may be set to indicate the error.22216 **ERRORS**22217 The *cfsetispeed()* function may fail if:22218 [EINVAL] The *speed* value is not a valid baud rate.22219 [EINVAL] The value of *speed* is outside the range of possible speed values as specified in
22220 <termios.h>.22221 **EXAMPLES**

22222 None.

22223 **APPLICATION USAGE**

22224 None.

22225 **RATIONALE**22226 Refer to *cfgetispeed()*.22227 **FUTURE DIRECTIONS**

22228 None.

22229 **SEE ALSO**22230 *cfgetispeed()*, *cfgetospeed()*, *cfsetospeed()*, *tcsetattr()*

22231 XBD Chapter 11 (on page 199), <termios.h>

22232 **CHANGE HISTORY**

22233 First released in Issue 3. Included for alignment with the POSIX.1-1988 standard.

22234 **Issue 6**22235 The following new requirements on POSIX implementations derive from alignment with the
22236 Single UNIX Specification:

- 22237
- The optional setting of *errno* and the [EINVAL] error conditions are added.

22238 **NAME**

22239 cfsetospeed — set output baud rate

22240 **SYNOPSIS**

22241 #include <termios.h>

22242 int cfsetospeed(struct termios *termios_p, speed_t speed);

22243 **DESCRIPTION**22244 The *cfsetospeed()* function shall set the output baud rate stored in the structure pointed to by
22245 *termios_p* to *speed*.22246 There shall be no effect on the baud rates set in the hardware until a subsequent successful call
22247 to *tcsetattr()* with the same **termios** structure. Similarly, errors resulting from attempts to set
22248 baud rates not supported by the terminal device need not be detected until the *tcsetattr()*
22249 function is called.22250 **RETURN VALUE**22251 Upon successful completion, *cfsetospeed()* shall return 0; otherwise, it shall return -1 and *errno*
22252 may be set to indicate the error.22253 **ERRORS**22254 The *cfsetospeed()* function may fail if:22255 [EINVAL] The *speed* value is not a valid baud rate.22256 [EINVAL] The value of *speed* is outside the range of possible speed values as specified in
22257 <termios.h>.22258 **EXAMPLES**

22259 None.

22260 **APPLICATION USAGE**

22261 None.

22262 **RATIONALE**22263 Refer to *cfgetispeed()*.22264 **FUTURE DIRECTIONS**

22265 None.

22266 **SEE ALSO**22267 *cfgetispeed()*, *cfgetospeed()*, *cfsetispeed()*, *tcsetattr()*

22268 XBD Chapter 11 (on page 199), <termios.h>

22269 **CHANGE HISTORY**

22270 First released in Issue 3. Included for alignment with the POSIX.1-1988 standard.

22271 **Issue 6**22272 The following new requirements on POSIX implementations derive from alignment with the
22273 Single UNIX Specification:

- 22274
- The optional setting of *errno* and the [EINVAL] error conditions are added.

22275 **NAME**

22276 chdir — change working directory

22277 **SYNOPSIS**

22278 #include <unistd.h>

22279 int chdir(const char *path);

22280 **DESCRIPTION**

22281 The *chdir()* function shall cause the directory named by the pathname pointed to by the *path*
 22282 argument to become the current working directory; that is, the starting point for path searches
 22283 for pathnames not beginning with '/ '.

22284 **RETURN VALUE**

22285 Upon successful completion, 0 shall be returned. Otherwise, -1 shall be returned, the current
 22286 working directory shall remain unchanged, and *errno* shall be set to indicate the error.

22287 **ERRORS**22288 The *chdir()* function shall fail if:

22289 [EACCES] Search permission is denied for any component of the pathname.

22290 [ELOOP] A loop exists in symbolic links encountered during resolution of the *path*
 22291 argument.

22292 [ENAMETOOLONG]

22293 The length of a component of a pathname is longer than {NAME_MAX}.

22294 [ENOENT] A component of *path* does not name an existing directory or *path* is an empty
 22295 string.

22296 [ENOTDIR] A component of the pathname names an existing file that is neither a directory
 22297 nor a symbolic link to a directory.

22298 The *chdir()* function may fail if:

22299 [ELOOP] More than {SYMLOOP_MAX} symbolic links were encountered during
 22300 resolution of the *path* argument.

22301 [ENAMETOOLONG]

22302 The length of a pathname exceeds {PATH_MAX}, or pathname resolution of a
 22303 symbolic link produced an intermediate result with a length that exceeds
 22304 {PATH_MAX}.

22305 **EXAMPLES**22306 **Changing the Current Working Directory**

22307 The following example makes the value pointed to by **directory**, */tmp*, the current working
 22308 directory.

22309 #include <unistd.h>

22310 ...

22311 char *directory = "/tmp";

22312 int ret;

22313 ret = chdir (directory);

22314 **APPLICATION USAGE**

22315 None.

22316 **RATIONALE**22317 The *chdir()* function only affects the working directory of the current process.22318 **FUTURE DIRECTIONS**

22319 None.

22320 **SEE ALSO**22321 *getcwd()*

22322 XBD <unistd.h>

22323 **CHANGE HISTORY**

22324 First released in Issue 1. Derived from Issue 1 of the SVID.

22325 **Issue 6**

22326 The APPLICATION USAGE section is added.

22327 The following new requirements on POSIX implementations derive from alignment with the
22328 Single UNIX Specification:

- 22329 • The [ELOOP] mandatory error condition is added.
- 22330 • A second [ENAMETOOLONG] is added as an optional error condition.

22331 The following changes were made to align with the IEEE P1003.1a draft standard:

- 22332 • The [ELOOP] optional error condition is added.

22333 **Issue 7**

22334 Austin Group Interpretation 1003.1-2001 #143 is applied.

22335 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0047 [324] is applied.

+

22336 **NAME**

22337 chmod, fchmodat — change mode of a file relative to directory file descriptor

22338 **SYNOPSIS**

22339 #include <sys/stat.h>

22340 int chmod(const char *path, mode_t mode);

22341 int fchmodat(int fd, const char *path, mode_t mode, int flag);

22342 **DESCRIPTION**

22343 XSI The *chmod()* function shall change S_ISUID, S_ISGID, S_ISVTX, and the file permission bits of
 22344 the file named by the pathname pointed to by the *path* argument to the corresponding bits in the
 22345 *mode* argument. The application shall ensure that the effective user ID of the process matches the
 22346 owner of the file or the process has appropriate privileges in order to do this.

22347 XSI S_ISUID, S_ISGID, S_ISVTX, and the file permission bits are described in <sys/stat.h>.

22348 If the calling process does not have appropriate privileges, and if the group ID of the file does
 22349 not match the effective group ID or one of the supplementary group IDs and if the file is a
 22350 regular file, bit S_ISGID (set-group-ID on execution) in the file's mode shall be cleared upon
 22351 successful return from *chmod()*.

22352 Additional implementation-defined restrictions may cause the S_ISUID and S_ISGID bits in
 22353 *mode* to be ignored.

22354 Upon successful completion, *chmod()* shall mark for update the last file status change timestamp
 22355 of the file.

22356 The *fchmodat()* function shall be equivalent to the *chmod()* function except in the case where *path*
 22357 specifies a relative path. In this case the file to be changed is determined relative to the directory
 22358 associated with the file descriptor *fd* instead of the current working directory. If the file
 22359 descriptor was opened without O_SEARCH, the function shall check whether directory searches
 22360 are permitted using the current permissions of the directory underlying the file descriptor. If the
 22361 file descriptor was opened with O_SEARCH, the function shall not perform the check.

22362 Values for *flag* are constructed by a bitwise-inclusive OR of flags from the following list, defined
 22363 in <fcntl.h>:

22364 AT_SYMLINK_NOFOLLOW

22365 If *path* names a symbolic link, then the mode of the symbolic link is changed.

22366 If *fchmodat()* is passed the special value AT_FDCWD in the *fd* parameter, the current working
 22367 directory shall be used. If also *flag* is zero, the behavior shall be identical to a call to *chmod()*.

22368 **RETURN VALUE**

22369 Upon successful completion, these functions shall return 0. Otherwise, these functions shall
 22370 return -1 and set *errno* to indicate the error. If -1 is returned, no change to the file mode occurs.

22371 **ERRORS**

22372 These functions shall fail if:

22373 [EACCES] Search permission is denied on a component of the path prefix.

22374 [ELOOP] A loop exists in symbolic links encountered during resolution of the *path*
 22375 argument.

22376 [ENAMETOOLONG]

22377 The length of a component of a pathname is longer than {NAME_MAX}.

22378	[ENOENT]	A component of <i>path</i> does not name an existing file or <i>path</i> is an empty string.	
22379	[ENOTDIR]	A component of the <i>path</i> prefix names an existing file that is neither a directory nor a symbolic link to a directory, or the <i>path</i> argument contains at least one non- <code><slash></code> character and ends with one or more trailing <code><slash></code> characters and the last pathname component names an existing file that is neither a directory nor a symbolic link to a directory.	
22380			
22381			
22382			
22383			
22384	[EPERM]	The effective user ID does not match the owner of the file and the process does not have appropriate privileges.	
22385			
22386	[EROFS]	The named file resides on a read-only file system.	
22387	The <i>fchmodat()</i> function shall fail if:		
22388	[EACCES]	<i>fd</i> was not opened with <code>O_SEARCH</code> and the permissions of the directory underlying <i>fd</i> do not permit directory searches.	
22389			
22390	[EBADF]	The <i>path</i> argument does not specify an absolute path and the <i>fd</i> argument is neither <code>AT_FDCWD</code> nor a valid file descriptor open for reading or searching.	
22391			
22392	[ENOTDIR]	The <i>path</i> argument is not an absolute path and <i>fd</i> is a file descriptor associated with a non-directory file.	+
22393			
22394	These functions may fail if:		
22395	[EINTR]	A signal was caught during execution of the function.	
22396	[EINVAL]	The value of the <i>mode</i> argument is invalid.	
22397	[ELOOP]	More than <code>{SYMLOOP_MAX}</code> symbolic links were encountered during resolution of the <i>path</i> argument.	
22398			
22399	[ENAMETOOLONG]	The length of a pathname exceeds <code>{PATH_MAX}</code> , or pathname resolution of a symbolic link produced an intermediate result with a length that exceeds <code>{PATH_MAX}</code> .	
22400			
22401			
22402			
22403	The <i>fchmodat()</i> function may fail if:		
22404	[EINVAL]	The value of the <i>flag</i> argument is invalid.	-
22405	[EOPNOTSUPP]	The <code>AT_SYMLINK_NOFOLLOW</code> bit is set in the <i>flag</i> argument, <i>path</i> names a symbolic link, and the system does not support changing the mode of a symbolic link.	
22406			
22407			

EXAMPLES

Setting Read Permissions for User, Group, and Others

The following example sets read permissions for the owner, group, and others.

```
#include <sys/stat.h>

const char *path;
...
chmod(path, S_IRUSR | S_IRGRP | S_IROTH);
```

Setting Read, Write, and Execute Permissions for the Owner Only

The following example sets read, write, and execute permissions for the owner, and no permissions for group and others.

```
#include <sys/stat.h>

const char *path;
...
chmod(path, S_IRWXU);
```

Setting Different Permissions for Owner, Group, and Other

The following example sets owner permissions for CHANGEFILE to read, write, and execute, group permissions to read and execute, and other permissions to read.

```
#include <sys/stat.h>

#define CHANGEFILE "/etc/myfile"
...
chmod(CHANGEFILE, S_IRWXU|S_IRGRP|S_IXGRP|S_IROTH);
```

Setting and Checking File Permissions

The following example sets the file permission bits for a file named **/home/cnd/mod1**, then calls the *stat()* function to verify the permissions.

```
#include <sys/types.h>
#include <sys/stat.h>

int status;
struct stat buffer
...
chmod("home/cnd/mod1", S_IRWXU|S_IRWXG|S_IROTH|S_IWOTH);
status = stat("home/cnd/mod1", &buffer);
```

APPLICATION USAGE

In order to ensure that the *S_ISUID* and *S_ISGID* bits are set, an application requiring this should use *stat()* after a successful *chmod()* to verify this.

Any file descriptors currently open by any process on the file could possibly become invalid if the mode of the file is changed to a value which would deny access to that process. One situation where this could occur is on a stateless file system. This behavior will not occur in a conforming environment.

RATIONALE

This volume of POSIX.1-2008 specifies that the *S_ISGID* bit is cleared by *chmod()* on a regular file under certain conditions. This is specified on the assumption that regular files may be executed, and the system should prevent users from making executable *setgid()* files perform with privileges that the caller does not have. On implementations that support execution of other file types, the *S_ISGID* bit should be cleared for those file types under the same circumstances.

Implementations that use the *S_ISUID* bit to indicate some other function (for example, mandatory record locking) on non-executable files need not clear this bit on writing. They should clear the bit for executable files and any other cases where the bit grants special powers to processes that change the file contents. Similar comments apply to the *S_ISGID* bit.

The purpose of the *fchmodat()* function is to enable changing the mode of files in directories

other than the current working directory without exposure to race conditions. Any part of the path of a file could be changed in parallel to a call to *chmod()*, resulting in unspecified behavior. By opening a file descriptor for the target directory and using the *fchmodat()* function it can be guaranteed that the changed file is located relative to the desired directory. Some implementations might allow changing the mode of symbolic links. This is not supported by the interfaces in the POSIX specification. Systems with such support provide an interface named *lchmod()*. To support such implementations *fchmodat()* has a *flag* parameter.

FUTURE DIRECTIONS

None.

SEE ALSO

access(), *chown()*, *exec*, *fstatat()*, *fstatvfs()*, *mkdir()*, *mkfifo()*, *mknod()*, *open()*

XBD [*<fcntl.h>*](#), [*<sys/stat.h>*](#), [*<sys/types.h>*](#)

CHANGE HISTORY

First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The requirement to include [*<sys/types.h>*](#) has been removed. Although [*<sys/types.h>*](#) was required for conforming implementations of previous POSIX specifications, it was not required for UNIX applications.
- The [EINVAL] and [EINTR] optional error conditions are added.
- A second [ENAMETOOLONG] is added as an optional error condition.

The following changes were made to align with the IEEE P1003.1a draft standard:

- The [ELOOP] optional error condition is added.

The normative text is updated to avoid use of the term “must” for application requirements.

Issue 7

Austin Group Interpretation 1003.1-2001 #143 is applied.

The *fchmodat()* function is added from The Open Group Technical Standard, 2006, Extended API Set Part 2.

Changes are made related to support for finegrained timestamps.

Changes are made to allow a directory to be opened for searching.

The [ENOTDIR] error condition is clarified to cover the condition where the last component of a pathname exists but is not a directory or a symbolic link to a directory.

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0048 [300], XSH/TC1-2008/0049 [461], + XSH/TC1-2008/0050 [324], XSH/TC1-2008/0051 [278], and XSH/TC1-2008/0052 [278] are + applied.

NAME

chown, fchownat — change owner and group of a file relative to directory file descriptor

SYNOPSIS

```
#include <unistd.h>

int chown(const char *path, uid_t owner, gid_t group);
int fchownat(int fd, const char *path, uid_t owner, gid_t group,
             int flag);
```

DESCRIPTION

The *chown()* function shall change the user and group ownership of a file.

The *path* argument points to a pathname naming a file. The user ID and group ID of the named file shall be set to the numeric values contained in *owner* and *group*, respectively.

Only processes with an effective user ID equal to the user ID of the file or with appropriate privileges may change the ownership of a file. If `_POSIX_CHOWN_RESTRICTED` is in effect for *path*:

- Changing the user ID is restricted to processes with appropriate privileges.
- Changing the group ID is permitted to a process with an effective user ID equal to the user ID of the file, but without appropriate privileges, if and only if *owner* is equal to the file's user ID or `(uid_t)-1` and *group* is equal either to the calling process' effective group ID or to one of its supplementary group IDs.

If the specified file is a regular file, one or more of the `S_IXUSR`, `S_IXGRP`, or `S_IXOTH` bits of the file mode are set, and the process does not have appropriate privileges, the set-user-ID (`S_ISUID`) and set-group-ID (`S_ISGID`) bits of the file mode shall be cleared upon successful return from *chown()*. If the specified file is a regular file, one or more of the `S_IXUSR`, `S_IXGRP`, or `S_IXOTH` bits of the file mode are set, and the process has appropriate privileges, it is implementation-defined whether the set-user-ID and set-group-ID bits are altered. If the *chown()* function is successfully invoked on a file that is not a regular file and one or more of the `S_IXUSR`, `S_IXGRP`, or `S_IXOTH` bits of the file mode are set, the set-user-ID and set-group-ID bits may be cleared.

If *owner* or *group* is specified as `(uid_t)-1` or `(gid_t)-1`, respectively, the corresponding ID of the file shall not be changed. If both *owner* and *group* are `-1`, the times need not be updated.

Upon successful completion, *chown()* shall mark for update the last file status change timestamp of the file.

The *fchownat()* function shall be equivalent to the *chown()* and *lchown()* functions except in the case where *path* specifies a relative path. In this case the file to be changed is determined relative to the directory associated with the file descriptor *fd* instead of the current working directory. If the file descriptor was opened without `O_SEARCH`, the function shall check whether directory searches are permitted using the current permissions of the directory underlying the file descriptor. If the file descriptor was opened with `O_SEARCH`, the function shall not perform the check.

Values for *flag* are constructed by a bitwise-inclusive OR of flags from the following list, defined in `<fcntl.h>`:

`AT_SYMLINK_NOFOLLOW`

If *path* names a symbolic link, ownership of the symbolic link is changed.

If *fchownat()* is passed the special value `AT_FDCWD` in the *fd* parameter, the current working directory shall be used and the behavior shall be identical to a call to *chown()* or *lchown()*.

22538 respectively, depending on whether or not the AT_SYMLINK_NOFOLLOW bit is set in the *flag*
 22539 argument.

22540 RETURN VALUE

22541 Upon successful completion, these functions shall return 0. Otherwise, these functions shall
 22542 return -1 and set *errno* to indicate the error. If -1 is returned, no changes are made in the user ID
 22543 and group ID of the file.

22544 ERRORS

22545 These functions shall fail if:

22546 [EACCES] Search permission is denied on a component of the path prefix.

22547 [ELOOP] A loop exists in symbolic links encountered during resolution of the *path*
 22548 argument.

22549 [ENAMETOOLONG]

22550 The length of a component of a pathname is longer than {NAME_MAX}.

22551 [ENOENT] A component of *path* does not name an existing file or *path* is an empty string.

22552 [ENOTDIR] A component of the path prefix names an existing file that is neither a |
 22553 directory nor a symbolic link to a directory, or the *path* argument contains at
 22554 least one non-*<slash>* character and ends with one or more trailing *<slash>*
 22555 characters and the last pathname component names an existing file that is
 22556 neither a directory nor a symbolic link to a directory.

22557 [EPERM] The effective user ID does not match the owner of the file, or the calling
 22558 process does not have appropriate privileges and
 22559 _POSIX_CHOWN_RESTRICTED indicates that such privilege is required.

22560 [EROFS] The named file resides on a read-only file system.

22561 The *fchownat()* function shall fail if:

22562 [EACCES] *fd* was not opened with O_SEARCH and the permissions of the directory
 22563 underlying *fd* do not permit directory searches.

22564 [EBADF] The *path* argument does not specify an absolute path and the *fd* argument is
 22565 neither AT_FDCWD nor a valid file descriptor open for reading or searching.

22566 [ENOTDIR] The *path* argument is not an absolute path and *fd* is a file descriptor associated +
 22567 with a non-directory file.

22568 These functions may fail if:

22569 [EIO] An I/O error occurred while reading or writing to the file system.

22570 [EINTR] The *chown()* function was interrupted by a signal which was caught.

22571 [EINVAL] The owner or group ID supplied is not a value supported by the
 22572 implementation.

22573 [ELOOP] More than {SYMLOOP_MAX} symbolic links were encountered during
 22574 resolution of the *path* argument.

22575 [ENAMETOOLONG]

22576 The length of a pathname exceeds {PATH_MAX}, or pathname resolution of a
 22577 symbolic link produced an intermediate result with a length that exceeds
 22578 {PATH_MAX}.

22579 The *fchownat()* function may fail if:
 22580 [EINVAL] The value of the *flag* argument is not valid. -

22581 EXAMPLES

22582 None.

22583 APPLICATION USAGE

22584 Although *chown()* can be used on some implementations by the file owner to change the owner
 22585 and group to any desired values, the only portable use of this function is to change the group of
 22586 a file to the effective GID of the calling process or to a member of its group set.

22587 RATIONALE

22588 System III and System V allow a user to give away files; that is, the owner of a file may change
 22589 its user ID to anything. This is a serious problem for implementations that are intended to meet
 22590 government security regulations. Version 7 and 4.3 BSD permit only the superuser to change the
 22591 user ID of a file. Some government agencies (usually not ones concerned directly with security)
 22592 find this limitation too confining. This volume of POSIX.1-2008 uses *may* to permit secure
 22593 implementations while not disallowing System V.

22594 System III and System V allow the owner of a file to change the group ID to anything. Version 7
 22595 permits only the superuser to change the group ID of a file. 4.3 BSD permits the owner to
 22596 change the group ID of a file to its effective group ID or to any of the groups in the list of
 22597 supplementary group IDs, but to no others.

22598 The POSIX.1-1990 standard requires that the *chown()* function invoked by a non-appropriate
 22599 privileged process clear the S_ISGID and the S_ISUID bits for regular files, and permits them to
 22600 be cleared for other types of files. This is so that changes in accessibility do not accidentally
 22601 cause files to become security holes. Unfortunately, requiring these bits to be cleared on non-
 22602 executable data files also clears the mandatory file locking bit (shared with S_ISGID), which is
 22603 an extension on many implementations (it first appeared in System V). These bits should only be
 22604 required to be cleared on regular files that have one or more of their execute bits set.

22605 The purpose of the *fchownat()* function is to enable changing ownership of files in directories
 22606 other than the current working directory without exposure to race conditions. Any part of the
 22607 path of a file could be changed in parallel to a call to *chown()* or *lchown()*, resulting in
 22608 unspecified behavior. By opening a file descriptor for the target directory and using the
 22609 *fchownat()* function it can be guaranteed that the changed file is located relative to the desired
 22610 directory.

22611 FUTURE DIRECTIONS

22612 None.

22613 SEE ALSO

22614 *chmod()*, *fpathconf()*, *lchown()*

22615 XBD <fcntl.h>, <sys/types.h>, <unistd.h>

22616 CHANGE HISTORY

22617 First released in Issue 1. Derived from Issue 1 of the SVID.

22618 Issue 6

22619 The following changes are made for alignment with the ISO POSIX-1:1996 standard:

- 22620 • The wording describing the optional dependency on `_POSIX_CHOWN_RESTRICTED` is
- 22621 restored.

- The [EPERM] error is restored as an error dependent on _POSIX_CHOWN_RESTRICTED. This is since its operand is a pathname and applications should be aware that the error may not occur for that pathname if the file system does not support _POSIX_CHOWN_RESTRICTED.

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The requirement to include `<sys/types.h>` has been removed. Although `<sys/types.h>` was required for conforming implementations of previous POSIX specifications, it was not required for UNIX applications.
- The value for *owner* of `(uid_t)-1` allows the use of `-1` by the owner of a file to change the group ID only. A corresponding change is made for group.
- The [ELOOP] mandatory error condition is added.
- The [EIO] and [EINTR] optional error conditions are added.
- A second [ENAMETOOLONG] is added as an optional error condition.

The following changes were made to align with the IEEE P1003.1a draft standard:

- Clarification is added that the `S_ISUID` and `S_ISGID` bits do not need to be cleared when the process has appropriate privileges.
- The [ELOOP] optional error condition is added.

Issue 7

Austin Group Interpretation 1003.1-2001 #143 is applied.

The `fchownat()` function is added from The Open Group Technical Standard, 2006, Extended API Set Part 2.

Changes are made related to support for finegrained timestamps.

Changes are made to allow a directory to be opened for searching.

The [ENOTDIR] error condition is clarified to cover the condition where the last component of a pathname exists but is not a directory or a symbolic link to a directory.

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0053 [461], XSH/TC1-2008/0054 [324], + XSH/TC1-2008/0055 [278], and XSH/TC1-2008/0056 [278] are applied.

22650 NAME

22651 cimag, cimagf, cimagl — complex imaginary functions

22652 SYNOPSIS

```
22653 #include <complex.h>
22654 double cimag(double complex z);
22655 float cimagf(float complex z);
22656 long double cimagl(long double complex z);
```

22657 DESCRIPTION

22658 CX The functionality described on this reference page is aligned with the ISO C standard. Any
 22659 conflict between the requirements described here and the ISO C standard is unintentional. This
 22660 volume of POSIX.1-2008 defers to the ISO C standard.

22661 These functions shall compute the imaginary part of *z*.

22662 RETURN VALUE

22663 These functions shall return the imaginary part value (as a real).

22664 ERRORS

22665 No errors are defined.

22666 EXAMPLES

22667 None.

22668 APPLICATION USAGE

22669 For a variable *z* of complex type:

```
22670 z == creal(z) + cimag(z)*I
```

22671 RATIONALE

22672 None.

22673 FUTURE DIRECTIONS

22674 None.

22675 SEE ALSO

22676 *carg()*, *conj()*, *cproj()*, *creal()*

22677 XBD **<complex.h>**

22678 CHANGE HISTORY

22679 First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

22680 **NAME**

22681 clearerr — clear indicators on a stream

22682 **SYNOPSIS**

22683 #include <stdio.h>

22684 void clearerr(FILE **stream*);22685 **DESCRIPTION**

22686 CX The functionality described on this reference page is aligned with the ISO C standard. Any
 22687 conflict between the requirements described here and the ISO C standard is unintentional. This
 22688 volume of POSIX.1-2008 defers to the ISO C standard.

22689 The *clearerr()* function shall clear the end-of-file and error indicators for the stream to which
 22690 *stream* points.

22691 CX The *clearerr()* function shall not change the setting of *errno* if *stream* is valid. +

22692 **RETURN VALUE**22693 The *clearerr()* function shall not return a value.22694 **ERRORS**

22695 No errors are defined.

22696 **EXAMPLES**

22697 None.

22698 **APPLICATION USAGE**

22699 None.

22700 **RATIONALE**

22701 None.

22702 **FUTURE DIRECTIONS**

22703 None.

22704 **SEE ALSO**

22705 XBD <stdio.h>

22706 **CHANGE HISTORY**

22707 First released in Issue 1. Derived from Issue 1 of the SVID.

22708 **Issue 7**

22709 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0057 [401] is applied. +

22710 **NAME**

22711 clock — report CPU time used

22712 **SYNOPSIS**

22713 #include <time.h>

22714 clock_t clock(void);

22715 **DESCRIPTION**

22716 CX The functionality described on this reference page is aligned with the ISO C standard. Any
 22717 conflict between the requirements described here and the ISO C standard is unintentional. This
 22718 volume of POSIX.1-2008 defers to the ISO C standard.

22719 The *clock()* function shall return the implementation's best approximation to the processor time
 22720 used by the process since the beginning of an implementation-defined era related only to the
 22721 process invocation.

22722 **RETURN VALUE**

22723 To determine the time in seconds, the value returned by *clock()* should be divided by the value
 22724 XSI of the macro CLOCKS_PER_SEC. CLOCKS_PER_SEC is defined to be one million in <time.h>.
 22725 If the processor time used is not available or its value cannot be represented, the function shall
 22726 return the value (clock_t)−1.

22727 **ERRORS**

22728 No errors are defined.

22729 **EXAMPLES**

22730 None.

22731 **APPLICATION USAGE**

22732 In order to measure the time spent in a program, *clock()* should be called at the start of the
 22733 program and its return value subtracted from the value returned by subsequent calls. The value
 22734 returned by *clock()* is defined for compatibility across systems that have clocks with different
 22735 resolutions. The resolution on any particular system need not be to microsecond accuracy.

22736 The value returned by *clock()* may wrap around on some implementations. For example, on a
 22737 machine with 32-bit values for **clock_t**, it wraps after 2 147 seconds or 36 minutes.

22738 **RATIONALE**

22739 None.

22740 **FUTURE DIRECTIONS**

22741 None.

22742 **SEE ALSO**22743 *asctime()*, *ctime()*, *difftime()*, *gmtime()*, *localtime()*, *mktime()*, *strftime()*, *strptime()*, *time()*, *utime()*

22744 XBD <time.h>

22745 **CHANGE HISTORY**

22746 First released in Issue 1. Derived from Issue 1 of the SVID.

NAME

clock_getcpuclockid — access a process CPU-time clock (**ADVANCED REALTIME**)

SYNOPSIS

```
#include <time.h>

int clock_getcpuclockid(pid_t pid, clockid_t *clock_id);
```

DESCRIPTION

The *clock_getcpuclockid()* function shall return the clock ID of the CPU-time clock of the process specified by *pid*. If the process described by *pid* exists and the calling process has permission, the clock ID of this clock shall be returned in *clock_id*.

If *pid* is zero, the *clock_getcpuclockid()* function shall return the clock ID of the CPU-time clock of the process making the call, in *clock_id*.

The conditions under which one process has permission to obtain the CPU-time clock ID of other processes are implementation-defined.

RETURN VALUE

Upon successful completion, *clock_getcpuclockid()* shall return zero; otherwise, an error number shall be returned to indicate the error.

ERRORS

The *clock_getcpuclockid()* function shall fail if:

[EPERM] The requesting process does not have permission to access the CPU-time clock for the process.

The *clock_getcpuclockid()* function may fail if:

[ESRCH] No process can be found corresponding to the process specified by *pid*.

EXAMPLES

None.

APPLICATION USAGE

The *clock_getcpuclockid()* function is part of the Process CPU-Time Clocks option and need not be provided on all implementations.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

clock_getres(), *timer_create()*

XBD **<time.h>**

CHANGE HISTORY

First released in Issue 6. Derived from IEEE Std 1003.1d-1999.

In the SYNOPSIS, the inclusion of **<sys/types.h>** is no longer required.

22784 **NAME**

22785 clock_getres, clock_gettime, clock_settime — clock and timer functions

22786 **SYNOPSIS**

```

22787 CX    #include <time.h>
22788
22788     int clock_getres(clockid_t clock_id, struct timespec *res);
22789     int clock_gettime(clockid_t clock_id, struct timespec *tp);
22790     int clock_settime(clockid_t clock_id, const struct timespec *tp);

```

22791 **DESCRIPTION**

22792 The *clock_getres()* function shall return the resolution of any clock. Clock resolutions are
 22793 implementation-defined and cannot be set by a process. If the argument *res* is not NULL, the
 22794 resolution of the specified clock shall be stored in the location pointed to by *res*. If *res* is NULL,
 22795 the clock resolution is not returned. If the *time* argument of *clock_settime()* is not a multiple of *res*,
 22796 then the value is truncated to a multiple of *res*.

22797 The *clock_gettime()* function shall return the current value *tp* for the specified clock, *clock_id*.

22798 The *clock_settime()* function shall set the specified clock, *clock_id*, to the value specified by *tp*.
 22799 Time values that are between two consecutive non-negative integer multiples of the resolution of
 22800 the specified clock shall be truncated down to the smaller multiple of the resolution.

22801 A clock may be system-wide (that is, visible to all processes) or per-process (measuring time that
 22802 is meaningful only within a process). All implementations shall support a *clock_id* of
 22803 CLOCK_REALTIME as defined in **<time.h>**. This clock represents the clock measuring real time
 22804 for the system. For this clock, the values returned by *clock_gettime()* and specified by
 22805 *clock_settime()* represent the amount of time (in seconds and nanoseconds) since the Epoch. An
 22806 implementation may also support additional clocks. The interpretation of time values for these
 22807 clocks is unspecified.

22808 If the value of the CLOCK_REALTIME clock is set via *clock_settime()*, the new value of the clock
 22809 shall be used to determine the time of expiration for absolute time services based upon the
 22810 CLOCK_REALTIME clock. This applies to the time at which armed absolute timers expire. If the
 22811 absolute time requested at the invocation of such a time service is before the new value of the
 22812 clock, the time service shall expire immediately as if the clock had reached the requested time
 22813 normally.

22814 Setting the value of the CLOCK_REALTIME clock via *clock_settime()* shall have no effect on
 22815 threads that are blocked waiting for a relative time service based upon this clock, including the
 22816 *nanosleep()* function; nor on the expiration of relative timers based upon this clock.
 22817 Consequently, these time services shall expire when the requested relative interval elapses,
 22818 independently of the new or old value of the clock.

22819 MON If the Monotonic Clock option is supported, all implementations shall support a *clock_id* of
 22820 CLOCK_MONOTONIC defined in **<time.h>**. This clock represents the monotonic clock for the
 22821 system. For this clock, the value returned by *clock_gettime()* represents the amount of time (in
 22822 seconds and nanoseconds) since an unspecified point in the past (for example, system start-up
 22823 time, or the Epoch). This point does not change after system start-up time. The value of the
 22824 CLOCK_MONOTONIC clock cannot be set via *clock_settime()*. This function shall fail if it is
 22825 invoked with a *clock_id* argument of CLOCK_MONOTONIC.

22826 The effect of setting a clock via *clock_settime()* on armed per-process timers associated with a
 22827 clock other than CLOCK_REALTIME is implementation-defined.

22828 If the value of the CLOCK_REALTIME clock is set via *clock_settime()*, the new value of the clock
 22829 shall be used to determine the time at which the system shall awaken a thread blocked on an

absolute *clock_nanosleep()* call based upon the CLOCK_REALTIME clock. If the absolute time requested at the invocation of such a time service is before the new value of the clock, the call shall return immediately as if the clock had reached the requested time normally.

Setting the value of the CLOCK_REALTIME clock via *clock_settime()* shall have no effect on any thread that is blocked on a relative *clock_nanosleep()* call. Consequently, the call shall return when the requested relative interval elapses, independently of the new or old value of the clock.

Appropriate privileges to set a particular clock are implementation-defined.

CPT If `_POSIX_CPUTIME` is defined, implementations shall support clock ID values obtained by invoking *clock_getcpuclockid()*, which represent the CPU-time clock of a given process. Implementations shall also support the special `clockid_t` value `CLOCK_PROCESS_CPUTIME_ID`, which represents the CPU-time clock of the calling process when invoking one of the *clock_**() or *timer_**() functions. For these clock IDs, the values returned by *clock_gettime()* and specified by *clock_settime()* represent the amount of execution time of the process associated with the clock. Changing the value of a CPU-time clock via *clock_settime()* shall have no effect on the behavior of the sporadic server scheduling policy (see [Scheduling Policies](#)).

TCT If `_POSIX_THREAD_CPUTIME` is defined, implementations shall support clock ID values obtained by invoking *pthread_getcpuclockid()*, which represent the CPU-time clock of a given thread. Implementations shall also support the special `clockid_t` value `CLOCK_THREAD_CPUTIME_ID`, which represents the CPU-time clock of the calling thread when invoking one of the *clock_**() or *timer_**() functions. For these clock IDs, the values returned by *clock_gettime()* and specified by *clock_settime()* shall represent the amount of execution time of the thread associated with the clock. Changing the value of a CPU-time clock via *clock_settime()* shall have no effect on the behavior of the sporadic server scheduling policy (see [Scheduling Policies](#)).

22855 RETURN VALUE

A return value of 0 shall indicate that the call succeeded. A return value of -1 shall indicate that an error occurred, and *errno* shall be set to indicate the error.

22858 ERRORS

The *clock_getres()*, *clock_gettime()*, and *clock_settime()* functions shall fail if:

[EINVAL] The *clock_id* argument does not specify a known clock.

The *clock_gettime()* function shall fail if: +

[EOVERFLOW] The number of seconds will not fit in an object of type `time_t`. +

The *clock_settime()* function shall fail if:

[EINVAL] The *tp* argument to *clock_settime()* is outside the range for the given clock ID.

[EINVAL] The *tp* argument specified a nanosecond value less than zero or greater than or equal to 1 000 million.

MON [EINVAL] The value of the *clock_id* argument is `CLOCK_MONOTONIC`.

The *clock_settime()* function may fail if:

[EPERM] The requesting process does not have appropriate privileges to set the specified clock.

EXAMPLES

None.

APPLICATION USAGE

Note that the absolute value of the monotonic clock is meaningless (because its origin is arbitrary), and thus there is no need to set it. Furthermore, realtime applications can rely on the fact that the value of this clock is never set and, therefore, that time intervals measured with this clock will not be affected by calls to *clock_settime()*.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

[Scheduling Policies](#) (on page 506), *clock_getcpuclockid()*, *clock_nanosleep()*, *ctime()*, *mq_receive()*, *mq_send()*, *nanosleep()*, *pthread_mutex_timedlock()*, *sem_timedwait()*, *time()*, *timer_create()*, *timer_getoverrun()*

XBD [<time.h>](#)

CHANGE HISTORY

First released in Issue 5. Included for alignment with the POSIX Realtime Extension.

Issue 6

The [ENOSYS] error condition has been removed as stubs need not be provided if an implementation does not support the Timers option.

The APPLICATION USAGE section is added.

The following changes were made to align with the IEEE P1003.1a draft standard:

- Clarification is added of the effect of resetting the clock resolution.

CPU-time clocks and the *clock_getcpuclockid()* function are added for alignment with IEEE Std 1003.1d-1999.

The following changes are added for alignment with IEEE Std 1003.1j-2000:

- The DESCRIPTION is updated as follows:
 - The value returned by *clock_gettime()* for CLOCK_MONOTONIC is specified.
 - The *clock_settime()* function failing for CLOCK_MONOTONIC is specified.
 - The effects of *clock_settime()* on the *clock_nanosleep()* function with respect to CLOCK_REALTIME are specified.
- An [EINVAL] error is added to the ERRORS section, indicating that *clock_settime()* fails for CLOCK_MONOTONIC.
- The APPLICATION USAGE section notes that the CLOCK_MONOTONIC clock need not and shall not be set by *clock_settime()* since the absolute value of the CLOCK_MONOTONIC clock is meaningless.
- The *clock_nanosleep()*, *mq_timedreceive()*, *mq_timedsend()*, *pthread_mutex_timedlock()*, *sem_timedwait()*, *timer_create()*, and *timer_settime()* functions are added to the SEE ALSO section.

22911	Issue 7	
22912	Functionality relating to the Clock Selection option is moved to the Base.	
22913	The <i>clock_getres()</i> , <i>clock_gettime()</i> , and <i>clock_settime()</i> functions are moved from the Timers	
22914	option to the Base.	
22915	POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0058 [106] is applied.	+

NAME

clock_nanosleep — high resolution sleep with specifiable clock

SYNOPSIS

```
#include <time.h>

int clock_nanosleep(clockid_t clock_id, int flags,
    const struct timespec *rqtp, struct timespec *rmtp);
```

DESCRIPTION

If the flag `TIMER_ABSTIME` is not set in the *flags* argument, the *clock_nanosleep()* function shall cause the current thread to be suspended from execution until either the time interval specified by the *rqtp* argument has elapsed, or a signal is delivered to the calling thread and its action is to invoke a signal-catching function, or the process is terminated. The clock used to measure the time shall be the clock specified by *clock_id*.

If the flag `TIMER_ABSTIME` is set in the *flags* argument, the *clock_nanosleep()* function shall cause the current thread to be suspended from execution until either the time value of the clock specified by *clock_id* reaches the absolute time specified by the *rqtp* argument, or a signal is delivered to the calling thread and its action is to invoke a signal-catching function, or the process is terminated. If, at the time of the call, the time value specified by *rqtp* is less than or equal to the time value of the specified clock, then *clock_nanosleep()* shall return immediately and the calling process shall not be suspended.

The suspension time caused by this function may be longer than requested because the argument value is rounded up to an integer multiple of the sleep resolution, or because of the scheduling of other activity by the system. But, except for the case of being interrupted by a signal, the suspension time for the relative *clock_nanosleep()* function (that is, with the `TIMER_ABSTIME` flag not set) shall not be less than the time interval specified by *rqtp*, as measured by the corresponding clock. The suspension for the absolute *clock_nanosleep()* function (that is, with the `TIMER_ABSTIME` flag set) shall be in effect at least until the value of the corresponding clock reaches the absolute time specified by *rqtp*, except for the case of being interrupted by a signal.

The use of the *clock_nanosleep()* function shall have no effect on the action or blockage of any signal.

The *clock_nanosleep()* function shall fail if the *clock_id* argument refers to the CPU-time clock of the calling thread. It is unspecified whether *clock_id* values of other CPU-time clocks are allowed.

RETURN VALUE

If the *clock_nanosleep()* function returns because the requested time has elapsed, its return value shall be zero.

If the *clock_nanosleep()* function returns because it has been interrupted by a signal, it shall return the corresponding error value. For the relative *clock_nanosleep()* function, if the *rmtp* argument is non-NULL, the **timespec** structure referenced by it shall be updated to contain the amount of time remaining in the interval (the requested time minus the time actually slept). If the *rmtp* argument is NULL, the remaining time is not returned. The absolute *clock_nanosleep()* function has no effect on the structure referenced by *rmtp*.

If *clock_nanosleep()* fails, it shall return the corresponding error value.

ERRORS

The *clock_nanosleep()* function shall fail if:

- [EINTR] The *clock_nanosleep()* function was interrupted by a signal.
- [EINVAL] The *rqt* argument specified a nanosecond value less than zero or greater than or equal to 1 000 million; or the *TIMER_ABSTIME* flag was specified in *flags* and the *rqt* argument is outside the range for the clock specified by *clock_id*; or the *clock_id* argument does not specify a known clock, or specifies the CPU-time clock of the calling thread.
- [ENOTSUP] The *clock_id* argument specifies a clock for which *clock_nanosleep()* is not supported, such as a CPU-time clock.

EXAMPLES

None.

APPLICATION USAGE

Calling *clock_nanosleep()* with the value *TIMER_ABSTIME* not set in the *flags* argument and with a *clock_id* of *CLOCK_REALTIME* is equivalent to calling *nanosleep()* with the same *rqt* and *rmt* arguments.

RATIONALE

The *nanosleep()* function specifies that the system-wide clock *CLOCK_REALTIME* is used to measure the elapsed time for this time service. However, with the introduction of the monotonic clock *CLOCK_MONOTONIC* a new relative sleep function is needed to allow an application to take advantage of the special characteristics of this clock.

There are many applications in which a process needs to be suspended and then activated multiple times in a periodic way; for example, to poll the status of a non-interrupting device or to refresh a display device. For these cases, it is known that precise periodic activation cannot be achieved with a relative *sleep()* or *nanosleep()* function call. Suppose, for example, a periodic process that is activated at time *T0*, executes for a while, and then wants to suspend itself until time *T0+T*, the period being *T*. If this process wants to use the *nanosleep()* function, it must first call *clock_gettime()* to get the current time, then calculate the difference between the current time and *T0+T* and, finally, call *nanosleep()* using the computed interval. However, the process could be preempted by a different process between the two function calls, and in this case the interval computed would be wrong; the process would wake up later than desired. This problem would not occur with the absolute *clock_nanosleep()* function, since only one function call would be necessary to suspend the process until the desired time. In other cases, however, a relative sleep is needed, and that is why both functionalities are required.

Although it is possible to implement periodic processes using the timers interface, this implementation would require the use of signals, and the reservation of some signal numbers. In this regard, the reasons for including an absolute version of the *clock_nanosleep()* function in POSIX.1-2008 are the same as for the inclusion of the relative *nanosleep()*.

It is also possible to implement precise periodic processes using *pthread_cond_timedwait()*, in which an absolute timeout is specified that takes effect if the condition variable involved is never signaled. However, the use of this interface is unnatural, and involves performing other operations on mutexes and condition variables that imply an unnecessary overhead. Furthermore, *pthread_cond_timedwait()* is not available in implementations that do not support threads.

Although the interface of the relative and absolute versions of the new high resolution sleep service is the same *clock_nanosleep()* function, the *rmt* argument is only used in the relative sleep. This argument is needed in the relative *clock_nanosleep()* function to reissue the function

23005 call if it is interrupted by a signal, but it is not needed in the absolute *clock_nanosleep()* function
23006 call; if the call is interrupted by a signal, the absolute *clock_nanosleep()* function can be invoked
23007 again with the same *rqtp* argument used in the interrupted call.

23008 **FUTURE DIRECTIONS**

23009 None.

23010 **SEE ALSO**

23011 *clock_getres()*, *nanosleep()*, *pthread_cond_timedwait()*, *sleep()*

23012 XBD <**time.h**>

23013 **CHANGE HISTORY**

23014 First released in Issue 6. Derived from IEEE Std 1003.1j-2000.

23015 **Issue 7**

23016 The *clock_nanosleep()* function is moved from the Clock Selection option to the Base.

23017 **NAME**

23018 clock_settime — clock and timer functions

23019 **SYNOPSIS**

```
23020 CX      #include <time.h>
23021          int clock_settime(clockid_t clock_id, const struct timespec *tp);
```

23022 **DESCRIPTION**23023 Refer to *clock_getres()*.

23024 **NAME**

23025 clog, clogf, clogl — complex natural logarithm functions

23026 **SYNOPSIS**

```
23027 #include <complex.h>
23028 double complex clog(double complex z);
23029 float complex clogf(float complex z);
23030 long double complex clogl(long double complex z);
```

23031 **DESCRIPTION**

23032 CX The functionality described on this reference page is aligned with the ISO C standard. Any
23033 conflict between the requirements described here and the ISO C standard is unintentional. This
23034 volume of POSIX.1-2008 defers to the ISO C standard.

23035 These functions shall compute the complex natural (base e) logarithm of z , with a branch cut
23036 along the negative real axis.

23037 **RETURN VALUE**

23038 These functions shall return the complex natural logarithm value, in the range of a strip
23039 mathematically unbounded along the real axis and in the interval $[-i\pi, +i\pi]$ along the imaginary
23040 axis.

23041 **ERRORS**

23042 No errors are defined.

23043 **EXAMPLES**

23044 None.

23045 **APPLICATION USAGE**

23046 None.

23047 **RATIONALE**

23048 None.

23049 **FUTURE DIRECTIONS**

23050 None.

23051 **SEE ALSO**23052 [cexp\(\)](#)23053 XBD [<complex.h>](#)23054 **CHANGE HISTORY**

23055 First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

23056 **NAME**

23057 close — close a file descriptor

23058 **SYNOPSIS**

23059 #include <unistd.h>

23060 int close(int *fildes*);23061 **DESCRIPTION**

23062 The *close()* function shall deallocate the file descriptor indicated by *fildes*. To deallocate means to
 23063 make the file descriptor available for return by subsequent calls to *open()* or other functions that
 23064 allocate file descriptors. All outstanding record locks owned by the process on the file associated
 23065 with the file descriptor shall be removed (that is, unlocked).

23066 If *close()* is interrupted by a signal that is to be caught, it shall return -1 with *errno* set to [EINTR]
 23067 and the state of *fildes* is unspecified. If an I/O error occurred while reading from or writing to
 23068 the file system during *close()*, it may return -1 with *errno* set to [EIO]; if this error is returned, the
 23069 state of *fildes* is unspecified.

23070 When all file descriptors associated with a pipe or FIFO special file are closed, any data
 23071 remaining in the pipe or FIFO shall be discarded.

23072 When all file descriptors associated with an open file description have been closed, the open file
 23073 description shall be freed.

23074 If the link count of the file is 0, when all file descriptors associated with the file are closed, the
 23075 space occupied by the file shall be freed and the file shall no longer be accessible.

23076 OB XSR If a STREAMS-based *fildes* is closed and the calling process was previously registered to receive
 23077 a SIGPOLL signal for events associated with that STREAM, the calling process shall be
 23078 unregistered for events associated with the STREAM. The last *close()* for a STREAM shall cause
 23079 the STREAM associated with *fildes* to be dismantled. If O_NONBLOCK is not set and there have
 23080 been no signals posted for the STREAM, and if there is data on the module's write queue, *close()*
 23081 shall wait for an unspecified time (for each module and driver) for any output to drain before
 23082 dismantling the STREAM. The time delay can be changed via an I_SETCLTIME *ioctl()* request. If
 23083 the O_NONBLOCK flag is set, or if there are any pending signals, *close()* shall not wait for
 23084 output to drain, and shall dismantle the STREAM immediately.

23085 If the implementation supports STREAMS-based pipes, and *fildes* is associated with one end of a
 23086 pipe, the last *close()* shall cause a hangup to occur on the other end of the pipe. In addition, if the
 23087 other end of the pipe has been named by *fattach()*, then the last *close()* shall force the named end
 23088 to be detached by *fdetach()*. If the named end has no open file descriptors associated with it and
 23089 gets detached, the STREAM associated with that end shall also be dismantled.

23090 XSI If *fildes* refers to the master side of a pseudo-terminal, and this is the last close, a SIGHUP signal
 23091 shall be sent to the controlling process, if any, for which the slave side of the pseudo-terminal is
 23092 the controlling terminal. It is unspecified whether closing the master side of the pseudo-terminal
 23093 flushes all queued input and output.

23094 OB XSR If *fildes* refers to the slave side of a STREAMS-based pseudo-terminal, a zero-length message
 23095 may be sent to the master.

23096 When there is an outstanding cancelable asynchronous I/O operation against *fildes* when *close()*
 23097 is called, that I/O operation may be canceled. An I/O operation that is not canceled completes
 23098 as if the *close()* operation had not yet occurred. All operations that are not canceled shall
 23099 complete as if the *close()* blocked until the operations completed. The *close()* operation itself
 23100 need not block awaiting such I/O completion. Whether any I/O operation is canceled, and
 23101 which I/O operation may be canceled upon *close()*, is implementation-defined.

23102 SHM If a memory mapped file or a shared memory object remains referenced at the last close (that is,
 23103 a process has it mapped), then the entire contents of the memory object shall persist until the
 23104 SHM memory object becomes unreferenced. If this is the last close of a memory mapped file or a
 23105 shared memory object and the close results in the memory object becoming unreferenced, and
 23106 the memory object has been unlinked, then the memory object shall be removed.

23107 If *fildest* refers to a socket, *close()* shall cause the socket to be destroyed. If the socket is in
 23108 connection-mode, and the `SO_LINGER` option is set for the socket with non-zero linger time,
 23109 and the socket has untransmitted data, then *close()* shall block for up to the current linger
 23110 interval until all data is transmitted.

23111 RETURN VALUE

23112 Upon successful completion, 0 shall be returned; otherwise, -1 shall be returned and *errno* set to
 23113 indicate the error.

23114 ERRORS

23115 The *close()* function shall fail if:

23116 [EBADF] The *fildest* argument is not a open file descriptor. |

23117 [EINTR] The *close()* function was interrupted by a signal.

23118 The *close()* function may fail if:

23119 [EIO] An I/O error occurred while reading from or writing to the file system.

23120 EXAMPLES

23121 Reassigning a File Descriptor

23122 The following example closes the file descriptor associated with standard output for the current
 23123 process, re-assigns standard output to a new file descriptor, and closes the original file descriptor
 23124 to clean up. This example assumes that the file descriptor 0 (which is the descriptor for standard
 23125 input) is not closed.

```
23126 #include <unistd.h>
23127 ...
23128 int pfd;
23129 ...
23130 close(1);
23131 dup(pfd);
23132 close(pfd);
23133 ...
```

23134 Incidentally, this is exactly what could be achieved using:

```
23135 dup2(pfd, 1);
23136 close(pfd);
```


Closing a File Descriptor

In the following example, *close()* is used to close a file descriptor after an unsuccessful attempt is made to associate that file descriptor with a stream.

```
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>

#define LOCKFILE "/etc/ptmp"
...
int pfd;
FILE *fpfd;
...
if ((fpfd = fdopen (pfd, "w")) == NULL) {
    close(pfd);
    unlink(LOCKFILE);
    exit(1);
}
...
```

APPLICATION USAGE

An application that had used the *stdio* routine *fopen()* to open a file should use the corresponding *fclose()* routine rather than *close()*. Once a file is closed, the file descriptor no longer exists, since the integer corresponding to it no longer refers to a file.

Implementations may use file descriptors that must be inherited into child processes for the child process to remain conforming, such as for message catalog or tracing purposes. Therefore, an application that calls *close()* on an arbitrary integer risks non-conforming behavior, and *close()* can only portably be used on file descriptor values that the application has obtained through explicit actions, as well as the three file descriptors corresponding to the standard file streams. In multi-threaded parent applications, the practice of calling *close()* in a loop after *fork()* and before an *exec* call in order to avoid a race condition of leaking an unintended file descriptor into a child process, is therefore unsafe, and the race should instead be combatted by opening all file descriptors with the FD_CLOEXEC bit set unless the file descriptor is intended to be inherited across *exec*.

RATIONALE

The use of interruptible device close routines should be discouraged to avoid problems with the implicit closes of file descriptors by *exec* and *exit()*. This volume of POSIX.1-2008 only intends to permit such behavior by specifying the [EINTR] error condition.

Note that the requirement for *close()* on a socket to block for up to the current linger interval is not conditional on the O_NONBLOCK setting.

The standard developers rejected a proposal to add *closefrom()* to the standard. Because the standard permits implementations to use inherited file descriptors as a means of providing a conforming environment for the child process, it is not possible to standardize an interface that closes arbitrary file descriptors above a certain value while still guaranteeing a conforming environment.

FUTURE DIRECTIONS

None.

23181 **SEE ALSO**23182 Section 2.6 (on page 499), *exec*, *fattach()*, *fclose()*, *fdetach()*, *fopen()*, *ioctl()*, *open()*, *unlink()*

23183 XBD <unistd.h>

23184 **CHANGE HISTORY**

23185 First released in Issue 1. Derived from Issue 1 of the SVID.

23186 **Issue 5**

23187 The DESCRIPTION is updated for alignment with the POSIX Realtime Extension.

23188 **Issue 6**23189 The DESCRIPTION related to a STREAMS-based file or pseudo-terminal is marked as part of
23190 the XSI STREAMS Option Group.23191 The following new requirements on POSIX implementations derive from alignment with the
23192 Single UNIX Specification:

- 23193 • The [EIO] error condition is added as an optional error.
- 23194 • The DESCRIPTION is updated to describe the state of the *fildes* file descriptor as
23195 unspecified if an I/O error occurs and an [EIO] error condition is returned.

23196 Text referring to sockets is added to the DESCRIPTION.

23197 The DESCRIPTION is updated for alignment with IEEE Std 1003.1j-2000 by specifying that
23198 shared memory objects and memory mapped files (and not typed memory objects) are the types
23199 of memory objects to which the paragraph on last closes applies.

23200 IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/12 is applied, correcting the XSH shaded
23201 text relating to the master side of a pseudo-terminal. The reason for the change is that the
23202 behavior of pseudo-terminals and regular terminals should be as much alike as possible in this
23203 case; the change achieves that and matches historical behavior.

23204 **Issue 7**

23205 Functionality relating to the XSI STREAMS option is marked obsolescent.

23206 Functionality relating to the Asynchronous Input and Output and Memory Mapped Files
23207 options is moved to the Base.

23208 Austin Group Interpretation 1003.1-2001 #139 is applied, clarifying that the requirement for
23209 *close()* on a socket to block for up to the current linger interval is not conditional on the
23210 O_NONBLOCK setting.

23211 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0059 [419], XSH/TC1-2008/0060 [149], +
23212 and XSH/TC1-2008/0061 [149] are applied.

23213 **NAME**

23214 closedir — close a directory stream

23215 **SYNOPSIS**

23216 #include <dirent.h>

23217 int closedir(DIR *dirp);

23218 **DESCRIPTION**

23219 The *closedir()* function shall close the directory stream referred to by the argument *dirp*. Upon
 23220 return, the value of *dirp* may no longer point to an accessible object of the type **DIR**. If a file
 23221 descriptor is used to implement type **DIR**, that file descriptor shall be closed.

23222 **RETURN VALUE**

23223 Upon successful completion, *closedir()* shall return 0; otherwise, -1 shall be returned and *errno*
 23224 set to indicate the error.

23225 **ERRORS**23226 The *closedir()* function may fail if:

23227 [EBADF] The *dirp* argument does not refer to an open directory stream.

23228 [EINTR] The *closedir()* function was interrupted by a signal.

23229 **EXAMPLES**23230 **Closing a Directory Stream**23231 The following program fragment demonstrates how the *closedir()* function is used.

```

23232       ...
23233       DIR *dir;
23234       struct dirent *dp;
23235       ...
23236       if ((dir = opendir(".")) == NULL) {
23237       ...
23238       }
23239       while ((dp = readdir (dir)) != NULL) {
23240       ...
23241       }
23242       closedir(dir);
23243       ...

```

23244 **APPLICATION USAGE**

23245 None.

23246 **RATIONALE**

23247 None.

23248 **FUTURE DIRECTIONS**

23249 None.

23250 **SEE ALSO**23251 *dirfd()*, *fdopendir()*

23252 XBD <dirent.h>

23253 **CHANGE HISTORY**

23254 First released in Issue 2.

23255 **Issue 6**23256 In the SYNOPSIS, the optional include of the **<sys/types.h>** header is removed.23257 The following new requirements on POSIX implementations derive from alignment with the
23258 Single UNIX Specification:

- 23259 • The requirement to include **<sys/types.h>** has been removed. Although **<sys/types.h>** was
23260 required for conforming implementations of previous POSIX specifications, it was not
23261 required for UNIX applications.
- 23262 • The [EINTR] error condition is added as an optional error condition.

NAME

closelog, openlog, setlogmask, syslog — control system log

SYNOPSIS

```
XSI    #include <syslog.h>

void closelog(void);
void openlog(const char *ident, int logopt, int facility);
int setlogmask(int maskpri);
void syslog(int priority, const char *message, ... /* arguments */);
```

DESCRIPTION

The *syslog()* function shall send a message to an implementation-defined logging facility, which may log it in an implementation-defined system log, write it to the system console, forward it to a list of users, or forward it to the logging facility on another host over the network. The logged message shall include a message header and a message body. The message header contains at least a timestamp and a tag string.

The message body is generated from the *message* and following arguments in the same manner as if these were arguments to *printf()*, except that the additional conversion specification *%m* shall be recognized; it shall convert no arguments, shall cause the output of the error message string associated with the value of *errno* on entry to *syslog()*, and may be mixed with argument specifications of the "*%n\$*" form. If a complete conversion specification with the *m* conversion specifier character is not just *%m*, the behavior is undefined. A trailing <newline> may be added if needed.

Values of the *priority* argument are formed by OR'ing together a severity-level value and an optional facility value. If no facility value is specified, the current default facility value is used.

Possible values of severity level include:

LOG_EMERG	A panic condition.
LOG_ALERT	A condition that should be corrected immediately, such as a corrupted system database.
LOG_CRIT	Critical conditions, such as hard device errors.
LOG_ERR	Errors.
LOG_WARNING	Warning messages.
LOG_NOTICE	Conditions that are not error conditions, but that may require special handling.
LOG_INFO	Informational messages.
LOG_DEBUG	Messages that contain information normally of use only when debugging a program.

The facility indicates the application or system component generating the message. Possible facility values include:

LOG_USER	Messages generated by arbitrary processes. This is the default facility identifier if none is specified.
LOG_LOCAL0	Reserved for local use.

23304	LOG_LOCAL1	Reserved for local use.
23305	LOG_LOCAL2	Reserved for local use.
23306	LOG_LOCAL3	Reserved for local use.
23307	LOG_LOCAL4	Reserved for local use.
23308	LOG_LOCAL5	Reserved for local use.
23309	LOG_LOCAL6	Reserved for local use.
23310	LOG_LOCAL7	Reserved for local use.
23311	The <i>openlog()</i> function shall set process attributes that affect subsequent calls to <i>syslog()</i> . The	
23312	<i>ident</i> argument is a string that is prepended to every message. The <i>logopt</i> argument indicates	
23313	logging options. Values for <i>logopt</i> are constructed by a bitwise-inclusive OR of zero or more of	
23314	the following:	
23315	LOG_PID	Log the process ID with each message. This is useful for identifying specific
23316		processes.
23317	LOG_CONS	Write messages to the system console if they cannot be sent to the logging
23318		facility. The <i>syslog()</i> function ensures that the process does not acquire the
23319		console as a controlling terminal in the process of writing the message.
23320	LOG_NDELAY	Open the connection to the logging facility immediately. Normally the open is
23321		delayed until the first message is logged. This is useful for programs that need
23322		to manage the order in which file descriptors are allocated.
23323	LOG_ODELAY	Delay open until <i>syslog()</i> is called.
23324	LOG_NOWAIT	Do not wait for child processes that may have been created during the course
23325		of logging the message. This option should be used by processes that enable
23326		notification of child termination using SIGCHLD, since <i>syslog()</i> may
23327		otherwise block waiting for a child whose exit status has already been
23328		collected.
23329	The <i>facility</i> argument encodes a default facility to be assigned to all messages that do not have an	
23330	explicit facility already encoded. The initial default facility is LOG_USER.	
23331	The <i>openlog()</i> and <i>syslog()</i> functions may allocate a file descriptor. It is not necessary to call	
23332	<i>openlog()</i> prior to calling <i>syslog()</i> .	
23333	The <i>closelog()</i> function shall close any open file descriptors allocated by previous calls to	
23334	<i>openlog()</i> or <i>syslog()</i> .	
23335	The <i>setlogmask()</i> function shall set the log priority mask for the current process to <i>maskpri</i> and	
23336	return the previous mask. If the <i>maskpri</i> argument is 0, the current log mask is not modified.	
23337	Calls by the current process to <i>syslog()</i> with a priority not set in <i>maskpri</i> shall be rejected. The	
23338	default log mask allows all priorities to be logged. A call to <i>openlog()</i> is not required prior to	
23339	calling <i>setlogmask()</i> .	
23340	Symbolic constants for use as values of the <i>logopt</i> , <i>facility</i> , <i>priority</i> , and <i>maskpri</i> arguments are	
23341	defined in the <syslog.h> header.	
23342	RETURN VALUE	
23343	The <i>setlogmask()</i> function shall return the previous log priority mask. The <i>closelog()</i> , <i>openlog()</i> ,	
23344	and <i>syslog()</i> functions shall not return a value.	

23345 ERRORS

23346 No errors are defined.

23347 EXAMPLES**23348 Using openlog()**

23349 The following example causes subsequent calls to *syslog()* to log the process ID with each
23350 message, and to write messages to the system console if they cannot be sent to the logging
23351 facility.

```
23352 #include <syslog.h>
23353 char *ident = "Process demo";
23354 int logopt = LOG_PID | LOG_CONS;
23355 int facility = LOG_USER;
23356 ...
23357 openlog(ident, logopt, facility);
```

23358 Using setlogmask()

23359 The following example causes subsequent calls to *syslog()* to accept error messages, and to reject
23360 all other messages.

```
23361 #include <syslog.h>
23362 int result;
23363 int mask = LOG_MASK (LOG_ERR);
23364 ...
23365 result = setlogmask(mask);
```

23366 Using syslog

23367 The following example sends the message "This is a message" to the default logging
23368 facility, marking the message as an error message generated by random processes.

```
23369 #include <syslog.h>
23370 char *message = "This is a message";
23371 int priority = LOG_ERR | LOG_USER;
23372 ...
23373 syslog(priority, message);
```

23374 APPLICATION USAGE

23375 None.

23376 RATIONALE

23377 None.

23378 FUTURE DIRECTIONS

23379 None.

23380 SEE ALSO

23381 *fprintf()*

23382 XBD *<syslog.h>*

23383 **CHANGE HISTORY**

23384 First released in Issue 4, Version 2.

23385 **Issue 5**

23386 Moved from X/OPEN UNIX extension to BASE.

23387 **Issue 6**23388 IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/13 is applied, correcting the EXAMPLES
23389 section.

23390 **NAME**

23391 confstr — get configurable variables

23392 **SYNOPSIS**

23393 #include <unistd.h>

23394 size_t confstr(int *name*, char **buf*, size_t *len*);23395 **DESCRIPTION**

23396 The *confstr()* function shall return configuration-defined string values. Its use and purpose are
 23397 similar to *sysconf()*, but it is used where string values rather than numeric values are returned.

23398 The *name* argument represents the system variable to be queried. The implementation shall
 23399 support the following name values, defined in <unistd.h>. It may support others:

23400 _CS_PATH
 23401 _CS_POSIX_V7_ILP32_OFF32_CFLAGS
 23402 _CS_POSIX_V7_ILP32_OFF32_LDFLAGS
 23403 _CS_POSIX_V7_ILP32_OFF32_LIBS
 23404 _CS_POSIX_V7_ILP32_OFFBIG_CFLAGS
 23405 _CS_POSIX_V7_ILP32_OFFBIG_LDFLAGS
 23406 _CS_POSIX_V7_ILP32_OFFBIG_LIBS
 23407 _CS_POSIX_V7_LP64_OFF64_CFLAGS
 23408 _CS_POSIX_V7_LP64_OFF64_LDFLAGS
 23409 _CS_POSIX_V7_LP64_OFF64_LIBS
 23410 _CS_POSIX_V7_LPBIG_OFFBIG_CFLAGS
 23411 _CS_POSIX_V7_LPBIG_OFFBIG_LDFLAGS
 23412 _CS_POSIX_V7_LPBIG_OFFBIG_LIBS
 23413 _CS_POSIX_V7_THREADS_CFLAGS
 23414 _CS_POSIX_V7_THREADS_LDFLAGS
 23415 _CS_POSIX_V7_WIDTH_RESTRICTED_ENVS
 23416 _CS_V7_ENV
 23417 OB _CS_POSIX_V6_ILP32_OFF32_CFLAGS
 23418 _CS_POSIX_V6_ILP32_OFF32_LDFLAGS
 23419 _CS_POSIX_V6_ILP32_OFF32_LIBS
 23420 _CS_POSIX_V6_ILP32_OFFBIG_CFLAGS
 23421 _CS_POSIX_V6_ILP32_OFFBIG_LDFLAGS
 23422 _CS_POSIX_V6_ILP32_OFFBIG_LIBS
 23423 _CS_POSIX_V6_LP64_OFF64_CFLAGS
 23424 _CS_POSIX_V6_LP64_OFF64_LDFLAGS
 23425 _CS_POSIX_V6_LP64_OFF64_LIBS
 23426 _CS_POSIX_V6_LPBIG_OFFBIG_CFLAGS
 23427 _CS_POSIX_V6_LPBIG_OFFBIG_LDFLAGS
 23428 _CS_POSIX_V6_LPBIG_OFFBIG_LIBS
 23429 _CS_POSIX_V6_WIDTH_RESTRICTED_ENVS
 23430 _CS_V6_ENV

23431 If *len* is not 0, and if *name* has a configuration-defined value, *confstr()* shall copy that value into
 23432 the *len*-byte buffer pointed to by *buf*. If the string to be returned is longer than *len* bytes,
 23433 including the terminating null, then *confstr()* shall truncate the string to *len*–1 bytes and null-
 23434 terminate the result. The application can detect that the string was truncated by comparing the
 23435 value returned by *confstr()* with *len*.

23436 If *len* is 0 and *buf* is a null pointer, then *confstr()* shall still return the integer value as defined
 23437 below, but shall not return a string. If *len* is 0 but *buf* is not a null pointer, the result is

23438 unspecified.

23439 After a call to:

23440 `confstr(_CS_V7_ENV, buf, sizeof(buf))`

23441 the string stored in *buf* will contain the <space>-separated list of variable=value environment
23442 variable pairs required by the implementation to create a conforming environment, as described
23443 in the implementations' conformance documentation.

23444 If the implementation supports the POSIX shell option, the string stored in *buf* after a call to:

23445 `confstr(_CS_PATH, buf, sizeof(buf))`

23446 can be used as a value of the *PATH* environment variable that accesses all of the standard
23447 utilities of POSIX.1-2008, if the return value is less than or equal to *sizeof(buf)*.

23448 RETURN VALUE

23449 If *name* has a configuration-defined value, *confstr()* shall return the size of buffer that would be
23450 needed to hold the entire configuration-defined value including the terminating null. If this
23451 return value is greater than *len*, the string returned in *buf* is truncated.

23452 If *name* is invalid, *confstr()* shall return 0 and set *errno* to indicate the error.

23453 If *name* does not have a configuration-defined value, *confstr()* shall return 0 and leave *errno*
23454 unchanged.

23455 ERRORS

23456 The *confstr()* function shall fail if:

23457 [EINVAL] The value of the *name* argument is invalid.

23458 EXAMPLES

23459 None.

23460 APPLICATION USAGE

23461 An application can distinguish between an invalid *name* parameter value and one that
23462 corresponds to a configurable variable that has no configuration-defined value by checking if
23463 *errno* is modified. This mirrors the behavior of *sysconf()*.

23464 The original need for this function was to provide a way of finding the configuration-defined
23465 default value for the environment variable *PATH*. Since *PATH* can be modified by the user to
23466 include directories that could contain utilities replacing the standard utilities in the Shell and
23467 Utilities volume of POSIX.1-2008, applications need a way to determine the system-supplied
23468 *PATH* environment variable value that contains the correct search path for the standard utilities.

23469 An application could use:

23470 `confstr(name, (char *)NULL, (size_t)0)`

23471 to find out how big a buffer is needed for the string value; use *malloc()* to allocate a buffer to
23472 hold the string; and call *confstr()* again to get the string. Alternately, it could allocate a fixed,
23473 static buffer that is big enough to hold most answers (perhaps 512 or 1024 bytes), but then use
23474 *malloc()* to allocate a larger buffer if it finds that this is too small.

23475 RATIONALE

23476 Application developers can normally determine any configuration variable by means of reading
23477 from the stream opened by a call to:

23478 `popen("command -p getconf variable", "r");`

23479 The *confstr()* function with a *name* argument of *_CS_PATH* returns a string that can be used as a

23480 *PATH* environment variable setting that will reference the standard shell and utilities as
 23481 described in the Shell and Utilities volume of POSIX.1-2008.

23482 The *confstr()* function copies the returned string into a buffer supplied by the application instead
 23483 of returning a pointer to a string. This allows a cleaner function in some implementations (such
 23484 as those with lightweight threads) and resolves questions about when the application must copy
 23485 the string returned.

23486 FUTURE DIRECTIONS

23487 None.

23488 SEE ALSO

23489 *exec*, *fpathconf()*, *sysconf()*

23490 XBD <*unistd.h*>

23491 XCU *c99*

23492 CHANGE HISTORY

23493 First released in Issue 4. Derived from the ISO POSIX-2 standard.

23494 Issue 5

23495 A table indicating the permissible values of *name* is added to the DESCRIPTION. All those
 23496 marked EX are new in this version.

23497 Issue 6

23498 The Open Group Corrigendum U033/7 is applied. The return value for the case returning the
 23499 size of the buffer now explicitly states that this includes the terminating null.

23500 The following new requirements on POSIX implementations derive from alignment with the
 23501 Single UNIX Specification:

- 23502 • The DESCRIPTION is updated with new arguments which can be used to determine
- 23503 configuration strings for C compiler flags, linker/loader flags, and libraries for each
- 23504 different supported programming environment. This is a change to support data size
- 23505 neutrality.

23506 The following changes were made to align with the IEEE P1003.1a draft standard:

- 23507 • The DESCRIPTION is updated to include text describing how *_CS_PATH* can be used to
- 23508 obtain a *PATH* to access the standard utilities.

23509 The macros associated with the *c89* programming models are marked LEGACY and new
 23510 equivalent macros associated with *c99* are introduced.

23511 Issue 7

23512 Austin Group Interpretation 1003.1-2001 #047 is applied, adding the *_CS_V7_ENV* variable.

23513 Austin Group Interpretations 1003.1-2001 #166 is applied to permit an additional compiler flag
 23514 to enable threads.

23515 The V6 variables for the supported programming environments are marked obsolescent.

23516 The variables for the supported programming environments are updated to be V7.

23517 The LEGACY variables and obsolescent values are removed.

23518 **NAME**

23519 conj, conjf, conjl — complex conjugate functions

23520 **SYNOPSIS**23521 `#include <complex.h>`23522 `double complex conj(double complex z);`23523 `float complex conjf(float complex z);`23524 `long double complex conjl(long double complex z);`23525 **DESCRIPTION**

23526 CX The functionality described on this reference page is aligned with the ISO C standard. Any
23527 conflict between the requirements described here and the ISO C standard is unintentional. This
23528 volume of POSIX.1-2008 defers to the ISO C standard.

23529 These functions shall compute the complex conjugate of *z*, by reversing the sign of its imaginary
23530 part.

23531 **RETURN VALUE**

23532 These functions return the complex conjugate value.

23533 **ERRORS**

23534 No errors are defined.

23535 **EXAMPLES**

23536 None.

23537 **APPLICATION USAGE**

23538 None.

23539 **RATIONALE**

23540 None.

23541 **FUTURE DIRECTIONS**

23542 None.

23543 **SEE ALSO**23544 *carg()*, *cimag()*, *cproj()*, *creal()*23545 XBD **<complex.h>**23546 **CHANGE HISTORY**

23547 First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

NAME

connect — connect a socket

SYNOPSIS

```
#include <sys/socket.h>

int connect(int socket, const struct sockaddr *address,
            socklen_t address_len);
```

DESCRIPTION

The *connect()* function shall attempt to make a connection on a connection-mode socket or to set or reset the peer address of a connectionless-mode socket. The function takes the following arguments:

<i>socket</i>	Specifies the file descriptor associated with the socket.
<i>address</i>	Points to a sockaddr structure containing the peer address. The length and format of the address depend on the address family of the socket.
<i>address_len</i>	Specifies the length of the sockaddr structure pointed to by the <i>address</i> argument.

If the socket has not already been bound to a local address, *connect()* shall bind it to an address which, unless the socket's address family is AF_UNIX, is an unused local address.

If the initiating socket is not connection-mode, then *connect()* shall set the socket's peer address, and no connection is made. For SOCK_DGRAM sockets, the peer address identifies where all datagrams are sent on subsequent *send()* functions, and limits the remote sender for subsequent *recv()* functions. If the *sa_family* member of *address* is AF_UNSPEC, the socket's peer address shall be reset. Note that despite no connection being made, the term “connected” is used to describe a connectionless-mode socket for which a peer address has been set.

If the initiating socket is connection-mode, then *connect()* shall attempt to establish a connection to the address specified by the *address* argument. If the connection cannot be established immediately and O_NONBLOCK is not set for the file descriptor for the socket, *connect()* shall block for up to an unspecified timeout interval until the connection is established. If the timeout interval expires before the connection is established, *connect()* shall fail and the connection attempt shall be aborted. If *connect()* is interrupted by a signal that is caught while blocked waiting to establish a connection, *connect()* shall fail and set *errno* to [EINTR], but the connection request shall not be aborted, and the connection shall be established asynchronously.

If the connection cannot be established immediately and O_NONBLOCK is set for the file descriptor for the socket, *connect()* shall fail and set *errno* to [EINPROGRESS], but the connection request shall not be aborted, and the connection shall be established asynchronously. Subsequent calls to *connect()* for the same socket, before the connection is established, shall fail and set *errno* to [EALREADY].

When the connection has been established asynchronously, *pselect()*, *select()*, and *poll()* shall indicate that the file descriptor for the socket is ready for writing.

The socket in use may require the process to have appropriate privileges to use the *connect()* function.

RETURN VALUE

Upon successful completion, *connect()* shall return 0; otherwise, -1 shall be returned and *errno* set to indicate the error.

23591 **ERRORS**23592 The *connect()* function shall fail if:

23593 [EADDRNOTAVAIL]

23594 The specified address is not available from the local machine.

23595 [EAFNOSUPPORT]

23596 The specified address is not a valid address for the address family of the
23597 specified socket.

23598 [EALREADY] A connection request is already in progress for the specified socket.

23599 [EBADF] The *socket* argument is not a valid file descriptor.

23600 [ECONNREFUSED]

23601 The target address was not listening for connections or refused the connection
23602 request.23603 [EINPROGRESS] O_NONBLOCK is set for the file descriptor for the socket and the connection
23604 cannot be immediately established; the connection shall be established
23605 asynchronously.23606 [EINTR] The attempt to establish a connection was interrupted by delivery of a signal
23607 that was caught; the connection shall be established asynchronously.

23608 [EISCONN] The specified socket is connection-mode and is already connected.

23609 [ENETUNREACH]

23610 No route to the network is present.

23611 [ENOTSOCK] The *socket* argument does not refer to a socket.23612 [EPROTOTYPE] The specified address has a different type than the socket bound to the
23613 specified peer address.

23614 [ETIMEDOUT] The attempt to connect timed out before a connection was made.

23615 If the address family of the socket is AF_UNIX, then *connect()* shall fail if:

23616 [EIO] An I/O error occurred while reading from or writing to the file system.

23617 [ELOOP] A loop exists in symbolic links encountered during resolution of the pathname
23618 in *address*.

23619 [ENAMETOOLONG]

23620 The length of a component of a pathname is longer than {NAME_MAX}.

23621 [ENOENT] A component of the pathname does not name an existing file or the pathname
23622 is an empty string.23623 [ENOTDIR] A component of the path prefix of the pathname in *address* names an existing |
23624 file that is neither a directory nor a symbolic link to a directory, or the |
23625 pathname in *address* contains at least one non-*<slash>* character and ends with
23626 one or more trailing *<slash>* characters and the last pathname component
23627 names an existing file that is neither a directory nor a symbolic link to a
23628 directory.23629 The *connect()* function may fail if:23630 [EACCES] Search permission is denied for a component of the path prefix; or write access
23631 to the named socket is denied.

- 23632 [EADDRINUSE] Attempt to establish a connection that uses addresses that are already in use.
- 23633 [ECONNRESET] Remote host reset the connection request.
- 23634 [EHOSTUNREACH]
- 23635 The destination host cannot be reached (probably because the host is down or
- 23636 a remote router cannot reach it).
- 23637 [EINVAL] The *address_len* argument is not a valid length for the address family; or
- 23638 invalid address family in the **sockaddr** structure.
- 23639 [ELOOP] More than {SYMLOOP_MAX} symbolic links were encountered during
- 23640 resolution of the pathname in *address*.
- 23641 [ENAMETOOLONG]
- 23642 The length of a pathname exceeds {PATH_MAX}, or pathname resolution of a
- 23643 symbolic link produced an intermediate result with a length that exceeds
- 23644 {PATH_MAX}.
- 23645 [ENETDOWN] The local network interface used to reach the destination is down.
- 23646 [ENOBUFS] No buffer space is available.
- 23647 [EOPNOTSUPP] The socket is listening and cannot be connected.

EXAMPLES

23648 None.

APPLICATION USAGE

23651 If *connect()* fails, the state of the socket is unspecified. Conforming applications should close the

23652 file descriptor and create a new socket before attempting to reconnect.

RATIONALE

23653 None.

FUTURE DIRECTIONS

23655 None.

SEE ALSO

23658 *accept()*, *bind()*, *close()*, *getsockname()*, *poll()*, *pselect()*, *send()*, *shutdown()*, *socket()*

23659 XBD <sys/socket.h>

CHANGE HISTORY

23661 First released in Issue 6. Derived from the XNS, Issue 5.2 specification.

23662 The wording of the mandatory [ELOOP] error condition is updated, and a second optional

23663 [ELOOP] error condition is added.

Issue 7

23665 Austin Group Interpretation 1003.1-2001 #035 is applied, clarifying the description of connected

23666 sockets.

23667 Austin Group Interpretation 1003.1-2001 #143 is applied.

23668 Austin Group Interpretation 1003.1-2001 #188 is applied, changing the method used to reset a

23669 peer address for a datagram socket.

23670 SD5-XSH-ERN-185 is applied.

23671 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0062 [324] is applied.

+

23672 **NAME**

23673 copysign, copysignf, copysignl — number manipulation function

23674 **SYNOPSIS**

```
23675 #include <math.h>
23676 double copysign(double x, double y);
23677 float copysignf(float x, float y);
23678 long double copysignl(long double x, long double y);
```

23679 **DESCRIPTION**

23680 CX The functionality described on this reference page is aligned with the ISO C standard. Any
23681 conflict between the requirements described here and the ISO C standard is unintentional. This
23682 volume of POSIX.1-2008 defers to the ISO C standard.

23683 These functions shall produce a value with the magnitude of *x* and the sign of *y*. On
23684 implementations that represent a signed zero but do not treat negative zero consistently in
23685 arithmetic operations, these functions regard the sign of zero as positive.

23686 **RETURN VALUE**

23687 Upon successful completion, these functions shall return a value with the magnitude of *x* and
23688 the sign of *y*.

23689 **ERRORS**

23690 No errors are defined.

23691 **EXAMPLES**

23692 None.

23693 **APPLICATION USAGE**

23694 None.

23695 **RATIONALE**

23696 None.

23697 **FUTURE DIRECTIONS**

23698 None.

23699 **SEE ALSO**23700 [signbit\(\)](#)23701 XBD [<math.h>](#)23702 **CHANGE HISTORY**

23703 First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

23704 **NAME**

23705 cos, cosf, cosl — cosine function

23706 **SYNOPSIS**

```
23707 #include <math.h>
23708 double cos(double x);
23709 float cosf(float x);
23710 long double cosl(long double x);
```

23711 **DESCRIPTION**

23712 CX The functionality described on this reference page is aligned with the ISO C standard. Any
 23713 conflict between the requirements described here and the ISO C standard is unintentional. This
 23714 volume of POSIX.1-2008 defers to the ISO C standard.

23715 These functions shall compute the cosine of their argument x , measured in radians.

23716 An application wishing to check for error situations should set *errno* to zero and call
 23717 *feclearexcept*(FE_ALL_EXCEPT) before calling these functions. On return, if *errno* is non-zero or
 23718 *fetestexcept*(FE_INVALID | FE_DIVBYZERO | FE_OVERFLOW | FE_UNDERFLOW) is non-
 23719 zero, an error has occurred.

23720 **RETURN VALUE**

23721 Upon successful completion, these functions shall return the cosine of x .

23722 MX If x is NaN, a NaN shall be returned.

23723 If x is ± 0 , the value 1.0 shall be returned.

23724 If x is $\pm \text{Inf}$, a domain error shall occur, and a NaN shall be returned.

23725 **ERRORS**

23726 These functions shall fail if:

23727 MX **Domain Error** The x argument is $\pm \text{Inf}$.

23728 If the integer expression (*math_errhandling* & MATH_ERRNO) is non-zero,
 23729 then *errno* shall be set to [EDOM]. If the integer expression (*math_errhandling*
 23730 & MATH_ERREXCEPT) is non-zero, then the invalid floating-point exception
 23731 shall be raised.

23732 **EXAMPLES**23733 **Taking the Cosine of a 45-Degree Angle**

```
23734 #include <math.h>
23735 ...
23736 double radians = 45 * M_PI / 180;
23737 double result;
23738 ...
23739 result = cos(radians);
```

23740 **APPLICATION USAGE**

23741 These functions may lose accuracy when their argument is near an odd multiple of $\pi/2$ or is far
 23742 from 0.

23743 On error, the expressions (*math_errhandling* & MATH_ERRNO) and (*math_errhandling* &
 23744 MATH_ERREXCEPT) are independent of each other, but at least one of them must be non-zero.

23745 **RATIONALE**

23746 None.

23747 **FUTURE DIRECTIONS**

23748 None.

23749 **SEE ALSO**23750 *acos()*, *feclearexcept()*, *fetestexcept()*, *isnan()*, *sin()*, *tan()*23751 XBD Section 4.19 (on page 116), **<math.h>**23752 **CHANGE HISTORY**

23753 First released in Issue 1. Derived from Issue 1 of the SVID.

23754 **Issue 5**23755 The DESCRIPTION is updated to indicate how an application should check for an error. This
23756 text was previously published in the APPLICATION USAGE section.23757 **Issue 6**23758 The *cosf()* and *cosl()* functions are added for alignment with the ISO/IEC 9899:1999 standard.23759 The DESCRIPTION, RETURN VALUE, ERRORS, and APPLICATION USAGE sections are
23760 revised to align with the ISO/IEC 9899:1999 standard.23761 IEC 60559:1989 standard floating-point extensions over the ISO/IEC 9899:1999 standard are
23762 marked.23763 **Issue 7**

23764 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0063 [320] is applied.

+

23765 **NAME**

23766 cosh, coshf, coshl — hyperbolic cosine functions

23767 **SYNOPSIS**

```
23768 #include <math.h>
23769 double cosh(double x);
23770 float coshf(float x);
23771 long double coshl(long double x);
```

23772 **DESCRIPTION**

23773 CX The functionality described on this reference page is aligned with the ISO C standard. Any
 23774 conflict between the requirements described here and the ISO C standard is unintentional. This
 23775 volume of POSIX.1-2008 defers to the ISO C standard.

23776 These functions shall compute the hyperbolic cosine of their argument x .

23777 An application wishing to check for error situations should set *errno* to zero and call
 23778 *feclearexcept*(FE_ALL_EXCEPT) before calling these functions. On return, if *errno* is non-zero or
 23779 *fetestexcept*(FE_INVALID | FE_DIVBYZERO | FE_OVERFLOW | FE_UNDERFLOW) is non-
 23780 zero, an error has occurred.

23781 **RETURN VALUE**

23782 Upon successful completion, these functions shall return the hyperbolic cosine of x .

23783 If the correct value would cause overflow, a range error shall occur and *cosh()*, *coshf()*, and
 23784 *coshl()* shall return the value of the macro HUGE_VAL, HUGE_VALF, and HUGE_VALL,
 23785 respectively.

23786 MX If x is NaN, a NaN shall be returned.

23787 If x is ± 0 , the value 1.0 shall be returned.

23788 If x is $\pm\text{Inf}$, $+\text{Inf}$ shall be returned.

23789 **ERRORS**

23790 These functions shall fail if:

23791 Range Error The result would cause an overflow.

23792 If the integer expression (*math_errhandling* & MATH_ERRNO) is non-zero,
 23793 then *errno* shall be set to [ERANGE]. If the integer expression
 23794 (*math_errhandling* & MATH_ERREXCEPT) is non-zero, then the overflow
 23795 floating-point exception shall be raised.

23796 **EXAMPLES**

23797 None.

23798 **APPLICATION USAGE**

23799 On error, the expressions (*math_errhandling* & MATH_ERRNO) and (*math_errhandling* &
 23800 MATH_ERREXCEPT) are independent of each other, but at least one of them must be non-zero.

23801 For IEEE Std 754-1985 **double**, $710.5 < |x|$ implies that *cosh*(x) has overflowed.

23802 **RATIONALE**

23803 None.

23804 **FUTURE DIRECTIONS**

23805 None.

23806 **SEE ALSO**23807 *acosh()*, *feclearexcept()*, *fetestexcept()*, *isnan()*, *sinh()*, *tanh()*23808 XBD Section 4.19 (on page 116), **<math.h>**23809 **CHANGE HISTORY**

23810 First released in Issue 1. Derived from Issue 1 of the SVID.

23811 **Issue 5**23812 The DESCRIPTION is updated to indicate how an application should check for an error. This
23813 text was previously published in the APPLICATION USAGE section.23814 **Issue 6**23815 The *coshf()* and *coshl()* functions are added for alignment with the ISO/IEC 9899:1999 standard.23816 The DESCRIPTION, RETURN VALUE, ERRORS, and APPLICATION USAGE sections are
23817 revised to align with the ISO/IEC 9899:1999 standard.23818 IEC 60559:1989 standard floating-point extensions over the ISO/IEC 9899:1999 standard are
23819 marked.

23820 **NAME**

23821 cosl — cosine function

23822 **SYNOPSIS**

23823 #include <math.h>

23824 long double cosl(long double *x*);23825 **DESCRIPTION**23826 Refer to *cos()*.

23827 **NAME**

23828 cpow, cpowf, cpowl — complex power functions

23829 **SYNOPSIS**

23830 #include <complex.h>

23831 double complex cpow(double complex *x*, double complex *y*);23832 float complex cpowf(float complex *x*, float complex *y*);23833 long double complex cpowl(long double complex *x*,23834 long double complex *y*);23835 **DESCRIPTION**

23836 **CX** The functionality described on this reference page is aligned with the ISO C standard. Any
23837 conflict between the requirements described here and the ISO C standard is unintentional. This
23838 volume of POSIX.1-2008 defers to the ISO C standard.

23839 These functions shall compute the complex power function x^y , with a branch cut for the first
23840 parameter along the negative real axis.

23841 **RETURN VALUE**

23842 These functions shall return the complex power function value.

23843 **ERRORS**

23844 No errors are defined.

23845 **EXAMPLES**

23846 None.

23847 **APPLICATION USAGE**

23848 None.

23849 **RATIONALE**

23850 None.

23851 **FUTURE DIRECTIONS**

23852 None.

23853 **SEE ALSO**23854 *cabs()*, *csqrt()*

23855 XBD <complex.h>

23856 **CHANGE HISTORY**

23857 First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

23858 **NAME**

23859 cproj, cprojf, cprojl — complex projection functions

23860 **SYNOPSIS**

```
23861 #include <complex.h>
23862 double complex cproj(double complex z);
23863 float complex cprojf(float complex z);
23864 long double complex cprojl(long double complex z);
```

23865 **DESCRIPTION**

23866 CX The functionality described on this reference page is aligned with the ISO C standard. Any
 23867 conflict between the requirements described here and the ISO C standard is unintentional. This
 23868 volume of POSIX.1-2008 defers to the ISO C standard.

23869 These functions shall compute a projection of *z* onto the Riemann sphere: *z* projects to *z*, except
 23870 that all complex infinities (even those with one infinite part and one NaN part) project to
 23871 positive infinity on the real axis. If *z* has an infinite part, then *cproj(z)* shall be equivalent to:

```
23872 INFINITY + I * copysign(0.0, cimag(z))
```

23873 **RETURN VALUE**

23874 These functions shall return the value of the projection onto the Riemann sphere.

23875 **ERRORS**

23876 No errors are defined.

23877 **EXAMPLES**

23878 None.

23879 **APPLICATION USAGE**

23880 None.

23881 **RATIONALE**

23882 Two topologies are commonly used in complex mathematics: the complex plane with its
 23883 continuum of infinities, and the Riemann sphere with its single infinity. The complex plane is
 23884 better suited for transcendental functions, the Riemann sphere for algebraic functions. The
 23885 complex types with their multiplicity of infinities provide a useful (though imperfect) model for
 23886 the complex plane. The *cproj()* function helps model the Riemann sphere by mapping all
 23887 infinities to one, and should be used just before any operation, especially comparisons, that
 23888 might give spurious results for any of the other infinities. Note that a complex value with one
 23889 infinite part and one NaN part is regarded as an infinity, not a NaN, because if one part is
 23890 infinite, the complex value is infinite independent of the value of the other part. For the same
 23891 reason, *cabs()* returns an infinity if its argument has an infinite part and a NaN part.

23892 **FUTURE DIRECTIONS**

23893 None.

23894 **SEE ALSO**23895 *carg()*, *cimag()*, *conj()*, *creal()*23896 XBD **<complex.h>**23897 **CHANGE HISTORY**

23898 First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

23899 **NAME**

23900 creal, crealf, creall — complex real functions

23901 **SYNOPSIS**

```
23902     #include <complex.h>
23903     double creal(double complex z);
23904     float crealf(float complex z);
23905     long double creall(long double complex z);
```

23906 **DESCRIPTION**

23907 CX The functionality described on this reference page is aligned with the ISO C standard. Any
 23908 conflict between the requirements described here and the ISO C standard is unintentional. This
 23909 volume of POSIX.1-2008 defers to the ISO C standard.

23910 These functions shall compute the real part of *z*.23911 **RETURN VALUE**

23912 These functions shall return the real part value.

23913 **ERRORS**

23914 No errors are defined.

23915 **EXAMPLES**

23916 None.

23917 **APPLICATION USAGE**23918 For a variable *z* of type **complex**:23919 `z == creal(z) + cimag(z)*I`23920 **RATIONALE**

23921 None.

23922 **FUTURE DIRECTIONS**

23923 None.

23924 **SEE ALSO**23925 *carg()*, *cimag()*, *conj()*, *cproj()*23926 XBD **<complex.h>**23927 **CHANGE HISTORY**

23928 First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

23929 **NAME**

23930 creat — create a new file or rewrite an existing one

23931 **SYNOPSIS**23932 OH `#include <sys/stat.h>`23933 `#include <fcntl.h>`23934 `int creat(const char *path, mode_t mode);`23935 **DESCRIPTION**23936 The *creat()* function shall behave as if it is implemented as follows:23937 `int creat(const char *path, mode_t mode)`23938 `{`23939 `return open(path, O_WRONLY|O_CREAT|O_TRUNC, mode);`23940 `}`23941 **RETURN VALUE**23942 Refer to *open()*.23943 **ERRORS**23944 Refer to *open()*.23945 **EXAMPLES**23946 **Creating a File**

23947 The following example creates the file **/tmp/file** with read and write permissions for the file
 23948 owner and read permission for group and others. The resulting file descriptor is assigned to the
 23949 *fd* variable.

23950 `#include <fcntl.h>`23951 `...`23952 `int fd;`23953 `mode_t mode = S_IRUSR | S_IWUSR | S_IRGRP | S_IROTH;`23954 `char *pathname = "/tmp/file";`23955 `...`23956 `fd = creat(pathname, mode);`23957 `...`23958 **APPLICATION USAGE**

23959 None.

23960 **RATIONALE**

23961 The *creat()* function is redundant. Its services are also provided by the *open()* function. It has
 23962 been included primarily for historical purposes since many existing applications depend on it. It
 23963 is best considered a part of the C binding rather than a function that should be provided in other
 23964 languages.

23965 **FUTURE DIRECTIONS**

23966 None.

23967 **SEE ALSO**23968 *mknod()*, *open()*23969 XBD *<fcntl.h>*, *<sys/stat.h>*, *<sys/types.h>*

23970 **CHANGE HISTORY**

23971 First released in Issue 1. Derived from Issue 1 of the SVID.

23972 **Issue 6**23973 In the SYNOPSIS, the optional include of the **<sys/types.h>** header is removed.23974 The following new requirements on POSIX implementations derive from alignment with the
23975 Single UNIX Specification:

- 23976
- The requirement to include **<sys/types.h>** has been removed. Although **<sys/types.h>** was
-
- 23977 required for conforming implementations of previous POSIX specifications, it was not
-
- 23978 required for UNIX applications.

23979 **Issue 7**

23980 SD5-XSH-ERN-186 is applied.

23981 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0064 [291] is applied.

+

23982 **NAME**23983 `crypt` — string encoding function (**CRYPT**)23984 **SYNOPSIS**

```
23985 XSI      #include <unistd.h>
23986         char *crypt(const char *key, const char *salt);
```

23987 **DESCRIPTION**23988 The `crypt()` function is a string encoding function. The algorithm is implementation-defined.

23989 The *key* argument points to a string to be encoded. The *salt* argument shall be a string of at least
 23990 two bytes in length not including the null character chosen from the set:

```
23991 a b c d e f g h i j k l m n o p q r s t u v w x y z
23992 A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
23993 0 1 2 3 4 5 6 7 8 9 . /
```

23994 The first two bytes of this string may be used to perturb the encoding algorithm.

23995 The return value of `crypt()` points to static data that is overwritten by each call.23996 The `crypt()` function need not be thread-safe.23997 **RETURN VALUE**

23998 Upon successful completion, `crypt()` shall return a pointer to the encoded string. The first two
 23999 bytes of the returned value shall be those of the *salt* argument. Otherwise, it shall return a null
 24000 pointer and set *errno* to indicate the error.

24001 **ERRORS**24002 The `crypt()` function shall fail if:

24003 [ENOSYS] The functionality is not supported on this implementation.

24004 **EXAMPLES**24005 **Encoding Passwords**

24006 The following example finds a user database entry matching a particular user name and changes
 24007 the current password to a new password. The `crypt()` function generates an encoded version of
 24008 each password. The first call to `crypt()` produces an encoded version of the old password; that
 24009 encoded password is then compared to the password stored in the user database. The second
 24010 call to `crypt()` encodes the new password before it is stored.

24011 The `putpwent()` function, used in the following example, is not part of POSIX.1-2008.

```
24012 #include <unistd.h>
24013 #include <pwd.h>
24014 #include <string.h>
24015 #include <stdio.h>
24016 ...
24017 int valid_change;
24018 int pfd; /* Integer for file descriptor returned by open(). */
24019 FILE *fpfd; /* File pointer for use in putpwent(). */
24020 struct passwd *p;
24021 char user[100];
24022 char oldpasswd[100];
24023 char newpasswd[100];
24024 char savepasswd[100];
```

```

24025     ...
24026     valid_change = 0;
24027     while ((p = getpwent()) != NULL) {
24028         /* Change entry if found. */
24029         if (strcmp(p->pw_name, user) == 0) {
24030             if (strcmp(p->pw_passwd, crypt(oldpasswd, p->pw_passwd)) == 0) {
24031                 strcpy(savepasswd, crypt(newpasswd, user));
24032                 p->pw_passwd = savepasswd;
24033                 valid_change = 1;
24034             }
24035             else {
24036                 fprintf(stderr, "Old password is not valid\n");
24037             }
24038         }
24039         /* Put passwd entry into ptmp. */
24040         putpwent(p, fpfd);
24041     }

```

APPLICATION USAGE

The values returned by this function need not be portable among XSI-conformant systems.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

encrypt(), *setkey()*

XBD <unistd.h>

CHANGE HISTORY

First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 5

Normative text previously in the APPLICATION USAGE section is moved to the DESCRIPTION.

Issue 7

Austin Group Interpretation 1003.1-2001 #156 is applied.

SD5-XSH-ERN-178 is applied.

24059 **NAME**

24060 csin, csinf, csinl — complex sine functions

24061 **SYNOPSIS**

24062 #include <complex.h>

24063 double complex csin(double complex z);

24064 float complex csinf(float complex z);

24065 long double complex csinl(long double complex z);

24066 **DESCRIPTION**

24067 CX The functionality described on this reference page is aligned with the ISO C standard. Any
24068 conflict between the requirements described here and the ISO C standard is unintentional. This
24069 volume of POSIX.1-2008 defers to the ISO C standard.

24070 These functions shall compute the complex sine of z.

24071 **RETURN VALUE**

24072 These functions shall return the complex sine value.

24073 **ERRORS**

24074 No errors are defined.

24075 **EXAMPLES**

24076 None.

24077 **APPLICATION USAGE**

24078 None.

24079 **RATIONALE**

24080 None.

24081 **FUTURE DIRECTIONS**

24082 None.

24083 **SEE ALSO**24084 [casin\(\)](#)24085 XBD [<complex.h>](#)24086 **CHANGE HISTORY**

24087 First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

24088 **NAME**

24089 csinh, csinhf, csinhl — complex hyperbolic sine functions

24090 **SYNOPSIS**

24091 #include <complex.h>

24092 double complex csinh(double complex z);

24093 float complex csinhf(float complex z);

24094 long double complex csinhl(long double complex z);

24095 **DESCRIPTION**

24096 CX The functionality described on this reference page is aligned with the ISO C standard. Any
24097 conflict between the requirements described here and the ISO C standard is unintentional. This
24098 volume of POSIX.1-2008 defers to the ISO C standard.

24099 These functions shall compute the complex hyperbolic sine of z.

24100 **RETURN VALUE**

24101 These functions shall return the complex hyperbolic sine value.

24102 **ERRORS**

24103 No errors are defined.

24104 **EXAMPLES**

24105 None.

24106 **APPLICATION USAGE**

24107 None.

24108 **RATIONALE**

24109 None.

24110 **FUTURE DIRECTIONS**

24111 None.

24112 **SEE ALSO**

24113 casinh()

24114 XBD <complex.h>

24115 **CHANGE HISTORY**

24116 First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

24117 **NAME**

24118 csinl — complex sine functions

24119 **SYNOPSIS**

24120 #include <complex.h>

24121 long double complex csinl(long double complex z);

24122 **DESCRIPTION**24123 Refer to *csin()*.

24124 **NAME**

24125 csqrt, csqrtf, csqrtl — complex square root functions

24126 **SYNOPSIS**

24127 #include <complex.h>

24128 double complex csqrt(double complex z);

24129 float complex csqrtf(float complex z);

24130 long double complex csqrtl(long double complex z);

24131 **DESCRIPTION**

24132 CX The functionality described on this reference page is aligned with the ISO C standard. Any
 24133 conflict between the requirements described here and the ISO C standard is unintentional. This
 24134 volume of POSIX.1-2008 defers to the ISO C standard.

24135 These functions shall compute the complex square root of *z*, with a branch cut along the negative
 24136 real axis.

24137 **RETURN VALUE**

24138 These functions shall return the complex square root value, in the range of the right half-plane
 24139 (including the imaginary axis).

24140 **ERRORS**

24141 No errors are defined.

24142 **EXAMPLES**

24143 None.

24144 **APPLICATION USAGE**

24145 None.

24146 **RATIONALE**

24147 None.

24148 **FUTURE DIRECTIONS**

24149 None.

24150 **SEE ALSO**24151 *cabs()*, *cpow()*

24152 XBD <complex.h>

24153 **CHANGE HISTORY**

24154 First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

24155 **NAME**

24156 ctan, ctanf, ctanl — complex tangent functions

24157 **SYNOPSIS**

24158 #include <complex.h>

24159 double complex ctan(double complex z);

24160 float complex ctanf(float complex z);

24161 long double complex ctanl(long double complex z);

24162 **DESCRIPTION**

24163 CX The functionality described on this reference page is aligned with the ISO C standard. Any
24164 conflict between the requirements described here and the ISO C standard is unintentional. This
24165 volume of POSIX.1-2008 defers to the ISO C standard.

24166 These functions shall compute the complex tangent of z.

24167 **RETURN VALUE**

24168 These functions shall return the complex tangent value.

24169 **ERRORS**

24170 No errors are defined.

24171 **EXAMPLES**

24172 None.

24173 **APPLICATION USAGE**

24174 None.

24175 **RATIONALE**

24176 None.

24177 **FUTURE DIRECTIONS**

24178 None.

24179 **SEE ALSO**

24180 catan()

24181 XBD <complex.h>

24182 **CHANGE HISTORY**

24183 First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

24184 **NAME**

24185 ctanh, ctanhf, ctanhl — complex hyperbolic tangent functions

24186 **SYNOPSIS**

24187 #include <complex.h>

24188 double complex ctanh(double complex z);

24189 float complex ctanhf(float complex z);

24190 long double complex ctanhl(long double complex z);

24191 **DESCRIPTION**

24192 CX The functionality described on this reference page is aligned with the ISO C standard. Any
24193 conflict between the requirements described here and the ISO C standard is unintentional. This
24194 volume of POSIX.1-2008 defers to the ISO C standard.

24195 These functions shall compute the complex hyperbolic tangent of z.

24196 **RETURN VALUE**

24197 These functions shall return the complex hyperbolic tangent value.

24198 **ERRORS**

24199 No errors are defined.

24200 **EXAMPLES**

24201 None.

24202 **APPLICATION USAGE**

24203 None.

24204 **RATIONALE**

24205 None.

24206 **FUTURE DIRECTIONS**

24207 None.

24208 **SEE ALSO**

24209 catanh()

24210 XBD <complex.h>

24211 **CHANGE HISTORY**

24212 First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

24213 **NAME**

24214 ctanl — complex tangent functions

24215 **SYNOPSIS**

24216 #include <complex.h>

24217 long double complex ctanl(long double complex z);

24218 **DESCRIPTION**24219 Refer to *ctan()*.

NAME

ctermid — generate a pathname for the controlling terminal

SYNOPSIS

```
CX      #include <stdio.h>
      char *ctermid(char *s);
```

DESCRIPTION

The *ctermid()* function shall generate a string that, when used as a pathname, refers to the current controlling terminal for the current process. If *ctermid()* returns a pathname, access to the file is not guaranteed.

The *ctermid()* function need not be thread-safe if called with a NULL parameter.

RETURN VALUE

If *s* is a null pointer, the string shall be generated in an area that may be static, the address of which shall be returned. The application shall not modify the string returned. The returned pointer might be invalidated or the string content might be overwritten by a subsequent call to *ctermid()*. If *s* is not a null pointer, *s* is assumed to point to a character array of at least `L_ctermid` bytes; the string is placed in this array and the value of *s* shall be returned. The symbolic constant `L_ctermid` is defined in `<stdio.h>`, and shall have a value greater than 0.

The *ctermid()* function shall return an empty string if the pathname that would refer to the controlling terminal cannot be determined, or if the function is unsuccessful.

ERRORS

No errors are defined.

EXAMPLES**Determining the Controlling Terminal for the Current Process**

The following example returns a pointer to a string that identifies the controlling terminal for the current process. The pathname for the terminal is stored in the array pointed to by the *ptr* argument, which has a size of `L_ctermid` bytes, as indicated by the *term* argument.

```
#include <stdio.h>
...
char term[L_ctermid];
char *ptr;

ptr = ctermid(term);
```

APPLICATION USAGE

The difference between *ctermid()* and *ttyname()* is that *ttyname()* must be handed a file descriptor and return a path of the terminal associated with that file descriptor, while *ctermid()* returns a string (such as `"/dev/tty"`) that refers to the current controlling terminal if used as a pathname.

RATIONALE

`L_ctermid` must be defined appropriately for a given implementation and must be greater than zero so that array declarations using it are accepted by the compiler. The value includes the terminating null byte.

Conforming applications that use multiple threads cannot call *ctermid()* with NULL as the parameter. If *s* is not NULL, the *ctermid()* function generates a string that, when used as a pathname, refers to the current controlling terminal for the current process. If *s* is NULL, the

24263 return value of *ctermid()* is undefined.

24264 There is no additional burden on the programmer—changing to use a hypothetical thread-safe
24265 version of *ctermid()* along with allocating a buffer is more of a burden than merely allocating a
24266 buffer. Application code should not assume that the returned string is short, as some
24267 implementations have more than two pathname components before reaching a logical device
24268 name.

24269 **FUTURE DIRECTIONS**

24270 None.

24271 **SEE ALSO**

24272 *ttynname()*

24273 XBD <stdio.h>

24274 **CHANGE HISTORY**

24275 First released in Issue 1. Derived from Issue 1 of the SVID.

24276 **Issue 5**

24277 The DESCRIPTION is updated for alignment with the POSIX Threads Extension.

24278 **Issue 6**

24279 The normative text is updated to avoid use of the term “must” for application requirements.

24280 **Issue 7**

24281 Austin Group Interpretation 1003.1-2001 #148 is applied, updating the RATIONALE.

24282 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0065 [75,428] is applied.

+

24283 **NAME**

24284 ctime, ctime_r — convert a time value to a date and time string

24285 **SYNOPSIS**

```

24286 OB      #include <time.h>
24287          char *ctime(const time_t *clock);
24288 OB CX    char *ctime_r(const time_t *clock, char *buf);

```

24289 **DESCRIPTION**

24290 CX For *ctime()*: The functionality described on this reference page is aligned with the ISO C
 24291 standard. Any conflict between the requirements described here and the ISO C standard is
 24292 unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

24293 The *ctime()* function shall convert the time pointed to by *clock*, representing time in seconds
 24294 since the Epoch, to local time in the form of a string. It shall be equivalent to:

```
24295 asctime(localtime(clock))
```

24296 CX The *asctime()*, *ctime()*, *gmtime()*, and *localtime()* functions shall return values in one of two static
 24297 objects: a broken-down time structure and an array of **char**. Execution of any of the functions
 24298 may overwrite the information returned in either of these objects by any of the other functions.

24299 The *ctime()* function need not be thread-safe.

24300 The *ctime_r()* function shall convert the calendar time pointed to by *clock* to local time in exactly
 24301 the same form as *ctime()* and put the string into the array pointed to by *buf* (which shall be at
 24302 least 26 bytes in size) and return *buf*.

24303 Unlike *ctime()*, the *ctime_r()* function is not required to set *tzname*. If *ctime_r()* does not set +
 24304 *tzname*, it shall not set *daylight* and shall not set *timezone*.

24305 **RETURN VALUE**

24306 The *ctime()* function shall return the pointer returned by *asctime()* with that broken-down time
 24307 as an argument.

24308 CX Upon successful completion, *ctime_r()* shall return a pointer to the string pointed to by *buf*.
 24309 When an error is encountered, a null pointer shall be returned.

24310 **ERRORS**

24311 No errors are defined.

24312 **EXAMPLES**

24313 None.

24314 **APPLICATION USAGE**

24315 These functions are included only for compatibility with older implementations. They have
 24316 undefined behavior if the resulting string would be too long, so the use of these functions
 24317 should be discouraged. On implementations that do not detect output string length overflow, it
 24318 is possible to overflow the output buffers in such a way as to cause applications to fail, or
 24319 possible system security violations. Also, these functions do not support localized date and time
 24320 formats. To avoid these problems, applications should use *strftime()* to generate strings from
 24321 broken-down times.

24322 Values for the broken-down time structure can be obtained by calling *gmtime()* or *localtime()*.

24323 The *ctime_r()* function is thread-safe and shall return values in a user-supplied buffer instead of
 24324 possibly using a static data area that may be overwritten by each call.

24325 Attempts to use *ctime()* or *ctime_r()* for times before the Epoch or for times beyond the year 9999

24326 produce undefined results. Refer to *asctime()* (on page 595).

24327 RATIONALE

24328 The standard developers decided to mark the *ctime()* and *ctime_r()* functions obsolescent even
 24329 though they are in the ISO C standard due to the possibility of buffer overflow. The ISO C
 24330 standard also provides the *strftime()* function which can be used to avoid these problems.

24331 FUTURE DIRECTIONS

24332 These functions may be removed in a future version.

24333 SEE ALSO

24334 *asctime()*, *clock()*, *difftime()*, *gmtime()*, *localtime()*, *mktime()*, *strftime()*, *strptime()*, *time()*, *utime()*

24335 XBD **<time.h>**

24336 CHANGE HISTORY

24337 First released in Issue 1. Derived from Issue 1 of the SVID.

24338 Issue 5

24339 Normative text previously in the APPLICATION USAGE section is moved to the
 24340 DESCRIPTION.

24341 The *ctime_r()* function is included for alignment with the POSIX Threads Extension.

24342 A note indicating that the *ctime()* function need not be reentrant is added to the DESCRIPTION.

24343 Issue 6

24344 Extensions beyond the ISO C standard are marked.

24345 In the DESCRIPTION, the note about reentrancy is expanded to cover thread-safety.

24346 The APPLICATION USAGE section is updated to include a note on the thread-safe function and
 24347 its avoidance of possibly using a static data area.

24348 Issue 7

24349 Austin Group Interpretation 1003.1-2001 #156 is applied.

24350 SD5-XSH-ERN-25 is applied, updating the APPLICATION USAGE.

24351 Austin Group Interpretation 1003.1-2001 #053 is applied, marking these functions obsolescent.

24352 The *ctime_r()* function is moved from the Thread-Safe Functions option to the Base.

24353 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0066 [321,428] is applied.

+

24354 **NAME**

24355 daylight — daylight savings time flag

24356 **SYNOPSIS**

```
24357 XSI      #include <time.h>  
24358          extern int daylight;
```

24359 **DESCRIPTION**

24360 Refer to *tzset()*.

NAME

dbm_clearerr, dbm_close, dbm_delete, dbm_error, dbm_fetch, dbm_firstkey, dbm_nextkey, dbm_open, dbm_store — database functions

SYNOPSIS

```
XSI
#include <ndbm.h>

int dbm_clearerr(DBM *db);
void dbm_close(DBM *db);
int dbm_delete(DBM *db, datum key);
int dbm_error(DBM *db);
datum dbm_fetch(DBM *db, datum key);
datum dbm_firstkey(DBM *db);
datum dbm_nextkey(DBM *db);
DBM *dbm_open(const char *file, int open_flags, mode_t file_mode);
int dbm_store(DBM *db, datum key, datum content, int store_mode);
```

DESCRIPTION

These functions create, access, and modify a database.

A **datum** consists of at least two members, *dptr* and *dsize*. The *dptr* member points to an object that is *dsize* bytes in length. Arbitrary binary data, as well as character strings, may be stored in the object pointed to by *dptr*.

A database shall be stored in one or two files. When one file is used, the name of the database file shall be formed by appending the suffix **.db** to the *file* argument given to *dbm_open()*. When two files are used, the names of the database files shall be formed by appending the suffixes **.dir** and **.pag** respectively to the *file* argument.

The *dbm_open()* function shall open a database. The *file* argument to the function is the pathname of the database. The *open_flags* argument has the same meaning as the *flags* argument of *open()* except that a database opened for write-only access opens the files for read and write access and the behavior of the **O_APPEND** flag is unspecified. The *file_mode* argument has the same meaning as the third argument of *open()*.

The *dbm_open()* function need not accept pathnames longer than {PATH_MAX}-4 bytes (including the terminating null), or pathnames with a last component longer than {NAME_MAX}-4 bytes (excluding the terminating null).

The *dbm_close()* function shall close a database. The application shall ensure that argument *db* is a pointer to a **dbm** structure that has been returned from a call to *dbm_open()*.

These database functions shall support an internal block size large enough to support key/content pairs of at least 1 023 bytes.

The *dbm_fetch()* function shall read a record from a database. The argument *db* is a pointer to a database structure that has been returned from a call to *dbm_open()*. The argument *key* is a **datum** that has been initialized by the application to the value of the key that matches the key of the record the program is fetching.

The *dbm_store()* function shall write a record to a database. The argument *db* is a pointer to a database structure that has been returned from a call to *dbm_open()*. The argument *key* is a **datum** that has been initialized by the application to the value of the key that identifies (for subsequent reading, writing, or deleting) the record the application is writing. The argument *content* is a **datum** that has been initialized by the application to the value of the record the program is writing. The argument *store_mode* controls whether *dbm_store()* replaces any pre-existing record that has the same key that is specified by the *key* argument. The application shall

set *store_mode* to either DBM_INSERT or DBM_REPLACE. If the database contains a record that matches the *key* argument and *store_mode* is DBM_REPLACE, the existing record shall be replaced with the new record. If the database contains a record that matches the *key* argument and *store_mode* is DBM_INSERT, the existing record shall be left unchanged and the new record ignored. If the database does not contain a record that matches the *key* argument and *store_mode* is either DBM_INSERT or DBM_REPLACE, the new record shall be inserted in the database.

If the sum of a key/content pair exceeds the internal block size, the result is unspecified. Moreover, the application shall ensure that all key/content pairs that hash together fit on a single block. The *dbm_store()* function shall return an error in the event that a disk block fills with inseparable data.

The *dbm_delete()* function shall delete a record and its key from the database. The argument *db* is a pointer to a database structure that has been returned from a call to *dbm_open()*. The argument *key* is a **datum** that has been initialized by the application to the value of the key that identifies the record the program is deleting.

The *dbm_firstkey()* function shall return the first key in the database. The argument *db* is a pointer to a database structure that has been returned from a call to *dbm_open()*.

The *dbm_nextkey()* function shall return the next key in the database. The argument *db* is a pointer to a database structure that has been returned from a call to *dbm_open()*. The application shall ensure that the *dbm_firstkey()* function is called before calling *dbm_nextkey()*. Subsequent calls to *dbm_nextkey()* return the next key until all of the keys in the database have been returned.

The *dbm_error()* function shall return the error condition of the database. The argument *db* is a pointer to a database structure that has been returned from a call to *dbm_open()*.

The *dbm_clearerr()* function shall clear the error condition of the database. The argument *db* is a pointer to a database structure that has been returned from a call to *dbm_open()*.

The *dptr* pointers returned by these functions may point into static storage that may be changed by subsequent calls.

These functions need not be thread-safe.

RETURN VALUE

The *dbm_store()* and *dbm_delete()* functions shall return 0 when they succeed and a negative value when they fail.

The *dbm_store()* function shall return 1 if it is called with a *flags* value of DBM_INSERT and the function finds an existing record with the same key.

The *dbm_error()* function shall return 0 if the error condition is not set and return a non-zero value if the error condition is set.

The return value of *dbm_clearerr()* is unspecified.

The *dbm_firstkey()* and *dbm_nextkey()* functions shall return a key **datum**. When the end of the database is reached, the *dptr* member of the key is a null pointer. If an error is detected, the *dptr* member of the key shall be a null pointer and the error condition of the database shall be set.

The *dbm_fetch()* function shall return a content **datum**. If no record in the database matches the key or if an error condition has been detected in the database, the *dptr* member of the content shall be a null pointer.

The *dbm_open()* function shall return a pointer to a database structure. If an error is detected during the operation, *dbm_open()* shall return a (DBM *)0.

ERRORS

No errors are defined.

EXAMPLES

None.

APPLICATION USAGE

The following code can be used to traverse the database:

```
for(key = dbm_firstkey(db); key.dptr != NULL; key = dbm_nextkey(db))
```

The *dbm_** functions provided in this library should not be confused in any way with those of a general-purpose database management system. These functions do not provide for multiple search keys per entry, they do not protect against multi-user access (in other words they do not lock records or files), and they do not provide the many other useful database functions that are found in more robust database management systems. Creating and updating databases by use of these functions is relatively slow because of data copies that occur upon hash collisions. These functions are useful for applications requiring fast lookup of relatively static information that is to be indexed by a single key.

Note that a strictly conforming application is extremely limited by these functions: since there is no way to determine that the keys in use do not all hash to the same value (although that would be rare), a strictly conforming application cannot be guaranteed that it can store more than one block's worth of data in the database. As long as a key collision does not occur, additional data may be stored, but because there is no way to determine whether an error is due to a key collision or some other error condition (*dbm_error()* being effectively a Boolean), once an error is detected, the application is effectively limited to guessing what the error might be if it wishes to continue using these functions.

The *dbm_delete()* function need not physically reclaim file space, although it does make it available for reuse by the database.

After calling *dbm_store()* or *dbm_delete()* during a pass through the keys by *dbm_firstkey()* and *dbm_nextkey()*, the application should reset the database by calling *dbm_firstkey()* before again calling *dbm_nextkey()*. The contents of these files are unspecified and may not be portable.

Applications should take care that database pathname arguments specified to *dbm_open()* are not prefixes of unrelated files. This might be done, for example, by placing databases in a separate directory.

Since some implementations use three characters for a suffix and others use four characters for a suffix, applications should ensure that the maximum portable pathname length passed to *dbm_open()* is no greater than {PATH_MAX}-4 bytes, with the last component of the pathname no greater than {NAME_MAX}-4 bytes.

RATIONALE

Previously the standard required the database to be stored in two files, one file being a directory containing a bitmap of keys and having **.dir** as its suffix. The second file containing all data and having **.pag** as its suffix. This has been changed not to specify the use of the files and to allow newer implementations of the Berkeley DB interface using a single file that have evolved while remaining compatible with the application programming interface. The standard developers considered removing the specific suffixes altogether but decided to retain them so as not to pollute the application file name space more than necessary and to allow for portable backups of the database.

24495 **FUTURE DIRECTIONS**

24496 None.

24497 **SEE ALSO**24498 *open()*

24499 XBD <ndbm.h>

24500 **CHANGE HISTORY**

24501 First released in Issue 4, Version 2.

24502 **Issue 5**

24503 Moved from X/OPEN UNIX extension to BASE.

24504 Normative text previously in the APPLICATION USAGE section is moved to the
24505 DESCRIPTION.

24506 A note indicating that these functions need not be reentrant is added to the DESCRIPTION.

24507 **Issue 6**

24508 The normative text is updated to avoid use of the term “must” for application requirements.

24509 **Issue 7**24510 Austin Group Interpretation 1003.1-2001 #042 is applied so that the DESCRIPTION permits
24511 newer implementations of the Berkeley DB interface.

24512 Austin Group Interpretation 1003.1-2001 #156 is applied.

24513 **NAME**

24514 difftime — compute the difference between two calendar time values

24515 **SYNOPSIS**

24516 #include <time.h>

24517 double difftime(time_t *time1*, time_t *time0*);24518 **DESCRIPTION**

24519 CX The functionality described on this reference page is aligned with the ISO C standard. Any
24520 conflict between the requirements described here and the ISO C standard is unintentional. This
24521 volume of POSIX.1-2008 defers to the ISO C standard.

24522 The *difftime()* function shall compute the difference between two calendar times (as returned by
24523 *time()*): *time1* − *time0*.

24524 **RETURN VALUE**24525 The *difftime()* function shall return the difference expressed in seconds as a type **double**.24526 **ERRORS**

24527 No errors are defined.

24528 **EXAMPLES**

24529 None.

24530 **APPLICATION USAGE**

24531 None.

24532 **RATIONALE**

24533 None.

24534 **FUTURE DIRECTIONS**

24535 None.

24536 **SEE ALSO**24537 [*asctime\(\)*](#), [*clock\(\)*](#), [*ctime\(\)*](#), [*gmtime\(\)*](#), [*localtime\(\)*](#), [*mktime\(\)*](#), [*strftime\(\)*](#), [*strptime\(\)*](#), [*time\(\)*](#), [*utime\(\)*](#)24538 XBD [**<time.h>**](#)24539 **CHANGE HISTORY**

24540 First released in Issue 4. Derived from the ISO C standard.

NAME

dirfd — extract the file descriptor used by a DIR stream

SYNOPSIS

```
#include <dirent.h>

int dirfd(DIR *dirp);
```

DESCRIPTION

The *dirfd()* function shall return a file descriptor referring to the same directory as the *dirp* argument. This file descriptor shall be closed by a call to *closedir()*. If any attempt is made to close the file descriptor, or to modify the state of the associated description, other than by means of *closedir()*, *readdir()*, *readdir_r()*, *rewinddir()*, or *seekdir()*, the behavior is undefined.

RETURN VALUE

Upon successful completion, the *dirfd()* function shall return an integer which contains a file descriptor for the stream pointed to by *dirp*. Otherwise, it shall return *-1* and may set *errno* to indicate the error.

ERRORS

The *dirfd()* function may fail if:

[EINVAL]	The <i>dirp</i> argument does not refer to a valid directory stream.
[ENOTSUP]	The implementation does not support the association of a file descriptor with a directory.

EXAMPLES

None.

APPLICATION USAGE

The *dirfd()* function is intended to be a mechanism by which an application may obtain a file descriptor to use for the *fchdir()* function.

RATIONALE

This interface was introduced because the Base Definitions volume of POSIX.1-2008 does not make public the **DIR** data structure. Applications tend to use the *fchdir()* function on the file descriptor returned by this interface, and this has proven useful for security reasons; in particular, it is a better technique than others where directory names might change.

The description uses the term “a file descriptor” rather than “the file descriptor”. The implication intended is that an implementation that does not use an *fd* for *opendir()* could still *open()* the directory to implement the *dirfd()* function. Such a descriptor must be closed later during a call to *closedir()*.

An implementation that does not support file descriptors referring to directories may fail with [ENOTSUP].

If it is necessary to allocate an *fd* to be returned by *dirfd()*, it should be done at the time of a call to *opendir()*.

FUTURE DIRECTIONS

None.

SEE ALSO

closedir(), *fchdir()*, *fdopendir()*, *fileno()*, *open()*, *readdir()*
XBD <dirent.h>

24583	CHANGE HISTORY
24584	First released in Issue 7.
24585	POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0067 [422] is applied.

+

24586 **NAME**

24587 dirname — report the parent directory name of a file pathname

24588 **SYNOPSIS**

```
24589 XSI      #include <libgen.h>
24590      char *dirname(char *path);
```

24591 **DESCRIPTION**

24592 The *dirname()* function shall take a pointer to a character string that contains a pathname, and
 24593 return a pointer to a string that is a pathname of the parent directory of that file. Trailing '/'
 24594 characters in the path are not counted as part of the path.

24595 If *path* does not contain a '/' , then *dirname()* shall return a pointer to the string ".". If *path* is a
 24596 null pointer or points to an empty string, *dirname()* shall return a pointer to the string ".".

24597 The *dirname()* function need not be thread-safe.

24598 **RETURN VALUE**

24599 The *dirname()* function shall return a pointer to a string that is the parent directory of *path*. If
 24600 *path* is a null pointer or points to an empty string, a pointer to a string "." is returned.

24601 The *dirname()* function may modify the string pointed to by *path*, and may return a pointer to
 24602 internal storage. The returned pointer might be invalidated or the storage might be overwritten
 24603 by a subsequent call to *dirname()*.

24604 **ERRORS**

24605 No errors are defined.

24606 **EXAMPLES**

24607 The following code fragment reads a pathname, changes the current working directory to the
 24608 parent directory, and opens the file.

```
24609      char *path = NULL, *pathcopy;
24610      size_t buflen = 0;
24611      ssize_t linelen = 0;
24612      int fd;

24613      linelen = getline(&path, &buflen, stdin);

24614      path[linelen-1] = 0;
24615      pathcopy = strdup(path);
24616      if (chdir(dirname(pathcopy)) < 0) {
24617          ...
24618      }
24619      if ((fd = open(basename(path), O_RDONLY)) >= 0) {
24620          ...
24621          close (fd);
24622      }
24623      ...
24624      free (pathcopy);
24625      free (path);
```


Sample Input and Output Strings for `dirname()`

In the following table, the input string is the value pointed to by *path*, and the output string is the return value of the *dirname()* function.

Input String	Output String
<code>"/usr/lib"</code>	<code>"/usr"</code>
<code>"/usr/"</code>	<code>"/"</code>
<code>"usr"</code>	<code>."</code>
<code>"/"</code>	<code>"/"</code>
<code>."</code>	<code>."</code>
<code>".."</code>	<code>."</code>

APPLICATION USAGE

The *dirname()* and *basename()* functions together yield a complete pathname. The expression *dirname(path)* obtains the pathname of the directory where *basename(path)* is found.

Since the meaning of the leading `"/"` is implementation-defined, *dirname("//foo")* may return either `"/"` or `"/"` (but nothing else).

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

basename()
XBD `<libgen.h>`

CHANGE HISTORY

First released in Issue 4, Version 2.

Issue 5

- Moved from X/OPEN UNIX extension to BASE.
- Normative text previously in the APPLICATION USAGE section is moved to the DESCRIPTION.
- A note indicating that this function need not be reentrant is added to the DESCRIPTION.

Issue 7

- Austin Group Interpretation 1003.1-2001 #156 is applied.
- The EXAMPLES section is revised.
- POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0068 [75] is applied.

24659 **NAME**24660 `div` — compute the quotient and remainder of an integer division24661 **SYNOPSIS**24662 `#include <stdlib.h>`24663 `div_t div(int numer, int denom);`24664 **DESCRIPTION**

24665 CX The functionality described on this reference page is aligned with the ISO C standard. Any
24666 conflict between the requirements described here and the ISO C standard is unintentional. This
24667 volume of POSIX.1-2008 defers to the ISO C standard.

24668 The `div()` function shall compute the quotient and remainder of the division of the numerator
24669 *numer* by the denominator *denom*. If the division is inexact, the resulting quotient is the integer
24670 of lesser magnitude that is the nearest to the algebraic quotient. If the result cannot be
24671 represented, the behavior is undefined; otherwise, *quot*denom+rem* shall equal *numer*.

24672 **RETURN VALUE**

24673 The `div()` function shall return a structure of type `div_t`, comprising both the quotient and the
24674 remainder. The structure includes the following members, in any order:

```
24675 int quot; /* quotient */
24676 int rem; /* remainder */
```

24677 **ERRORS**

24678 No errors are defined.

24679 **EXAMPLES**

24680 None.

24681 **APPLICATION USAGE**

24682 None.

24683 **RATIONALE**

24684 None.

24685 **FUTURE DIRECTIONS**

24686 None.

24687 **SEE ALSO**24688 [*ldiv\(\)*](#)24689 XBD [`<stdlib.h>`](#)24690 **CHANGE HISTORY**

24691 First released in Issue 4. Derived from the ISO C standard.

NAME

dlclose — close a symbol table handle

SYNOPSIS

```
#include <dlfcn.h>
```

```
int dlclose(void *handle);
```

DESCRIPTION

The *dlclose()* function shall inform the system that the symbol table handle specified by *handle* is no longer needed by the application.

An application writer may use *dlclose()* to make a statement of intent on the part of the process, but this statement does not create any requirement upon the implementation. When the symbol table handle is closed, the implementation may unload the executable object files that were loaded by *dlopen()* when the symbol table handle was opened and those that were loaded by *dlsym()* when using the symbol table handle identified by *handle*.

Once a symbol table handle has been closed, an application should assume that any symbols (function identifiers and data object identifiers) made visible using *handle*, are no longer available to the process.

Although a *dlclose()* operation is not required to remove any functions or data objects from the address space, neither is an implementation prohibited from doing so. The only restriction on such a removal is that no function nor data object shall be removed to which references have been relocated, until or unless all such references are removed. For instance, an executable object file that had been loaded with a *dlopen()* operation specifying the *RTLD_GLOBAL* flag might provide a target for dynamic relocations performed in the processing of other relocatable objects—in such environments, an application may assume that no relocation, once made, shall be undone or remade unless the executable object file containing the relocated object has itself been removed.

RETURN VALUE

If the referenced symbol table handle was successfully closed, *dlclose()* shall return 0. If *handle* does not refer to an open symbol table handle or if the symbol table handle could not be closed, *dlclose()* shall return a non-zero value. More detailed diagnostic information shall be available through *dLError()*.

ERRORS

No errors are defined.

EXAMPLES

The following example illustrates use of *dlopen()* and *dlclose()*:

```
#include <dlfcn.h>
int eret;
void *mylib;
...
/* Open a dynamic library and then close it ... */
mylib = dlopen("mylib.so", RTLD_LOCAL | RTLD_LAZY);
...
eret = dlclose(mylib);
...
```

APPLICATION USAGE

A conforming application should employ a symbol table handle returned from a *dlopen()* invocation only within a given scope bracketed by a *dlopen()* operation and the corresponding *dlclose()* operation. Implementations are free to use reference counting or other techniques such that multiple calls to *dlopen()* referencing the same executable object file may return a pointer to the same data object as the symbol table handle.

Implementations are also free to re-use a handle. For these reasons, the value of a handle must be treated as an opaque data type by the application, used only in calls to *dlsym()* and *dlclose()*.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

dlerror(), *dlopen()*, *dlsym()*

XBD <dlfcn.h>

CHANGE HISTORY

First released in Issue 5.

Issue 6

The DESCRIPTION is updated to say that the referenced object is closed “if this is the last reference to it”.

Issue 7

The *dlopen()* function is moved from the XSI option to Base.

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0069 [74] is applied.

+

24758 NAME

24759 **dlerror** — get diagnostic information

24760 SYNOPSIS

```
24761 #include <dlfcn.h>
24762 char *dlerror(void);
```

24763 DESCRIPTION

24764 The *dlerror()* function shall return a null-terminated character string (with no trailing
24765 <newline>) that describes the last error that occurred during dynamic linking processing. If no
24766 dynamic linking errors have occurred since the last invocation of *dlerror()*, *dlerror()* shall return
24767 NULL. Thus, invoking *dlerror()* a second time, immediately following a prior invocation, shall
24768 result in NULL being returned.

24769 It is implementation-defined whether or not the *dlerror()* function is thread-safe. A thread-safe
24770 implementation shall return only errors that occur on the current thread.

24771 RETURN VALUE

24772 If successful, *dlerror()* shall return a null-terminated character string; otherwise, NULL shall be
24773 returned.

24774 The application shall not modify the string returned. The returned pointer might be invalidated +
24775 or the string content might be overwritten by a subsequent call to *dlerror()* in the same thread (if +
24776 *dlerror()* is thread-safe) or in any thread (if *dlerror()* is not thread-safe).

24777 ERRORS

24778 No errors are defined.

24779 EXAMPLES

24780 The following example prints out the last dynamic linking error:

```
24781 ...
24782 #include <dlfcn.h>
24783 char *errstr;
24784 errstr = dlerror();
24785 if (errstr != NULL)
24786     printf ("A dynamic linking error occurred: (%s)\n", errstr);
24787 ...
```

24788 APPLICATION USAGE

24789 Depending on the application environment with respect to asynchronous execution events, such -
24790 as signals or other asynchronous computation sharing the address space, conforming
24791 applications should use a critical section to retrieve the error pointer and buffer.

24792 RATIONALE

24793 None.

24794 FUTURE DIRECTIONS

24795 None.

24796 SEE ALSO

24797 *dlclose()*, *dlopen()*, *dlsym()*

24798 XBD [<dlfcn.h>](#)

CHANGE HISTORY

24799 First released in Issue 5.

Issue 6

24801 A note indicating that this function need not be reentrant is added to the DESCRIPTION.

Issue 7

24803 Austin Group Interpretation 1003.1-2001 #156 is applied.

24804 The *dlerror()* function is moved from the XSI option to the Base.

24806 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0070 [75], XSH/TC1-2008/0071 [97], +
24807 and XSH/TC1-2008/0072 [133] are applied.

24808 **NAME**

24809 dlopen — open a symbol table handle |

24810 **SYNOPSIS**

24811 #include <dlfcn.h>

24812 void *dlopen(const char *file, int mode);

24813 **DESCRIPTION**

24814 The *dlopen()* function shall make the symbols (function identifiers and data object identifiers) in |
 24815 the executable object file specified by *file* available to the calling program. |

24816 The class of executable object files eligible for this operation and the manner of their |
 24817 construction are implementation-defined, though typically such files are shared libraries or |
 24818 programs. |

24819 Implementations may permit the construction of embedded dependencies in executable object |
 24820 files. In such cases, a *dlopen()* operation shall load those dependencies in addition to the |
 24821 executable object file specified by *file*. Implementations may also impose specific constraints on |
 24822 the construction of programs that can employ *dlopen()* and its related services. |

24823 A successful *dlopen()* shall return a symbol table handle which the caller may use on subsequent |
 24824 calls to *dlsym()* and *dlclose()*. |

24825 The value of this symbol table handle should not be interpreted in any way by the caller. |

24826 The *file* argument is used to construct a pathname to the executable object file. If *file* contains a |
 24827 <slash> character, the *file* argument is used as the pathname for the file. Otherwise, *file* is used in |
 24828 an implementation-defined manner to yield a pathname. |

24829 If *file* is a null pointer, *dlopen()* shall return a global symbol table handle for the currently |
 24830 running process image. This symbol table handle shall provide access to the symbols from an |
 24831 ordered set of executable object files consisting of the original program image file, any |
 24832 executable object files loaded at program start-up as specified by that process file (for example, |
 24833 shared libraries), and the set of executable object files loaded using *dlopen()* operations with the |
 24834 RTLD_GLOBAL flag. As the latter set of executable object files can change during execution, the |
 24835 set of symbols made available by this symbol table handle can also change dynamically. |

24836 Only a single copy of an executable object file shall be brought into the address space, even if |
 24837 *dlopen()* is invoked multiple times in reference to the executable object file, and even if different |
 24838 pathnames are used to reference the executable object file. |

24839 The *mode* parameter describes how *dlopen()* shall operate upon *file* with respect to the processing |
 24840 of relocations and the scope of visibility of the symbols provided within *file*. When an |
 24841 executable object file is brought into the address space of a process, it may contain references to |
 24842 symbols whose addresses are not known until the executable object file is loaded. |

24843 These references shall be relocated before the symbols can be accessed. The *mode* parameter |
 24844 governs when these relocations take place and may have the following values:

24845	RTLD_LAZY	Relocations shall be performed at an implementation-defined time, ranging
24846		from the time of the <i>dlopen()</i> call until the first reference to a given symbol
24847		occurs. Specifying RTLD_LAZY should improve performance on
24848		implementations supporting dynamic symbol binding since a process might
24849		not reference all of the symbols in an executable object file. And, for systems
24850		supporting dynamic symbol resolution for normal process execution, this
24851		behavior mimics the normal handling of process execution.

RTLD_NOW All necessary relocations shall be performed when the executable object file is first loaded. This may waste some processing if relocations are performed for symbols that are never referenced. This behavior may be useful for applications that need to know that all symbols referenced during execution will be available before *dlopen()* returns.

Any executable object file loaded by *dlopen()* that requires relocations against global symbols can reference the symbols in the original process image file, any executable object files loaded at program start-up, from the initial process image itself, from any other executable object file included in the same *dlopen()* invocation, and any executable object files that were loaded in any *dlopen()* invocation and which specified the RTLD_GLOBAL flag. To determine the scope of visibility for the symbols loaded with a *dlopen()* invocation, the *mode* parameter should be a bitwise-inclusive OR with one of the following values:

RTLD_GLOBAL The executable object file's symbols shall be made available for relocation processing of any other executable object file. In addition, symbol lookup using *dlopen(NULL,mode)* and an associated *dlsym()* allows executable object files loaded with this mode to be searched.

RTLD_LOCAL The executable object file's symbols shall not be made available for relocation processing of any other executable object file.

If neither RTLD_GLOBAL nor RTLD_LOCAL is specified, the default behavior is unspecified.

If an executable object file is specified in multiple *dlopen()* invocations, *mode* is interpreted at each invocation.

If RTLD_NOW has been specified, all relocations shall have been completed rendering further RTLD_NOW operations redundant and any further RTLD_LAZY operations irrelevant.

If RTLD_GLOBAL has been specified, the executable object file shall maintain the RTLD_GLOBAL status regardless of any previous or future specification of RTLD_LOCAL, as long as the executable object file remains in the address space (see *dlclose()*).

Symbols introduced into the process image through calls to *dlopen()* may be used in relocation activities. Symbols so introduced may duplicate symbols already defined by the program or previous *dlopen()* operations. To resolve the ambiguities such a situation might present, the resolution of a symbol reference to symbol definition is based on a symbol resolution order. Two such resolution orders are defined: load order and dependency order. Load order establishes an ordering among symbol definitions, such that the first definition loaded (including definitions from the process image file and any dependent executable object files loaded with it) has priority over executable object files added later (by *dlopen()*). Load ordering is used in relocation processing. Dependency ordering uses a breadth-first order starting with a given executable object file, then all of its dependencies, then any dependents of those, iterating until all dependencies are satisfied. With the exception of the global symbol table handle obtained via a *dlopen()* operation with a null pointer as the *file* argument, dependency ordering is used by the *dlsym()* function. Load ordering is used in *dlsym()* operations upon the global symbol table handle.

When an executable object file is first made accessible via *dlopen()*, it and its dependent executable object files are added in dependency order. Once all the executable object files are added, relocations are performed using load order. Note that if an executable object file or its dependencies had been previously loaded, the load and dependency orders may yield different resolutions.

The symbols introduced by *dlopen()* operations and available through *dlsym()* are at a minimum those which are exported as identifiers of global scope by the executable object file. Typically,

24899 such identifiers shall be those that were specified in (for example) C source code as having |
 24900 **extern** linkage. The precise manner in which an implementation constructs the set of exported |
 24901 symbols for an executable object file is implementation-defined.

24902 **RETURN VALUE**

24903 Upon successful completion, *dlopen()* shall return a symbol table handle. If *file* cannot be found, |
 24904 cannot be opened for reading, is not of an appropriate executable object file format for |
 24905 processing by *dlopen()*, or if an error occurs during the process of loading *file* or relocating its |
 24906 symbolic references, *dlopen()* shall return a null pointer. More detailed diagnostic information |
 24907 shall be available through *dlderror()*.

24908 **ERRORS**

24909 No errors are defined.

24910 **EXAMPLES**

24911 Refer to *dlsym()*.

24912 **APPLICATION USAGE**

24913 None.

24914 **RATIONALE**

24915 None.

24916 **FUTURE DIRECTIONS**

24917 None.

24918 **SEE ALSO**

24919 *dlclose()*, *dlderror()*, *dlsym()*

24920 XBD <dlfcn.h>

24921 **CHANGE HISTORY**

24922 First released in Issue 5.

24923 **Issue 6**

24924 IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/21 is applied, changing the default
 24925 behavior in the DESCRIPTION when neither RTLD_GLOBAL nor RTLD_LOCAL are specified
 24926 from implementation-defined to unspecified.

24927 **Issue 7**

24928 The *dlopen()* function is moved from the XSI option to the Base.

24929 The EXAMPLES section is updated to refer to *dlsym()*.

24930 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0073 [74] is applied.

+

24931 **NAME**

24932 dlsym — get the address of a symbol from a symbol table handle |

24933 **SYNOPSIS**

24934 #include <dlfcn.h>

24935 void *dlsym(void *restrict handle, const char *restrict name);

24936 **DESCRIPTION**

24937 The *dlsym()* function shall obtain the address of a symbol (a function identifier or a data object
 24938 identifier) defined in the symbol table identified by the *handle* argument. The *handle* argument is
 24939 a symbol table handle returned from a call to *dlopen()* (and which has not since been released by
 24940 a call to *dlclose()*), and *name* is the symbol's name as a character string. The return value from
 24941 *dlsym()*, cast to a pointer to the type of the named symbol, can be used to call (in the case of a
 24942 function) or access the contents of (in the case of a data object) the named symbol.

24943 The *dlsym()* function shall search for the named symbol in the symbol table referenced by *handle*.
 24944 If the symbol table was created with lazy loading (see *RTLD_LAZY* in *dlopen()*), load ordering
 24945 shall be used in *dlsym()* operations to relocate executable object files needed to resolve the
 24946 symbol. The symbol resolution algorithm used shall be dependency order as described in
 24947 *dlopen()*.

24948 The *RTLD_DEFAULT* and *RTLD_NEXT* symbolic constants (which may be defined in
 24949 <dlfcn.h>) are reserved for future use as special values that applications may be allowed to use
 24950 for *handle*.

24951 **RETURN VALUE**

24952 Upon successful completion, if *name* names a function identifier, *dlsym()* shall return the address +
 24953 of the function converted from type pointer to function to type pointer to **void**; otherwise, +
 24954 *dlsym()* shall return the address of the data object associated with the data object identifier +
 24955 named by *name* converted from a pointer to the type of the data object to a pointer to **void**. If
 24956 *handle* does not refer to a valid symbol table handle or if the symbol named by *name* cannot be
 24957 found in the symbol table associated with *handle*, *dlsym()* shall return a null pointer. |

24958 More detailed diagnostic information shall be available through *dlerror()*.24959 **ERRORS**

24960 No errors are defined.

24961 **EXAMPLES**

24962 The following example shows how *dlopen()* and *dlsym()* can be used to access either a function |
 24963 or a data object. For simplicity, error checking has been omitted.

```
24964 void *handle; |
24965 int (*fptr)(int), *iptr, result; |
24966 /* open the needed symbol table */ |
24967 handle = dlopen("/usr/home/me/libfoo.so", RTLD_LOCAL | RTLD_LAZY); |
24968 /* find the address of the function my_function */ |
24969 fptr = (int (*)(int))dlsym(handle, "my_function"); |
24970 /* find the address of the data object my_object */ |
24971 iptr = (int *)dlsym(handle, "my_OBJ"); |
24972 /* invoke my_function, passing the value of my_OBJ as the parameter */ |
24973 result = (*fptr)(*iptr); |
```

APPLICATION USAGE

The following special purpose values for *handle* are reserved for future use and have the indicated meanings:

RTLD_DEFAULT The identifier lookup happens in the normal global scope; that is, a search for an identifier using *handle* would find the same definition as a direct use of this identifier in the program code.

RTLD_NEXT Specifies the next executable object file after this one that defines *name*. This one refers to the executable object file containing the invocation of *dlsym()*. The next executable object file is the one found upon the application of a load order symbol resolution algorithm (see *dlopen()*). The next symbol is either one of global scope (because it was introduced as part of the original process image or because it was added with a *dlopen()* operation including the **RTLD_GLOBAL** flag), or is in an executable object file that was included in the same *dlopen()* operation that loaded this one.

The **RTLD_NEXT** flag is useful to navigate an intentionally created hierarchy of multiply-defined symbols created through interposition. For example, if a program wished to create an implementation of *malloc()* that embedded some statistics gathering about memory allocations, such an implementation could use the real *malloc()* definition to perform the memory allocation — and itself only embed the necessary logic to implement the statistics gathering function.

Note that conversion from a **void *** pointer to a function pointer as in:

```
fptr = (int (*)(int))dlsym(handle, "my_function");
```

is not defined by the ISO C standard. This standard requires this conversion to work correctly on conforming implementations.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

dlclose(), *dlerror()*, *dlopen()*

XBD [**<dlfcn.h>**](#)

CHANGE HISTORY

First released in Issue 5.

Issue 6

The **restrict** keyword is added to the *dlsym()* prototype for alignment with the ISO/IEC 9899:1999 standard.

The **RTLD_DEFAULT** and **RTLD_NEXT** flags are reserved for future use.

IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/14 is applied, correcting an example, and adding text to the **RATIONALE** describing issues related to conversion of pointers to functions and back again.

Issue 7

The *dlsym()* function is moved from the XSI option to the Base.

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0074 [74] is applied.

25016 **NAME**

25017 dprintf — print formatted output

25018 **SYNOPSIS**

25019 CX #include <stdio.h>
25020 int dprintf(int *fildes*, const char *restrict *format*, ...);

25021 **DESCRIPTION**

25022 Refer to *fprintf()*.

NAME

drand48, *erand48*, *jrand48*, *lcong48*, *lrand48*, *mrand48*, *rand48*, *seed48*, *srand48* — generate uniformly distributed pseudo-random numbers

SYNOPSIS

```
#include <stdlib.h>

double drand48(void);
double erand48(unsigned short xsubi[3]);
long jrand48(unsigned short xsubi[3]);
void lcong48(unsigned short param[7]);
long lrand48(void);
long mrand48(void);
long nrand48(unsigned short xsubi[3]);
unsigned short *seed48(unsigned short seed16v[3]);
void srand48(long seedval);
```

DESCRIPTION

This family of functions shall generate pseudo-random numbers using a linear congruential algorithm and 48-bit integer arithmetic.

The *drand48()* and *erand48()* functions shall return non-negative, double-precision, floating-point values, uniformly distributed over the interval [0.0,1.0).

The *lrand48()* and *nrand48()* functions shall return non-negative, long integers, uniformly distributed over the interval $[0, 2^{31})$.

The *mrand48()* and *jrand48()* functions shall return signed long integers uniformly distributed over the interval $[-2^{31}, 2^{31})$.

The *srand48()*, *seed48()*, and *lcong48()* functions are initialization entry points, one of which should be invoked before either *drand48()*, *lrand48()*, or *mrand48()* is called. (Although it is not recommended practice, constant default initializer values shall be supplied automatically if *drand48()*, *lrand48()*, or *mrand48()* is called without a prior call to an initialization entry point.) The *erand48()*, *nrand48()*, and *jrand48()* functions do not require an initialization entry point to be called first.

All the routines work by generating a sequence of 48-bit integer values, X_i , according to the linear congruential formula:

$$X_{n+1} = (aX_n + c)_{\text{mod } m} \quad n \geq 0$$

The parameter $m = 2^{48}$; hence 48-bit integer arithmetic is performed. Unless *lcong48()* is invoked, the multiplier value a and the addend value c are given by:

$$a = 5\text{DEECE66D}_{16} = 273673163155_8$$

$$c = \text{B}_{16} = 13_8$$

The value returned by any of the *drand48()*, *erand48()*, *jrand48()*, *lrand48()*, *mrand48()*, or *nrand48()* functions is computed by first generating the next 48-bit X_i in the sequence. Then the appropriate number of bits, according to the type of data item to be returned, are copied from the high-order (leftmost) bits of X_i and transformed into the returned value.

The *drand48()*, *lrand48()*, and *mrand48()* functions store the last 48-bit X_i generated in an internal buffer; that is why the application shall ensure that these are initialized prior to being invoked. The *erand48()*, *nrand48()*, and *jrand48()* functions require the calling program to

provide storage for the successive X_i values in the array specified as an argument when the functions are invoked. That is why these routines do not have to be initialized; the calling program merely has to place the desired initial value of X_i into the array and pass it as an argument. By using different arguments, *erand48()*, *nrand48()*, and *jrand48()* allow separate modules of a large program to generate several *independent* streams of pseudo-random numbers; that is, the sequence of numbers in each stream shall *not* depend upon how many times the routines are called to generate numbers for the other streams.

The initializer function *srand48()* sets the high-order 32 bits of X_i to the low-order 32 bits contained in its argument. The low-order 16 bits of X_i are set to the arbitrary value $330E_{16}$.

The initializer function *seed48()* sets the value of X_i to the 48-bit value specified in the argument array. The low-order 16 bits of X_i are set to the low-order 16 bits of *seed16v*[0]. The mid-order 16 bits of X_i are set to the low-order 16 bits of *seed16v*[1]. The high-order 16 bits of X_i are set to the low-order 16 bits of *seed16v*[2]. In addition, the previous value of X_i is copied into a 48-bit internal buffer, used only by *seed48()*, and a pointer to this buffer is the value returned by *seed48()*. This returned pointer, which can just be ignored if not needed, is useful if a program is to be restarted from a given point at some future time—use the pointer to get at and store the last X_i value, and then use this value to reinitialize via *seed48()* when the program is restarted.

The initializer function *lcong48()* allows the user to specify the initial X_i , the multiplier value a , and the addend value c . Argument array elements *param*[0-2] specify X_i , *param*[3-5] specify the multiplier a , and *param*[6] specifies the 16-bit addend c . After *lcong48()* is called, a subsequent call to either *srand48()* or *seed48()* shall restore the standard multiplier and addend values, a and c , specified above.

The *drand48()*, *lrand48()*, and *mrand48()* functions need not be thread-safe.

RETURN VALUE

As described in the DESCRIPTION above.

ERRORS

No errors are defined.

EXAMPLES

None.

APPLICATION USAGE

None.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

rand()

XBD <stdlib.h>

CHANGE HISTORY

First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 5

A note indicating that the *drand48()*, *lrand48()*, and *mrand48()* functions need not be reentrant is added to the DESCRIPTION.

25109	Issue 6	
25110		The normative text is updated to avoid use of the term “must” for application requirements.
25111	Issue 7	
25112		Austin Group Interpretation 1003.1-2001 #156 is applied.

25113 **NAME**25114 `dup, dup2` — duplicate an open file descriptor25115 **SYNOPSIS**

```
25116 #include <unistd.h>
25117 int dup(int fd);
25118 int dup2(int fd, int fd2);
```

25119 **DESCRIPTION**

25120 The `dup()` function provides an alternative interface to the service provided by `fcntl()` using the
 25121 `F_DUPFD` command. The call `dup(fd)` shall be equivalent to:

```
25122 fcntl(fd, F_DUPFD, 0);
```

25123 The `dup2()` function shall cause the file descriptor *fd2* to refer to the same open file
 25124 description as the file descriptor *fd* and to share any locks, and shall return *fd2*. If *fd2* is
 25125 already a valid open file descriptor, it shall be closed first, unless *fd* is equal to *fd2* in which
 25126 case `dup2()` shall return *fd2* without closing it. If the close operation fails to close *fd2*,
 25127 `dup2()` shall return `-1` without changing the open file description to which *fd2* refers. If *fd*
 25128 is not a valid file descriptor, `dup2()` shall return `-1` and shall not close *fd2*. If *fd2* is less than
 25129 0 or greater than or equal to `{OPEN_MAX}`, `dup2()` shall return `-1` with *errno* set to `[EBADF]`.

25130 Upon successful completion, if *fd* is not equal to *fd2*, the `FD_CLOEXEC` flag associated
 25131 with *fd2* shall be cleared. If *fd* is equal to *fd2*, the `FD_CLOEXEC` flag associated with
 25132 *fd2* shall not be changed.

25133 **TYM** If *fd* refers to a typed memory object, the result of the `dup2()` function is unspecified.

25134 **RETURN VALUE**

25135 Upon successful completion a non-negative integer, namely the file descriptor, shall be returned;
 25136 otherwise, `-1` shall be returned and *errno* set to indicate the error.

25137 **ERRORS**

25138 The `dup()` function shall fail if:

25139 `[EBADF]` The *fd* argument is not a valid open file descriptor.

25140 `[EMFILE]` All file descriptors available to the process are currently open.

25141 The `dup2()` function shall fail if:

25142 `[EBADF]` The *fd* argument is not a valid open file descriptor or the argument *fd2* is
 25143 negative or greater than or equal to `{OPEN_MAX}`.

25144 `[EINTR]` The `dup2()` function was interrupted by a signal.

25145 The `dup2()` function may fail if:

25146 `[EIO]` An I/O error occurred while attempting to close *fd2*.

25147 **EXAMPLES**25148 **Redirecting Standard Output to a File**

25149 The following example closes standard output for the current processes, re-assigns standard
 25150 output to go to the file referenced by *pfid*, and closes the original file descriptor to clean up.

```
25151 #include <unistd.h>
25152 ...
25153 int pfid;
25154 ...
```



```

25155     close(1);
25156     dup(pfd);
25157     close(pfd);
25158     ...

```

25159 Redirecting Error Messages

25160 The following example redirects messages from *stderr* to *stdout*.

```

25161     #include <unistd.h>
25162     ...
25163     dup2(1, 2);
25164     ...

```

25165 APPLICATION USAGE

25166 Implementations may use file descriptors that must be inherited into child processes for the
 25167 child process to remain conforming, such as for message catalog or tracing purposes. Therefore,
 25168 an application that calls *dup2()* with an arbitrary integer for *fildest* risks non-conforming
 25169 behavior, and *dup2()* can only portably be used to overwrite file descriptor values that the
 25170 application has obtained through explicit actions, or for the three file descriptors corresponding
 25171 to the standard file streams. In order to avoid a race condition of leaking an unintended file
 25172 descriptor into a child process, an application should consider opening all file descriptors with
 25173 the FD_CLOEXEC bit set unless the file descriptor is intended to be inherited across *exec*.

25174 RATIONALE

25175 The *dup()* function is redundant. Its services are also provided by the *fcntl()* function. It has been
 25176 included in this volume of POSIX.1-2008 primarily for historical reasons, since many existing
 25177 applications use it. On the other hand, the *dup2()* function provides unique services, as no other
 25178 interface is able to atomically replace an existing file descriptor.

25179 The *dup2()* function is not marked obsolescent because it presents a type-safe version of
 25180 functionality provided in a type-unsafe version by *fcntl()*. It is used in the POSIX Ada binding.

25181 The *dup2()* function is not intended for use in critical regions as a synchronization mechanism.

25182 In the description of [EBADF], the case of *fildest* being out of range is covered by the given case of
 25183 *fildest* not being valid. The descriptions for *fildest* and *fildest2* are different because the only kind of
 25184 invalidity that is relevant for *fildest2* is whether it is out of range; that is, it does not matter
 25185 whether *fildest2* refers to an open file when the *dup2()* call is made.

25186 FUTURE DIRECTIONS

25187 None.

25188 SEE ALSO

25189 *close()*, *fcntl()*, *open()*

25190 XBD **<unistd.h>**

25191 CHANGE HISTORY

25192 First released in Issue 1. Derived from Issue 1 of the SVID.

25193 Issue 7

25194 SD5-XSH-ERN-187 is applied.

25195 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0075 [149,428] and +
 25196 XSH/TC1-2008/0076 [149] are applied.

25197 **NAME**

25198 duplocale — duplicate a locale object

25199 **SYNOPSIS**

```

25200 CX      #include <locale.h>
25201         locale_t duplocale(locale_t locobj);

```

25202 **DESCRIPTION**

25203 The *duplocale()* function shall create a duplicate copy of the locale object referenced by the *locobj*
 25204 argument.

25205 If the *locobj* argument is `LC_GLOBAL_LOCALE`, *duplocale()* shall create a new locale object +
 25206 containing a copy of the global locale determined by the *setlocale()* function. +

25207 The behavior is undefined if the *locobj* argument is not a valid locale object handle.

25208 **RETURN VALUE**

25209 Upon successful completion, the *duplocale()* function shall return a handle for a new locale
 25210 object. Otherwise, *duplocale()* shall return `(locale_t)0` and set *errno* to indicate the error.

25211 **ERRORS**

25212 The *duplocale()* function shall fail if:

25213 [ENOMEM] There is not enough memory available to create the locale object or load the
 25214 locale data.

25215 **EXAMPLES**25216 **Constructing an Altered Version of an Existing Locale Object**

25217 The following example shows a code fragment to create a slightly altered version of an existing
 25218 locale object. The function takes a locale object and a locale name and it replaces the *LC_TIME*
 25219 category data in the locale object with that from the named locale.

```

25220     #include <locale.h>
25221     ...
25222     locale_t
25223     with_changed_lc_time (locale_t obj, const char *name)
25224     {
25225         locale_t retval = duplocale (obj);
25226         if (retval != (locale_t) 0)
25227         {
25228             locale_t changed = newlocale (LC_TIME_MASK, name, retval);
25229             if (changed == (locale_t) 0)
25230                 /* An error occurred. Free all allocated resources. */
25231                 freelocale (retval);
25232             retval = changed;
25233         }
25234         return retval; }
25235     }

```

APPLICATION USAGE

The use of the *duplocale()* function is recommended for situations where a locale object is being used in multiple places, and it is possible that the lifetime of the locale object might end before all uses are finished. Another reason to duplicate a locale object is if a slightly modified form is needed. This can be achieved by a call to *newlocale()* following the *duplocale()* call.

As with the *newlocale()* function, handles for locale objects created by the *duplocale()* function should be released by a corresponding call to *freelocale()*.

The *duplocale()* function can also be used in conjunction with *uselocale(locale_t)*. This returns the locale in effect for the calling thread, but can have the value LC_GLOBAL_LOCALE. Passing LC_GLOBAL_LOCALE to functions such as *isalnum_l()* results in undefined behavior, but applications can convert it into a usable locale object by using *duplocale()*.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

freelocale(), *newlocale()*, *uselocale()*

XBD <locale.h>

CHANGE HISTORY

First released in Issue 7.

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0077 [283,301], XSH/TC1-2008/0078 [283], and XSH/TC1-2008/0079 [301] are applied.

NAME

encrypt — encoding function (**CRYPT**)

SYNOPSIS

```
XSI      #include <unistd.h>
void encrypt(char block[64], int edflag);
```

DESCRIPTION

The *encrypt()* function shall provide access to an implementation-defined encoding algorithm. The key generated by *setkey()* is used to encrypt the string *block* with *encrypt()*.

The *block* argument to *encrypt()* shall be an array of length 64 bytes containing only the bytes with values of 0 and 1. The array is modified in place to a similar array using the key set by *setkey()*. If *edflag* is 0, the argument is encoded. If *edflag* is 1, the argument may be decoded (see the APPLICATION USAGE section); if the argument is not decoded, *errno* shall be set to [ENOSYS].

The *encrypt()* function shall not change the setting of *errno* if successful. An application wishing to check for error situations should set *errno* to 0 before calling *encrypt()*. If *errno* is non-zero on return, an error has occurred.

The *encrypt()* function need not be thread-safe.

RETURN VALUE

The *encrypt()* function shall not return a value.

ERRORS

The *encrypt()* function shall fail if:

[ENOSYS] The functionality is not supported on this implementation.

EXAMPLES

None.

APPLICATION USAGE

Historical implementations of the *encrypt()* function used a rather primitive encoding algorithm.

In some environments, decoding might not be implemented. This is related to some Government restrictions on encryption and decryption routines. Historical practice has been to ship a different version of the encryption library without the decryption feature in the routines supplied. Thus the exported version of *encrypt()* does encoding but not decoding.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

crypt(), *setkey()*

XBD <unistd.h>

CHANGE HISTORY

First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 5

A note indicating that this function need not be reentrant is added to the DESCRIPTION.

Issue 6

In the DESCRIPTION, the note about reentrancy is expanded to cover thread-safety.

Issue 7

Austin Group Interpretation 1003.1-2001 #156 is applied.

25303 **NAME**

25304 endgrent, getgrent, setgrent — group database entry functions

25305 **SYNOPSIS**

```

25306 XSI      #include <grp.h>
25307          void endgrent(void);
25308          struct group *getgrent(void);
25309          void setgrent(void);

```

25310 **DESCRIPTION**

25311 The *getgrent()* function shall return a pointer to a structure containing the broken-out fields of an
 25312 entry in the group database. When first called, *getgrent()* shall return a pointer to a **group**
 25313 structure containing the first entry in the group database. Thereafter, it shall return a pointer to a
 25314 **group** structure containing the next group structure in the group database, so successive calls
 25315 may be used to search the entire database.

25316 An implementation that provides extended security controls may impose further
 25317 implementation-defined restrictions on accessing the group database. In particular, the system
 25318 may deny the existence of some or all of the group database entries associated with groups other
 25319 than those groups associated with the caller and may omit users other than the caller from the
 25320 list of members of groups in database entries that are returned.

25321 The *setgrent()* function shall rewind the group database to allow repeated searches.

25322 The *endgrent()* function may be called to close the group database when processing is complete.

25323 These functions need not be thread-safe.

25324 **RETURN VALUE**

25325 When first called, *getgrent()* shall return a pointer to the first group structure in the group
 25326 database. Upon subsequent calls it shall return the next group structure in the group database.
 25327 The *getgrent()* function shall return a null pointer on end-of-file or an error and *errno* may be set
 25328 to indicate the error.

25329 The application shall not modify the structure to which the return value points, nor any storage |
 25330 areas pointed to by pointers within the structure. The returned pointer, and pointers within the |
 25331 structure, might be invalidated or the structure or the storage areas might be overwritten by a |
 25332 subsequent call to *getgrgid()*, *getgrnam()*, or *getgrent()*.

25333 **ERRORS**

25334 The *getgrent()* function may fail if:

25335	[EINTR]	A signal was caught during the operation.
25336	[EIO]	An I/O error has occurred.
25337	[EMFILE]	All file descriptors available to the process are currently open.
25338	[ENFILE]	The maximum allowable number of files is currently open in the system.

25339 EXAMPLES

25340 None.

25341 APPLICATION USAGE

25342 These functions are provided due to their historical usage. Applications should avoid
25343 dependencies on fields in the group database, whether the database is a single file, or where in
25344 the file system name space the database resides. Applications should use *getgrnam()* and
25345 *getgrgid()* whenever possible because it avoids these dependencies.

25346 RATIONALE

25347 None.

25348 FUTURE DIRECTIONS

25349 None.

25350 SEE ALSO

25351 *endpwent()*, *getgrgid()*, *getgrnam()*, *getlogin()*

25352 XBD <grp.h>

25353 CHANGE HISTORY

25354 First released in Issue 4, Version 2.

25355 Issue 5

25356 Moved from X/OPEN UNIX extension to BASE.

25357 Normative text previously in the APPLICATION USAGE section is moved to the RETURN
25358 VALUE section.

25359 A note indicating that these functions need not be reentrant is added to the DESCRIPTION.

25360 Issue 6

25361 In the DESCRIPTION, the note about reentrancy is expanded to cover thread-safety.

25362 Issue 7

25363 Austin Group Interpretation 1003.1-2001 #156 is applied.

25364 SD5-XBD-ERN-4 is applied, changing the definition of the [EMFILE] error.

25365 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0080 [75] is applied.

+

25366 **NAME**

25367 endhostent, gethostent, sethostent — network host database functions

25368 **SYNOPSIS**

```
25369     #include <netdb.h>
25370
25370     void endhostent(void);
25371     struct hostent *gethostent(void);
25372     void sethostent(int stayopen);
```

25373 **DESCRIPTION**

25374 These functions shall retrieve information about hosts. This information is considered to be
 25375 stored in a database that can be accessed sequentially or randomly. The implementation of this
 25376 database is unspecified.

25377 **Note:** In many cases this database is implemented by the Domain Name System, as documented in
 25378 RFC 1034, RFC 1035, and RFC 1886.

25379 The *sethostent()* function shall open a connection to the database and set the next entry for
 25380 retrieval to the first entry in the database. If the *stayopen* argument is non-zero, the connection
 25381 shall not be closed by a call to *gethostent()*, and the implementation may maintain an open file
 25382 descriptor.

25383 The *gethostent()* function shall read the next entry in the database, opening and closing a
 25384 connection to the database as necessary.

25385 Entries shall be returned in **hostent** structures.

25386 The *endhostent()* function shall close the connection to the database, releasing any open file
 25387 descriptor.

25388 These functions need not be thread-safe.

25389 **RETURN VALUE**

25390 Upon successful completion, the *gethostent()* function shall return a pointer to a **hostent**
 25391 structure if the requested entry was found, and a null pointer if the end of the database was
 25392 reached or the requested entry was not found.

25393 The application shall not modify the structure to which the return value points, nor any storage +
 25394 areas pointed to by pointers within the structure. The returned pointer, and pointers within the +
 25395 structure, might be invalidated or the structure or the storage areas might be overwritten by a +
 25396 subsequent call to *gethostent()*.

25397 **ERRORS**

25398 No errors are defined for *endhostent()*, *gethostent()*, and *sethostent()*.

25399 **EXAMPLES**

25400 None.

25401 **APPLICATION USAGE**

25402 None.

25403 **RATIONALE**

25404 None.

25405 **FUTURE DIRECTIONS**

25406 None.

25407 **SEE ALSO**25408 *endservent()*

25409 XBD <netdb.h>

25410 **CHANGE HISTORY**

25411 First released in Issue 6. Derived from the XNS, Issue 5.2 specification.

25412 **Issue 7**

25413 Austin Group Interpretation 1003.1-2001 #156 is applied.

25414 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0081 [75,428] and +
25415 XSH/TC1-2008/0082 [75] are applied.

NAME

endnetent, getnetbyaddr, getnetbyname, getnetent, setnetent — network database functions

SYNOPSIS

```
#include <netdb.h>

void endnetent(void);
struct netent *getnetbyaddr(uint32_t net, int type);
struct netent *getnetbyname(const char *name);
struct netent *getnetent(void);
void setnetent(int stayopen);
```

DESCRIPTION

These functions shall retrieve information about networks. This information is considered to be stored in a database that can be accessed sequentially or randomly. The implementation of this database is unspecified.

The *setnetent()* function shall open and rewind the database. If the *stayopen* argument is non-zero, the connection to the *net* database shall not be closed after each call to *getnetent()* (either directly, or indirectly through one of the other *getnet*()* functions), and the implementation may maintain an open file descriptor to the database.

The *getnetent()* function shall read the next entry of the database, opening and closing a connection to the database as necessary.

The *getnetbyaddr()* function shall search the database from the beginning, and find the first entry for which the address family specified by *type* matches the *n_addrtype* member and the network number *net* matches the *n_net* member, opening and closing a connection to the database as necessary. The *net* argument shall be the network number in host byte order.

The *getnetbyname()* function shall search the database from the beginning and find the first entry for which the network name specified by *name* matches the *n_name* member, opening and closing a connection to the database as necessary.

The *getnetbyaddr()*, *getnetbyname()*, and *getnetent()* functions shall each return a pointer to a **netent** structure, the members of which shall contain the fields of an entry in the network database.

The *endnetent()* function shall close the database, releasing any open file descriptor.

These functions need not be thread-safe.

RETURN VALUE

Upon successful completion, *getnetbyaddr()*, *getnetbyname()*, and *getnetent()* shall return a pointer to a **netent** structure if the requested entry was found, and a null pointer if the end of the database was reached or the requested entry was not found. Otherwise, a null pointer shall be returned.

The application shall not modify the structure to which the return value points, nor any storage areas pointed to by pointers within the structure. The returned pointer, and pointers within the structure, might be invalidated or the structure or the storage areas might be overwritten by a subsequent call to *getnetbyaddr()*, *getnetbyname()*, or *getnetent()*.

ERRORS

No errors are defined.

25458 **EXAMPLES**

25459 None.

25460 **APPLICATION USAGE**

25461 None. |

25462 **RATIONALE**

25463 None.

25464 **FUTURE DIRECTIONS**

25465 None.

25466 **SEE ALSO**25467 XBD <[netdb.h](#)>25468 **CHANGE HISTORY**

25469 First released in Issue 6. Derived from the XNS, Issue 5.2 specification.

25470 **Issue 7**

25471 Austin Group Interpretation 1003.1-2001 #156 is applied.

25472 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0083 [75] and XSH/TC1-2008/0084 +
25473 [75] are applied.

25474 **NAME**

25475 endprotoent, getprotobyname, getprotobynumber, getprotoent, setprotoent — network protocol
 25476 database functions

25477 **SYNOPSIS**

```
25478 #include <netdb.h>
25479
25479 void endprotoent(void);
25480 struct protoent *getprotobyname(const char *name);
25481 struct protoent *getprotobynumber(int proto);
25482 struct protoent *getprotoent(void);
25483 void setprotoent(int stayopen);
```

25484 **DESCRIPTION**

25485 These functions shall retrieve information about protocols. This information is considered to be
 25486 stored in a database that can be accessed sequentially or randomly. The implementation of this
 25487 database is unspecified.

25488 The *setprotoent()* function shall open a connection to the database, and set the next entry to the
 25489 first entry. If the *stayopen* argument is non-zero, the connection to the network protocol database
 25490 shall not be closed after each call to *getprotoent()* (either directly, or indirectly through one of the
 25491 other *getproto**() functions), and the implementation may maintain an open file descriptor for
 25492 the database.

25493 The *getprotobyname()* function shall search the database from the beginning and find the first
 25494 entry for which the protocol name specified by *name* matches the *p_name* member, opening and
 25495 closing a connection to the database as necessary.

25496 The *getprotobynumber()* function shall search the database from the beginning and find the first
 25497 entry for which the protocol number specified by *proto* matches the *p_proto* member, opening
 25498 and closing a connection to the database as necessary.

25499 The *getprotoent()* function shall read the next entry of the database, opening and closing a
 25500 connection to the database as necessary.

25501 The *getprotobyname()*, *getprotobynumber()*, and *getprotoent()* functions shall each return a pointer
 25502 to a **protoent** structure, the members of which shall contain the fields of an entry in the network
 25503 protocol database.

25504 The *endprotoent()* function shall close the connection to the database, releasing any open file
 25505 descriptor.

25506 These functions need not be thread-safe.

25507 **RETURN VALUE**

25508 Upon successful completion, *getprotobyname()*, *getprotobynumber()*, and *getprotoent()* return a
 25509 pointer to a **protoent** structure if the requested entry was found, and a null pointer if the end of
 25510 the database was reached or the requested entry was not found. Otherwise, a null pointer is
 25511 returned.

25512 The application shall not modify the structure to which the return value points, nor any storage +
 25513 areas pointed to by pointers within the structure. The returned pointer, and pointers within the +
 25514 structure, might be invalidated or the structure or the storage areas might be overwritten by a +
 25515 subsequent call to *getprotobyname()*, *getprotobynumber()*, or *getprotoent()*.

25516 ERRORS

25517 No errors are defined.

25518 EXAMPLES

25519 None.

25520 APPLICATION USAGE

25521 None.

25522 RATIONALE

25523 None.

25524 FUTURE DIRECTIONS

25525 None.

25526 SEE ALSO

25527 XBD [<netdb.h>](#)

25528 CHANGE HISTORY

25529 First released in Issue 6. Derived from the XNS, Issue 5.2 specification.

25530 Issue 7

25531 Austin Group Interpretation 1003.1-2001 #156 is applied.

25532 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0085 [75] and XSH/TC1-2008/0086 +
25533 [75] are applied.

NAME

endpwent, getpwent, setpwent — user database functions

SYNOPSIS

```
XSI    #include <pwd.h>

void endpwent(void);
struct passwd *getpwent(void);
void setpwent(void);
```

DESCRIPTION

These functions shall retrieve information about users.

The *getpwent()* function shall return a pointer to a structure containing the broken-out fields of an entry in the user database. Each entry in the user database contains a **passwd** structure. When first called, *getpwent()* shall return a pointer to a **passwd** structure containing the first entry in the user database. Thereafter, it shall return a pointer to a **passwd** structure containing the next entry in the user database. Successive calls can be used to search the entire user database.

If an end-of-file or an error is encountered on reading, *getpwent()* shall return a null pointer.

An implementation that provides extended security controls may impose further implementation-defined restrictions on accessing the user database. In particular, the system may deny the existence of some or all of the user database entries associated with users other than the caller.

The *setpwent()* function effectively rewinds the user database to allow repeated searches.

The *endpwent()* function may be called to close the user database when processing is complete.

These functions need not be thread-safe.

RETURN VALUE

The *getpwent()* function shall return a null pointer on end-of-file or error.

The application shall not modify the structure to which the return value points, nor any storage areas pointed to by pointers within the structure. The returned pointer, and pointers within the structure, might be invalidated or the structure or the storage areas might be overwritten by a subsequent call to *getpwuid()*, *getpwnam()*, or *getpwent()*.

ERRORS

These functions may fail if:

[EIO] An I/O error has occurred.

In addition, *getpwent()* and *setpwent()* may fail if:

[EMFILE] All file descriptors available to the process are currently open.

[ENFILE] The maximum allowable number of files is currently open in the system.

25568 **EXAMPLES**25569 **Searching the User Database**

25570 The following example uses the *getpwent()* function to get successive entries in the user
 25571 database, returning a pointer to a **passwd** structure that contains information about each user.
 25572 The call to *endpwent()* closes the user database and cleans up.

```
25573 #include <pwd.h>
25574 #include <stdio.h>

25575 void printname(uid_t uid)
25576 {
25577     struct passwd *pwd;

25578     setpwent();
25579     while((pwd = getpwent()) != NULL) {
25580         if (pwd->pw_uid == uid) {
25581             printf("name=%s\n", pwd->pw_name);
25582             break;
25583         }
25584     }
25585     endpwent();
25586 }
```

25587 **APPLICATION USAGE**

25588 These functions are provided due to their historical usage. Applications should avoid
 25589 dependencies on fields in the password database, whether the database is a single file, or where
 25590 in the file system name space the database resides. Applications should use *getpwuid()*
 25591 whenever possible because it avoids these dependencies.

25592 **RATIONALE**

25593 None.

25594 **FUTURE DIRECTIONS**

25595 None.

25596 **SEE ALSO**

25597 *endgrent()*, *getlogin()*, *getpwnam()*, *getpwuid()*

25598 XBD **<pwd.h>**

25599 **CHANGE HISTORY**

25600 First released in Issue 4, Version 2.

25601 **Issue 5**

25602 Moved from X/OPEN UNIX extension to BASE.

25603 Normative text previously in the APPLICATION USAGE section is moved to the RETURN
 25604 VALUE section.

25605 A note indicating that these functions need not be reentrant is added to the DESCRIPTION.

25606 **Issue 6**

25607 In the DESCRIPTION, the note about reentrancy is expanded to cover thread-safety.

25608	Issue 7	
25609	Austin Group Interpretation 1003.1-2001 #156 is applied.	
25610	SD5-XBD-ERN-4 is applied, changing the definition of the [EMFILE] error.	
25611	The EXAMPLES section is revised.	
25612	POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0087 [75] is applied.	+

NAME

endservent, getservbyname, getservbyport, getservent, setservent — network services database functions

SYNOPSIS

```
#include <netdb.h>

void endservent(void);
struct servent *getservbyname(const char *name, const char *proto);
struct servent *getservbyport(int port, const char *proto);
struct servent *getservent(void);
void setservent(int stayopen);
```

DESCRIPTION

These functions shall retrieve information about network services. This information is considered to be stored in a database that can be accessed sequentially or randomly. The implementation of this database is unspecified.

The *setservent()* function shall open a connection to the database, and set the next entry to the first entry. If the *stayopen* argument is non-zero, the *net* database shall not be closed after each call to the *getservent()* function (either directly, or indirectly through one of the other *getserv*()* functions), and the implementation may maintain an open file descriptor for the database.

The *getservent()* function shall read the next entry of the database, opening and closing a connection to the database as necessary.

The *getservbyname()* function shall search the database from the beginning and find the first entry for which the service name specified by *name* matches the *s_name* member and the protocol name specified by *proto* matches the *s_proto* member, opening and closing a connection to the database as necessary. If *proto* is a null pointer, any value of the *s_proto* member shall be matched.

The *getservbyport()* function shall search the database from the beginning and find the first entry for which the port specified by *port* matches the *s_port* member and the protocol name specified by *proto* matches the *s_proto* member, opening and closing a connection to the database as necessary. If *proto* is a null pointer, any value of the *s_proto* member shall be matched. The *port* argument shall be a value obtained by converting a **uint16_t** in network byte order to **int**.

The *getservbyname()*, *getservbyport()*, and *getservent()* functions shall each return a pointer to a **servent** structure, the members of which shall contain the fields of an entry in the network services database.

The *endservent()* function shall close the database, releasing any open file descriptor.

These functions need not be thread-safe.

RETURN VALUE

Upon successful completion, *getservbyname()*, *getservbyport()*, and *getservent()* return a pointer to a **servent** structure if the requested entry was found, and a null pointer if the end of the database was reached or the requested entry was not found. Otherwise, a null pointer is returned.

The application shall not modify the structure to which the return value points, nor any storage areas pointed to by pointers within the structure. The returned pointer, and pointers within the structure, might be invalidated or the structure or the storage areas might be overwritten by a subsequent call to *getservbyname()*, *getservbyport()*, or *getservent()*.

25656 **ERRORS**

25657 No errors are defined.

25658 **EXAMPLES**

25659 None.

25660 **APPLICATION USAGE**25661 The *port* argument of *getservbyport()* need not be compatible with the port values of all address
25662 families. -25663 **RATIONALE**

25664 None.

25665 **FUTURE DIRECTIONS**

25666 None.

25667 **SEE ALSO**25668 *endhostent()*, *endprotoent()*, *htonl()*, *inet_addr()*

25669 XBD <netdb.h>

25670 **CHANGE HISTORY**

25671 First released in Issue 6. Derived from the XNS, Issue 5.2 specification.

25672 **Issue 7**

25673 Austin Group Interpretation 1003.1-2001 #156 is applied.

25674 SD5-XBD-ERN-14 is applied.

25675 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0088 [75] and XSH/TC1-2008/0089 +
25676 [75] are applied.

NAME

endutxent, getutxent, getutxid, getutxline, pututxline, setutxent — user accounting database functions

SYNOPSIS

```
#include <utmpx.h>

void endutxent(void);
struct utmpx *getutxent(void);
struct utmpx *getutxid(const struct utmpx *id);
struct utmpx *getutxline(const struct utmpx *line);
struct utmpx *pututxline(const struct utmpx *utmpx);
void setutxent(void);
```

DESCRIPTION

These functions shall provide access to the user accounting database.

The *getutxent()* function shall read the next entry from the user accounting database. If the database is not already open, it shall open it. If it reaches the end of the database, it shall fail.

The *getutxid()* function shall search forward from the current point in the database. If the *ut_type* value of the **utmpx** structure pointed to by *id* is *BOOT_TIME*, *OLD_TIME*, or *NEW_TIME*, then it shall stop when it finds an entry with a matching *ut_type* value. If the *ut_type* value is *INIT_PROCESS*, *LOGIN_PROCESS*, *USER_PROCESS*, or *DEAD_PROCESS*, then it shall stop when it finds an entry whose type is one of these four and whose *ut_id* member matches the *ut_id* member of the **utmpx** structure pointed to by *id*. If the end of the database is reached without a match, *getutxid()* shall fail.

The *getutxline()* function shall search forward from the current point in the database until it finds an entry of the type *LOGIN_PROCESS* or *USER_PROCESS* which also has a *ut_line* value matching that in the **utmpx** structure pointed to by *line*. If the end of the database is reached without a match, *getutxline()* shall fail.

The *getutxid()* or *getutxline()* function may cache data. For this reason, to use *getutxline()* to search for multiple occurrences, the application shall zero out the static data after each success, or *getutxline()* may return a pointer to the same **utmpx** structure.

There is one exception to the rule about clearing the structure before further reads are done. The implicit read done by *pututxline()* (if it finds that it is not already at the correct place in the user accounting database) shall not modify the static structure returned by *getutxent()*, *getutxid()*, or *getutxline()*, if the application has modified this structure and passed the pointer back to *pututxline()*.

For all entries that match a request, the *ut_type* member indicates the type of the entry. Other members of the entry shall contain meaningful data based on the value of the *ut_type* member as follows:

ut_type Member	Other Members with Meaningful Data
EMPTY	No others
BOOT_TIME	ut_tv
OLD_TIME	ut_tv
NEW_TIME	ut_tv
USER_PROCESS	ut_id, ut_user (login name of the user), ut_line, ut_pid, ut_tv
INIT_PROCESS	ut_id, ut_pid, ut_tv
LOGIN_PROCESS	ut_id, ut_user (implementation-defined name of the login process), ut_line, ut_pid, ut_tv
DEAD_PROCESS	ut_id, ut_pid, ut_tv

An implementation that provides extended security controls may impose implementation-defined restrictions on accessing the user accounting database. In particular, the system may deny the existence of some or all of the user accounting database entries associated with users other than the caller.

If the process has appropriate privileges, the *pututxline()* function shall write out the structure into the user accounting database. It shall search for a record as if by *getutxid()* that satisfies the request. If this search succeeds, then the entry shall be replaced. Otherwise, a new entry shall be made at the end of the user accounting database.

The *endutxent()* function shall close the user accounting database.

The *setutxent()* function shall reset the input to the beginning of the database. This should be done before each search for a new entry if it is desired that the entire database be examined.

These functions need not be thread-safe.

RETURN VALUE

Upon successful completion, *getutxent()*, *getutxid()*, and *getutxline()* shall return a pointer to a **utmpx** structure containing a copy of the requested entry in the user accounting database. Otherwise, a null pointer shall be returned.

The return value may point to a static area which is overwritten by a subsequent call to *getutxid()* or *getutxline()*.

Upon successful completion, *pututxline()* shall return a pointer to a **utmpx** structure containing a copy of the entry added to the user accounting database. Otherwise, a null pointer shall be returned.

The *endutxent()* and *setutxent()* functions shall not return a value.

ERRORS

No errors are defined for the *endutxent()*, *getutxent()*, *getutxid()*, *getutxline()*, and *setutxent()* functions.

The *pututxline()* function may fail if:

[EPERM] The process does not have appropriate privileges.

25751 EXAMPLES

25752 None.

25753 APPLICATION USAGE

25754 The sizes of the arrays in the structure can be found using the *sizeof* operator.

25755 RATIONALE

25756 None.

25757 FUTURE DIRECTIONS

25758 None.

25759 SEE ALSO

25760 XBD [<utmpx.h>](#)

25761 CHANGE HISTORY

25762 First released in Issue 4, Version 2.

25763 Issue 5

25764 Moved from X/OPEN UNIX extension to BASE.

25765 Normative text previously in the APPLICATION USAGE section is moved to the
25766 DESCRIPTION.

25767 A note indicating that these functions need not be reentrant is added to the DESCRIPTION.

25768 Issue 6

25769 In the DESCRIPTION, the note about reentrancy is expanded to cover thread-safety.

25770 Issue 7

25771 Austin Group Interpretation 1003.1-2001 #156 is applied.

25772 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0090 [213,428] and +
25773 XSH/TC1-2008/0091 [213] are applied.

25774 NAME

25775 environ — array of character pointers to the environment strings

25776 SYNOPSIS

25777 extern char **environ;

25778 DESCRIPTION

25779 Refer to *exec* and XBD [Chapter 8](#) (on page 173).

25780 **NAME**

25781 erand48 — generate uniformly distributed pseudo-random numbers

25782 **SYNOPSIS**

25783 XSI #include <stdlib.h>

25784 double erand48(unsigned short xsubi[3]);

25785 **DESCRIPTION**25786 Refer to *drand48()*.

25787 **NAME**

25788 erf, erff, erfl — error functions

25789 **SYNOPSIS**

```
25790 #include <math.h>
25791 double erf(double x);
25792 float erff(float x);
25793 long double erfl(long double x);
```

25794 **DESCRIPTION**

25795 CX The functionality described on this reference page is aligned with the ISO C standard. Any
 25796 conflict between the requirements described here and the ISO C standard is unintentional. This
 25797 volume of POSIX.1-2008 defers to the ISO C standard.

25798 These functions shall compute the error function of their argument x , defined as:

$$25799 \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt$$

25800 An application wishing to check for error situations should set *errno* to zero and call
 25801 *feclearexcept*(FE_ALL_EXCEPT) before calling these functions. On return, if *errno* is non-zero or
 25802 *fetestexcept*(FE_INVALID | FE_DIVBYZERO | FE_OVERFLOW | FE_UNDERFLOW) is non-
 25803 zero, an error has occurred.

25804 **RETURN VALUE**

25805 Upon successful completion, these functions shall return the value of the error function.

25806 MX If x is NaN, a NaN shall be returned.

25807 If x is ± 0 , ± 0 shall be returned.

25808 If x is $\pm \text{Inf}$, ± 1 shall be returned.

25809 If the correct value would cause underflow, a range error may occur, and *erf()*, *erff()*, and *erfl()* |
 25810 shall return an implementation-defined value no greater in magnitude than DBL_MIN, |
 25811 FLT_MIN, and LDBL_MIN, respectively. |

25812 MXX If the IEC 60559 Floating-Point option is supported, $2 * x / \sqrt{\pi}$ should be returned.

25813 **ERRORS**

25814 These functions may fail if:

25815 MX **Range Error** The result underflows.

25816 If the integer expression (*math_errhandling* & MATH_ERRNO) is non-zero,
 25817 then *errno* shall be set to [ERANGE]. If the integer expression
 25818 (*math_errhandling* & MATH_ERREXCEPT) is non-zero, then the underflow
 25819 floating-point exception shall be raised.

25820 **EXAMPLES**25821 **Computing the Probability for a Normal Variate**

25822 This example shows how to use *erf()* to compute the probability that a normal variate assumes a
 25823 value in the range $[x_1, x_2]$ with $x_1 \leq x_2$.

25824 This example uses the constant `M_SQRT1_2` which is part of the XSI option.

```
25825 #include <math.h>
25826
25827 double
25828 Phi(const double x1, const double x2)
25829 {
25830     return ( erf(x2*M_SQRT1_2) - erf(x1*M_SQRT1_2) ) / 2;
```

25831 **APPLICATION USAGE**

25832 Underflow occurs when $|x| < \text{DBL\_MIN} * (\sqrt{\pi}/2)$.

25833 On error, the expressions `(math_errhandling & MATH_ERRNO)` and `(math_errhandling &`
 25834 `MATH_ERREXCEPT)` are independent of each other, but at least one of them must be non-zero.

25835 **RATIONALE**

25836 None.

25837 **FUTURE DIRECTIONS**

25838 None.

25839 **SEE ALSO**

25840 *erfc()*, *feclearexcept()*, *fetestexcept()*, *isnan()*

25841 XBD [Section 4.19](#) (on page 116), `<math.h>`

25842 **CHANGE HISTORY**

25843 First released in Issue 1. Derived from Issue 1 of the SVID.

25844 **Issue 5**

25845 The DESCRIPTION is updated to indicate how an application should check for an error. This
 25846 text was previously published in the APPLICATION USAGE section.

25847 **Issue 6**

25848 The *erf()* function is no longer marked as an extension.

25849 The *erfc()* function is split out onto its own reference page.

25850 The *erfff()* and *erfl()* functions are added for alignment with the ISO/IEC 9899:1999 standard.

25851 The DESCRIPTION, RETURN VALUE, ERRORS, and APPLICATION USAGE sections are
 25852 revised to align with the ISO/IEC 9899:1999 standard.

25853 IEC 60559:1989 standard floating-point extensions over the ISO/IEC 9899:1999 standard are
 25854 marked.

25855 IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/22 is applied, adding the example to the
 25856 EXAMPLES section.

25857 **Issue 7**

25858 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0092 [68] is applied.

+

25859 **NAME**25860 `erfc`, `erfcf`, `erfcl` — complementary error functions25861 **SYNOPSIS**

```
25862     #include <math.h>

25863     double erfc(double x);
25864     float erfcf(float x);
25865     long double erfcl(long double x);
```

25866 **DESCRIPTION**

25867 CX The functionality described on this reference page is aligned with the ISO C standard. Any
 25868 conflict between the requirements described here and the ISO C standard is unintentional. This
 25869 volume of POSIX.1-2008 defers to the ISO C standard.

25870 These functions shall compute the complementary error function $1.0 - \text{erf}(x)$.

25871 An application wishing to check for error situations should set *errno* to zero and call
 25872 *feclearexcept*(FE_ALL_EXCEPT) before calling these functions. On return, if *errno* is non-zero or
 25873 *fetestexcept*(FE_INVALID | FE_DIVBYZERO | FE_OVERFLOW | FE_UNDERFLOW) is non-
 25874 zero, an error has occurred.

25875 **RETURN VALUE**

25876 Upon successful completion, these functions shall return the value of the complementary error
 25877 function.

25878 MXX If the correct value would cause underflow, and is not representable, a range error may occur, |
 25879 MXX and *erfc*(), *erfcf*(), and *erfcl*() shall return 0.0, or (if the IEC 60559 Floating-Point option is not |
 25880 supported) an implementation-defined value no greater in magnitude than DBL_MIN, |
 25881 FLT_MIN, and LDBL_MIN, respectively.

25882 MX If *x* is NaN, a NaN shall be returned.

25883 If *x* is ± 0 , +1 shall be returned.

25884 If *x* is $-\text{Inf}$, +2 shall be returned.

25885 If *x* is $+\text{Inf}$, +0 shall be returned.

25886 MXX If the correct value would cause underflow and is representable, a range error may occur and
 25887 the correct value shall be returned.

25888 **ERRORS**

25889 These functions may fail if:

25890 Range Error The result underflows.

25891 If the integer expression (*math_errhandling* & MATH_ERRNO) is non-zero,
 25892 then *errno* shall be set to [ERANGE]. If the integer expression
 25893 (*math_errhandling* & MATH_ERREXCEPT) is non-zero, then the underflow
 25894 floating-point exception shall be raised.

25895 **EXAMPLES**

25896 None.

25897 **APPLICATION USAGE**

25898 The *erfc()* function is provided because of the extreme loss of relative accuracy if *erf(x)* is called
25899 for large *x* and the result subtracted from 1.0.

25900 Note for IEEE Std 754-1985 **double**, $26.55 < x$ implies *erfc(x)* has underflowed.

25901 On error, the expressions (*math_errhandling* & MATH_ERRNO) and (*math_errhandling* &
25902 MATH_ERREXCEPT) are independent of each other, but at least one of them must be non-zero.

25903 **RATIONALE**

25904 None.

25905 **FUTURE DIRECTIONS**

25906 None.

25907 **SEE ALSO**25908 *erf()*, *feclearexcept()*, *fetestexcept()*, *isnan()*

25909 XBD Section 4.19 (on page 116), <math.h>

25910 **CHANGE HISTORY**

25911 First released in Issue 1. Derived from Issue 1 of the SVID.

25912 **Issue 5**

25913 The DESCRIPTION is updated to indicate how an application should check for an error. This
25914 text was previously published in the APPLICATION USAGE section.

25915 **Issue 6**25916 The *erfc()* function is no longer marked as an extension.25917 These functions are split out from the *erf()* reference page.

25918 The DESCRIPTION, RETURN VALUE, ERRORS, and APPLICATION USAGE sections are
25919 revised to align with the ISO/IEC 9899:1999 standard.

25920 IEC 60559:1989 standard floating-point extensions over the ISO/IEC 9899:1999 standard are
25921 marked.

25922 **Issue 7**

25923 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0093 [68] and XSH/TC1-2008/0094 +
25924 [68] are applied.

25925 **NAME**25926 **erff, erfl** — error functions25927 **SYNOPSIS**25928 `#include <math.h>`25929 `float erff(float x);`25930 `long double erfl(long double x);`25931 **DESCRIPTION**25932 Refer to *erf()*.

25933 NAME

25934 `errno` — error return value

25935 SYNOPSIS

25936 `#include <errno.h>`

25937 DESCRIPTION

25938 The lvalue *errno* is used by many functions to return error values.

25939 Many functions provide an error number in *errno*, which has type `int` and is defined in
25940 **<errno.h>**. The value of *errno* shall be defined only after a call to a function for which it is
25941 explicitly stated to be set and until it is changed by the next function call or if the application
25942 assigns it a value. The value of *errno* should only be examined when it is indicated to be valid by
25943 a function's return value. Applications shall obtain the definition of *errno* by the inclusion of
25944 **<errno.h>**. No function in this volume of POSIX.1-2008 shall set *errno* to 0. The setting of *errno*
25945 after a successful call to a function is unspecified unless the description of that function specifies
25946 that *errno* shall not be modified.

25947 It is unspecified whether *errno* is a macro or an identifier declared with external linkage. If a
25948 macro definition is suppressed in order to access an actual object, or a program defines an
25949 identifier with the name *errno*, the behavior is undefined.

25950 The symbolic values stored in *errno* are documented in the ERRORS sections on all relevant
25951 pages.

25952 RETURN VALUE

25953 None.

25954 ERRORS

25955 None.

25956 EXAMPLES

25957 None.

25958 APPLICATION USAGE

25959 Previously both POSIX and X/Open documents were more restrictive than the ISO C standard
25960 in that they required *errno* to be defined as an external variable, whereas the ISO C standard
25961 required only that *errno* be defined as a modifiable lvalue with type `int`.

25962 An application that needs to examine the value of *errno* to determine the error should set it to 0
25963 before a function call, then inspect it before a subsequent function call.

25964 RATIONALE

25965 None.

25966 FUTURE DIRECTIONS

25967 None.

25968 SEE ALSO

25969 [Section 2.3](#)

25970 XBD **<errno.h>**

25971 CHANGE HISTORY

25972 First released in Issue 1. Derived from Issue 1 of the SVID.

25973 Issue 5

25974 The following sentence is deleted from the DESCRIPTION: “The value of *errno* is 0 at program
25975 start-up, but is never set to 0 by any XSI function”.

25976 The DESCRIPTION also no longer states that conforming implementations may support the
25977 declaration:

25978 `extern int errno;`

25979 **Issue 6**

25980 Obsolescent text regarding defining *errno* as:

25981 `extern int errno`

25982 is removed.

25983 Text regarding no function setting *errno* to zero to indicate an error is changed to no function
25984 shall set *errno* to zero. This is for alignment with the ISO/IEC 9899:1999 standard.

25985 IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/23 is applied, adding text to the
25986 DESCRIPTION stating that the setting of *errno* after a successful call to a function is unspecified
25987 unless the description of the function requires that it will not be modified.

25988 **NAME**

25989 environ, execl, execlx, execlp, execv, execve, execvp, fexecve — execute a file

25990 **SYNOPSIS**

```

25991 #include <unistd.h>

25992 extern char **environ;
25993 int execl(const char *path, const char *arg0, ... /*, (char *)0 */);
25994 int execlx(const char *path, const char *arg0, ... /*,
25995           (char *)0, char *const envp[] */);
25996 int execlp(const char *file, const char *arg0, ... /*, (char *)0 */);
25997 int execv(const char *path, char *const argv[]);
25998 int execve(const char *path, char *const argv[], char *const envp[]);
25999 int execvp(const char *file, char *const argv[]);
26000 int fexecve(int fd, char *const argv[], char *const envp[]);

```

26001 **DESCRIPTION**

26002 The *exec* family of functions shall replace the current process image with a new process image.
 26003 The new image shall be constructed from a regular, executable file called the *new process image*
 26004 *file*. There shall be no return from a successful *exec*, because the calling process image is overlaid
 26005 by the new process image.

26006 The *fexecve()* function shall be equivalent to the *execve()* function except that the file to be
 26007 executed is determined by the file descriptor *fd* instead of a pathname. The file offset of *fd* is
 26008 ignored.

26009 When a C-language program is executed as a result of a call to one of the *exec* family of
 26010 functions, it shall be entered as a C-language function call as follows:

```
26011 int main (int argc, char *argv[]);
```

26012 where *argc* is the argument count and *argv* is an array of character pointers to the arguments |
 26013 themselves. In addition, the following variable, which must be declared by the user if it is to be |
 26014 used directly:

```
26015 extern char **environ;
```

26016 is initialized as a pointer to an array of character pointers to the environment strings. The *argv*
 26017 and *environ* arrays are each terminated by a null pointer. The null pointer terminating the *argv*
 26018 array is not counted in *argc*.

26019 Applications can change the entire environment in a single operation by assigning the *environ* +
 26020 variable to point to an array of character pointers to the new environment strings. After +
 26021 assigning a new value to *environ*, applications should not rely on the new environment strings +
 26022 XSI remaining part of the environment, as a call to *getenv()*, *putenv()*, *setenv()*, *unsetenv()*, or any +
 26023 function that is dependent on an environment variable may, on noticing that *environ* has +
 26024 changed, copy the environment strings to a new array and assign *environ* to point to it. +

26025 Any application that directly modifies the pointers to which the *environ* variable points has +
 26026 undefined behavior. +

26027 Conforming multi-threaded applications shall not use the *environ* variable to access or modify
 26028 any environment variable while any other thread is concurrently modifying any environment
 26029 variable. A call to any function dependent on any environment variable shall be considered a
 26030 use of the *environ* variable to access that environment variable.

26031 The arguments specified by a program with one of the *exec* functions shall be passed on to the
 26032 new process image in the corresponding *main()* arguments.

26033 The argument *path* points to a pathname that identifies the new process image file.

26034 The argument *file* is used to construct a pathname that identifies the new process image file. If
 26035 the *file* argument contains a <slash> character, the *file* argument shall be used as the pathname
 26036 for this file. Otherwise, the path prefix for this file is obtained by a search of the directories
 26037 passed as the environment variable *PATH* (see XBD Chapter 8, on page 173). If this environment
 26038 variable is not present, the results of the search are implementation-defined.

26039 There are two distinct ways in which the contents of the process image file may cause the
 26040 execution to fail, distinguished by the setting of *errno* to either [ENOEXEC] or [EINVAL] (see the
 26041 ERRORS section). In the cases where the other members of the *exec* family of functions would
 26042 fail and set *errno* to [ENOEXEC], the *execlp()* and *execvp()* functions shall execute a command
 26043 interpreter and the environment of the executed command shall be as if the process invoked the
 26044 *sh* utility using *execl()* as follows:

26045 `execl(<shell path>, arg0, file, arg1, ..., (char *)0);`

26046 where <shell path> is an unspecified pathname for the *sh* utility, *file* is the process image file, and
 26047 for *execvp()*, where *arg0*, *arg1*, and so on correspond to the values passed to *execvp()* in *argv*[0],
 26048 *argv*[1], and so on.

26049 The arguments represented by *arg0*,... are pointers to null-terminated character strings. These
 26050 strings shall constitute the argument list available to the new process image. The list is
 26051 terminated by a null pointer. The argument *arg0* should point to a filename string that is
 26052 associated with the process being started by one of the *exec* functions.

26053 The argument *argv* is an array of character pointers to null-terminated strings. The application
 26054 shall ensure that the last member of this array is a null pointer. These strings shall constitute the
 26055 argument list available to the new process image. The value in *argv*[0] should point to a filename
 26056 string that is associated with the process being started by one of the *exec* functions.

26057 The argument *envp* is an array of character pointers to null-terminated strings. These strings
 26058 shall constitute the environment for the new process image. The *envp* array is terminated by a
 26059 null pointer.

26060 For those forms not containing an *envp* pointer (*execl()*, *execv()*, *execlp()*, and *execvp()*), the
 26061 environment for the new process image shall be taken from the external variable *environ* in the
 26062 calling process.

26063 The number of bytes available for the new process' combined argument and environment lists is
 26064 {ARG_MAX}. It is implementation-defined whether null terminators, pointers, and/or any
 26065 alignment bytes are included in this total.

26066 File descriptors open in the calling process image shall remain open in the new process image,
 26067 except for those whose close-on-exec flag FD_CLOEXEC is set. For those file descriptors that
 26068 remain open, all attributes of the open file description remain unchanged. For any file descriptor
 26069 that is closed for this reason, file locks are removed as a result of the close as described in *close()*.
 26070 Locks that are not removed by closing of file descriptors remain unchanged.

26071 If file descriptor 0, 1, or 2 would otherwise be closed after a successful call to one of the *exec*
 26072 family of functions, implementations may open an unspecified file for the file descriptor in the
 26073 new process image. If a standard utility or a conforming application is executed with file
 26074 descriptor 0 not open for reading or with file descriptor 1 or 2 not open for writing, the
 26075 environment in which the utility or application is executed shall be deemed non-conforming,
 26076 and consequently the utility or application might not behave as described in this standard.

26077 Directory streams open in the calling process image shall be closed in the new process image.

26078		The state of the floating-point environment in the initial thread of the new process image shall
26079		be set to the default.
26080		The state of conversion descriptors and message catalog descriptors in the new process image is
26081		undefined.
26082		For the new process image, the equivalent of:
26083		<code>setlocale(LC_ALL, "C")</code>
26084		shall be executed at start-up.
26085		Signals set to the default action (SIG_DFL) in the calling process image shall be set to the default
26086		action in the new process image. Except for SIGCHLD, signals set to be ignored (SIG_IGN) by
26087		the calling process image shall be set to be ignored by the new process image. Signals set to be
26088		caught by the calling process image shall be set to the default action in the new process image
26089		(see <signal.h>).
26090		If the SIGCHLD signal is set to be ignored by the calling process image, it is unspecified whether
26091		the SIGCHLD signal is set to be ignored or to the default action in the new process image.
26092	XSI	After a successful call to any of the <i>exec</i> functions, alternate signal stacks are not preserved and
26093		the SA_ONSTACK flag shall be cleared for all signals.
26094		After a successful call to any of the <i>exec</i> functions, any functions previously registered by the
26095		<i>atexit()</i> or <i>pthread_atfork()</i> functions are no longer registered.
26096	XSI	If the ST_NOSUID bit is set for the file system containing the new process image file, then the
26097		effective user ID, effective group ID, saved set-user-ID, and saved set-group-ID are unchanged
26098		in the new process image. Otherwise, if the set-user-ID mode bit of the new process image file is
26099		set, the effective user ID of the new process image shall be set to the user ID of the new process
26100		image file. Similarly, if the set-group-ID mode bit of the new process image file is set, the
26101		effective group ID of the new process image shall be set to the group ID of the new process
26102		image file. The real user ID, real group ID, and supplementary group IDs of the new process
26103		image shall remain the same as those of the calling process image. The effective user ID and
26104		effective group ID of the new process image shall be saved (as the saved set-user-ID and the
26105		saved set-group-ID) for use by <i>setuid()</i> .
26106	XSI	Any shared memory segments attached to the calling process image shall not be attached to the
26107		new process image.
26108		Any named semaphores open in the calling process shall be closed as if by appropriate calls to
26109		<i>sem_close()</i> .
26110	TYM	Any blocks of typed memory that were mapped in the calling process are unmapped, as if
26111		<i>munmap()</i> was implicitly called to unmap them.
26112	ML	Memory locks established by the calling process via calls to <i>mlockall()</i> or <i>mlock()</i> shall be
26113		removed. If locked pages in the address space of the calling process are also mapped into the
26114		address spaces of other processes and are locked by those processes, the locks established by the
26115		other processes shall be unaffected by the call by this process to the <i>exec</i> function. If the <i>exec</i>
26116		function fails, the effect on memory locks is unspecified.
26117		Memory mappings created in the process are unmapped before the address space is rebuilt for
26118		the new process image.
26119	SS	When the calling process image does not use the SCHED_FIFO, SCHED_RR, or
26120		SCHED_SPORADIC scheduling policies, the scheduling policy and parameters of the new
26121		process image and the initial thread in that new process image are implementation-defined.

26122	PS	When the calling process image uses the SCHED_FIFO, SCHED_RR, or SCHED_SPORADIC scheduling policies, the process policy and scheduling parameter settings shall not be changed by a call to an <i>exec</i> function. The initial thread in the new process image shall inherit the process scheduling policy and parameters. It shall have the default system contention scope, but shall inherit its allocation domain from the calling process image.
26123		
26124	TPS	
26125		
26126		
26127		Per-process timers created by the calling process shall be deleted before replacing the current process image with the new process image.
26128		
26129	MSG	All open message queue descriptors in the calling process shall be closed, as described in <i>mq_close()</i> .
26130		
26131		Any outstanding asynchronous I/O operations may be canceled. Those asynchronous I/O operations that are not canceled shall complete as if the <i>exec</i> function had not yet occurred, but any associated signal notifications shall be suppressed. It is unspecified whether the <i>exec</i> function itself blocks awaiting such I/O completion. In no event, however, shall the new process image created by the <i>exec</i> function be affected by the presence of outstanding asynchronous I/O operations at the time the <i>exec</i> function is called. Whether any I/O is canceled, and which I/O may be canceled upon <i>exec</i> , is implementation-defined.
26132		
26133		
26134		
26135		
26136		
26137		
26138	CPT	The new process image shall inherit the CPU-time clock of the calling process image. This inheritance means that the process CPU-time clock of the process being <i>exec</i> -ed shall not be reinitialized or altered as a result of the <i>exec</i> function other than to reflect the time spent by the process executing the <i>exec</i> function itself.
26139		
26140		
26141		
26142	TCT	The initial value of the CPU-time clock of the initial thread of the new process image shall be set to zero.
26143		
26144	OB TRC	If the calling process is being traced, the new process image shall continue to be traced into the same trace stream as the original process image, but the new process image shall not inherit the mapping of trace event names to trace event type identifiers that was defined by calls to the <i>posix_trace_eventid_open()</i> or the <i>posix_trace_trid_eventid_open()</i> functions in the calling process image.
26145		
26146		
26147		
26148		If the calling process is a trace controller process, any trace streams that were created by the calling process shall be shut down as described in the <i>posix_trace_shutdown()</i> function.
26149		
26150		
26151		The thread ID of the initial thread in the new process image is unspecified.
26152		The size and location of the stack on which the initial thread in the new process image runs is unspecified.
26153		
26154		The initial thread in the new process image shall have its cancellation type set to <code>PTHREAD_CANCEL_DEFERRED</code> and its cancellation state set to <code>PTHREAD_CANCEL_ENABLED</code> .
26155		
26156		
26157		The initial thread in the new process image shall have all thread-specific data values set to NULL and all thread-specific data keys shall be removed by the call to <i>exec</i> without running destructors.
26158		
26159		
26160		The initial thread in the new process image shall be joinable, as if created with the <i>detachstate</i> attribute set to <code>PTHREAD_CREATE_JOINABLE</code> .
26161		
26162		The new process shall inherit at least the following attributes from the calling process image:
26163	XSI	Nice value (see <i>nice()</i>)

26164	XSI	• <code>semadj</code> values (see <code>semop()</code>)
26165		• Process ID
26166		• Parent process ID
26167		• Process group ID
26168		• Session membership
26169		• Real user ID
26170		• Real group ID
26171		• Supplementary group IDs
26172		• Time left until an alarm clock signal (see <code>alarm()</code>)
26173		• Current working directory
26174		• Root directory
26175		• File mode creation mask (see <code>umask()</code>)
26176	XSI	• File size limit (see <code>getrlimit()</code> and <code>setrlimit()</code>)
26177		• Process signal mask (see <code>pthread_sigmask()</code>)
26178		• Pending signal (see <code>sigpending()</code>)
26179		• <code>tms_utime</code>, <code>tms_stime</code>, <code>tms_cutime</code>, and <code>tms_cstime</code> (see <code>times()</code>)
26180	XSI	• Resource limits
26181		• Controlling terminal
26182	XSI	• Interval timers
26183		The initial thread of the new process shall inherit at least the following attributes from the
26184		calling thread:
26185		• Signal mask (see <code>sigprocmask()</code> and <code>pthread_sigmask()</code>)
26186		• Pending signals (see <code>sigpending()</code>)
26187		All other process attributes defined in this volume of POSIX.1-2008 shall be inherited in the new
26188		process image from the old process image. All other thread attributes defined in this volume of
26189		POSIX.1-2008 shall be inherited in the initial thread in the new process image from the calling
26190		thread in the old process image. The inheritance of process or thread attributes not defined by
26191		this volume of POSIX.1-2008 is implementation-defined.
26192		A call to any <i>exec</i> function from a process with more than one thread shall result in all threads
26193		being terminated and the new executable image being loaded and executed. No destructor
26194		functions or cleanup handlers shall be called.
26195		Upon successful completion, the <i>exec</i> functions shall mark for update the last data access
26196		timestamp of the file. If an <i>exec</i> function failed but was able to locate the process image file,
26197		whether the last data access timestamp is marked for update is unspecified. Should the <i>exec</i>
26198		function succeed, the process image file shall be considered to have been opened with <code>open()</code> .
26199		The corresponding <code>close()</code> shall be considered to occur at a time after this open, but before
26200		process termination or successful completion of a subsequent call to one of the <i>exec</i> functions,
26201		<code>posix_spawn()</code> , or <code>posix_spawnp()</code> . The <code>argv[]</code> and <code>envp[]</code> arrays of pointers and the strings to
26202		which those arrays point shall not be modified by a call to one of the <i>exec</i> functions, except as a
26203		consequence of replacing the process image.

26204 XSI The saved resource limits in the new process image are set to be a copy of the process'
26205 corresponding hard and soft limits.

26206 RETURN VALUE

26207 If one of the *exec* functions returns to the calling process image, an error has occurred; the return
26208 value shall be `-1`, and *errno* shall be set to indicate the error.

26209 ERRORS

26210 The *exec* functions shall fail if:

26211 [E2BIG] The number of bytes used by the new process image's argument list and
26212 environment list is greater than the system-imposed limit of `{ARG_MAX}`
26213 bytes.

26214 [EACCES] The new process image file is not a regular file and the implementation does |
26215 not support execution of files of its type.

26216 [EINVAL] The new process image file has appropriate privileges and has a recognized
26217 executable binary format, but the system does not support execution of a file
26218 with this format.

26219 The *exec* functions, except for *fexecve()*, shall fail if: +

26220 [EACCES] Search permission is denied for a directory listed in the new process image +
26221 file's path prefix, or the new process image file denies execution permission.

26222 [ELOOP] A loop exists in symbolic links encountered during resolution of the *path* or *file*
26223 argument.

26224 [ENAMETOOLONG]

26225 The length of a component of a pathname is longer than `{NAME_MAX}`.

26226 [ENOENT] A component of *path* or *file* does not name an existing file or *path* or *file* is an
26227 empty string.

26228 [ENOTDIR] A component of the new process image file's path prefix names an existing file |
26229 that is neither a directory nor a symbolic link to a directory, or the new process |
26230 image file's pathname contains at least one non-`<slash>` character and ends
26231 with one or more trailing `<slash>` characters and the last pathname
26232 component names an existing file that is neither a directory nor a symbolic
26233 link to a directory.

26234 The *exec* functions, except for *execlp()* and *execvp()*, shall fail if:

26235 [ENOEXEC] The new process image file has the appropriate access permission but has an
26236 unrecognized format.

26237 The *fexecve()* function shall fail if:

26238 [EBADF] The *fd* argument is not a valid file descriptor open for executing.

26239 The *exec* functions may fail if:

26240 [ENOMEM] The new process image requires more memory than is allowed by the +
26241 hardware or system-imposed memory management constraints. +

26242 The *exec* functions, except for *fexecve()*, may fail if: +

26243 [ELOOP] More than `{SYMLOOP_MAX}` symbolic links were encountered during
26244 resolution of the *path* or *file* argument.

[ENAMETOOLONG]

The length of the *path* argument or the length of the pathname constructed from the *file* argument exceeds {PATH_MAX}, or pathname resolution of a symbolic link produced an intermediate result with a length that exceeds {PATH_MAX}.

[ETXTBSY]

The new process image file is a pure procedure (shared text) file that is currently open for writing by some process.

EXAMPLES

Using `execl()`

The following example executes the *ls* command, specifying the pathname of the executable (*/bin/ls*) and using arguments supplied directly to the command to produce single-column output.

```
#include <unistd.h>

int ret;
...
ret = execl ("/bin/ls", "ls", "-l", (char *)0);
```

Using `execle()`

The following example is similar to [Using `execl\(\)`](#). In addition, it specifies the environment for the new process image using the *env* argument.

```
#include <unistd.h>

int ret;
char *env[] = { "HOME=/usr/home", "LOGNAME=home", (char *)0 };
...
ret = execle ("/bin/ls", "ls", "-l", (char *)0, env);
```

Using `execlp()`

The following example searches for the location of the *ls* command among the directories specified by the *PATH* environment variable.

```
#include <unistd.h>

int ret;
...
ret = execlp ("ls", "ls", "-l", (char *)0);
```

Using `execv()`

The following example passes arguments to the *ls* command in the *cmd* array.

```
#include <unistd.h>

int ret;
char *cmd[] = { "ls", "-l", (char *)0 };
...
ret = execv ("/bin/ls", cmd);
```

Using `execve()`

The following example passes arguments to the `ls` command in the `cmd` array, and specifies the environment for the new process image using the `env` argument.

```
#include <unistd.h>

int ret;
char *cmd[] = { "ls", "-l", (char *)0 };
char *env[] = { "HOME=/usr/home", "LOGNAME=home", (char *)0 };
...
ret = execve ("/bin/ls", cmd, env);
```

Using `execvp()`

The following example searches for the location of the `ls` command among the directories specified by the `PATH` environment variable, and passes arguments to the `ls` command in the `cmd` array.

```
#include <unistd.h>

int ret;
char *cmd[] = { "ls", "-l", (char *)0 };
...
ret = execvp ("ls", cmd);
```

APPLICATION USAGE

As the state of conversion descriptors and message catalog descriptors in the new process image is undefined, conforming applications should not rely on their use and should close them prior to calling one of the *exec* functions.

Applications that require other than the default POSIX locale as the global locale in the new process image should call `setlocale()` with the appropriate parameters.

When assigning a new value to the *environ* variable, applications should ensure that the environment to which it will point contains at least the following:

1. Any implementation-defined variables required by the implementation to provide a conforming environment. See the `_CS_V7_ENV` entry in `<unistd.h>` and `confstr()` for details.
2. A value for `PATH` which finds conforming versions of all standard utilities before any other versions.

The same constraint applies to the `envp` array passed to `execle()` or `execve()`, in order to ensure that the new process image is invoked in a conforming environment.

Applications should not execute programs with file descriptor 0 not open for reading or with file descriptor 1 or 2 not open for writing, as this might cause the executed program to misbehave. In order not to pass on these file descriptors to an executed program, applications should not just close them but should reopen them on, for example, `/dev/null`. Some implementations may reopen them automatically, but applications should not rely on this being done.

If an application wants to perform a checksum test of the file being executed before executing it, the file will need to be opened with read permission to perform the checksum test.

Since execute permission is checked by `fexecve()`, the file description `fd` need not have been opened with the `O_EXEC` flag. However, if the file to be executed denies read and write permission for the process preparing to do the *exec*, the only way to provide the `fd` to `fexecve()`

will be to use the `O_EXEC` flag when opening *fd*. In this case, the application will not be able to perform a checksum test since it will not be able to read the contents of the file.

Note that when a file descriptor is opened with `O_RDONLY`, `O_RDWR`, or `O_WRONLY` mode, the file descriptor can be used to read, read and write, or write the file, respectively, even if the mode of the file changes after the file was opened. Using the `O_EXEC` open mode is different; *fexecve()* will ignore the mode that was used when the file descriptor was opened and the *exec* will fail if the mode of the file associated with *fd* does not grant execute permission to the calling process at the time *fexecve()* is called.

RATIONALE

Early proposals required that the value of *argc* passed to *main()* be “one or greater”. This was driven by the same requirement in drafts of the ISO C standard. In fact, historical implementations have passed a value of zero when no arguments are supplied to the caller of the *exec* functions. This requirement was removed from the ISO C standard and subsequently removed from this volume of POSIX.1-2008 as well. The wording, in particular the use of the word *should*, requires a Strictly Conforming POSIX Application to pass at least one argument to the *exec* function, thus guaranteeing that *argc* be one or greater when invoked by such an application. In fact, this is good practice, since many existing applications reference *argv[0]* without first checking the value of *argc*.

The requirement on a Strictly Conforming POSIX Application also states that the value passed as the first argument be a filename string associated with the process being started. Although some existing applications pass a pathname rather than a filename string in some circumstances, a filename string is more generally useful, since the common usage of *argv[0]* is in printing diagnostics. In some cases the filename passed is not the actual filename of the file; for example, many implementations of the *login* utility use a convention of prefixing a <hyphen> (‘-’) to the actual filename, which indicates to the command interpreter being invoked that it is a “login shell”.

Historically, there have been two ways that implementations can *exec* shell scripts.

One common historical implementation is that the *execl()*, *execv()*, *execle()*, and *execve()* functions return an `[ENOEXEC]` error for any file not recognizable as executable, including a shell script. When the *execlp()* and *execvp()* functions encounter such a file, they assume the file to be a shell script and invoke a known command interpreter to interpret such files. This is now required by POSIX.1-2008. These implementations of *execvp()* and *execlp()* only give the `[ENOEXEC]` error in the rare case of a problem with the command interpreter’s executable file. Because of these implementations, the `[ENOEXEC]` error is not mentioned for *execlp()* or *execvp()*, although implementations can still give it.

Another way that some historical implementations handle shell scripts is by recognizing the first two bytes of the file as the character string “#!” and using the remainder of the first line of the file as the name of the command interpreter to execute.

One potential source of confusion noted by the standard developers is over how the contents of a process image file affect the behavior of the *exec* family of functions. The following is a description of the actions taken:

1. If the process image file is a valid executable (in a format that is executable and valid and having appropriate privileges) for this system, then the system executes the file.
2. If the process image file has appropriate privileges and is in a format that is executable but not valid for this system (such as a recognized binary for another architecture), then this is an error and *errno* is set to `[EINVAL]` (see later RATIONALE on `[EINVAL]`).

3. If the process image file has appropriate privileges but is not otherwise recognized:
 - a. If this is a call to *execlp()* or *execvp()*, then they invoke a command interpreter assuming that the process image file is a shell script.
 - b. If this is not a call to *execlp()* or *execvp()*, then an error occurs and *errno* is set to [ENOEXEC].

Applications that do not require to access their arguments may use the form:

```
main(void)
```

as specified in the ISO C standard. However, the implementation will always provide the two arguments *argc* and *argv*, even if they are not used.

Some implementations provide a third argument to *main()* called *envp*. This is defined as a pointer to the environment. The ISO C standard specifies invoking *main()* with two arguments, so implementations must support applications written this way. Since this volume of POSIX.1-2008 defines the global variable *environ*, which is also provided by historical implementations and can be used anywhere that *envp* could be used, there is no functional need for the *envp* argument. Applications should use the *getenv()* function rather than accessing the environment directly via either *envp* or *environ*. Implementations are required to support the two-argument calling sequence, but this does not prohibit an implementation from supporting *envp* as an optional third argument.

This volume of POSIX.1-2008 specifies that signals set to SIG_IGN remain set to SIG_IGN, and that the new process image inherits the signal mask of the thread that called *exec* in the old process image. This is consistent with historical implementations, and it permits some useful functionality, such as the *nohup* command. However, it should be noted that many existing applications wrongly assume that they start with certain signals set to the default action and/or unblocked. In particular, applications written with a simpler signal model that does not include blocking of signals, such as the one in the ISO C standard, may not behave properly if invoked with some signals blocked. Therefore, it is best not to block or ignore signals across *execs* without explicit reason to do so, and especially not to block signals across *execs* of arbitrary (not closely cooperating) programs.

The *exec* functions always save the value of the effective user ID and effective group ID of the process at the completion of the *exec*, whether or not the set-user-ID or the set-group-ID bit of the process image file is set.

The statement about *argv[]* and *envp[]* being constants is included to make explicit to future writers of language bindings that these objects are completely constant. Due to a limitation of the ISO C standard, it is not possible to state that idea in standard C. Specifying two levels of *const-qualification* for the *argv[]* and *envp[]* parameters for the *exec* functions may seem to be the natural choice, given that these functions do not modify either the array of pointers or the characters to which the function points, but this would disallow existing correct code. Instead, only the array of pointers is noted as constant. The table of assignment compatibility for *dst=src* derived from the ISO C standard summarizes the compatibility:

<i>dst:</i>	char *[]	const char *[]	char *const[]	const char *const[]
<i>src:</i>				
char *[]	VALID	—	VALID	—
const char *[]	—	VALID	—	VALID
char * const []	—	—	VALID	—
const char *const[]	—	—	—	VALID

Since all existing code has a source type matching the first row, the column that gives the most

valid combinations is the third column. The only other possibility is the fourth column, but using it would require a cast on the *argv* or *envp* arguments. It is unfortunate that the fourth column cannot be used, because the declaration a non-expert would naturally use would be that in the second row.

The ISO C standard and this volume of POSIX.1-2008 do not conflict on the use of *environ*, but some historical implementations of *environ* may cause a conflict. As long as *environ* is treated in the same way as an entry point (for example, *fork()*), it conforms to both standards. A library can contain *fork()*, but if there is a user-provided *fork()*, that *fork()* is given precedence and no problem ensues. The situation is similar for *environ*: the definition in this volume of POSIX.1-2008 is to be used if there is no user-provided *environ* to take precedence. At least three implementations are known to exist that solve this problem.

[E2BIG] The limit {ARG_MAX} applies not just to the size of the argument list, but to the sum of that and the size of the environment list.

[EFAULT] Some historical systems return [EFAULT] rather than [ENOEXEC] when the new process image file is corrupted. They are non-conforming.

[EINVAL] This error condition was added to POSIX.1-2008 to allow an implementation to detect executable files generated for different architectures, and indicate this situation to the application. Historical implementations of shells, *execvp()*, and *execlp()* that encounter an [ENOEXEC] error will execute a shell on the assumption that the file is a shell script. This will not produce the desired effect when the file is a valid executable for a different architecture. An implementation may now choose to avoid this problem by returning [EINVAL] when a valid executable for a different architecture is encountered. Some historical implementations return [EINVAL] to indicate that the *path* argument contains a character with the high order bit set. The standard developers chose to deviate from historical practice for the following reasons:

1. The new utilization of [EINVAL] will provide some measure of utility to the user community.
2. Historical use of [EINVAL] is not acceptable in an internationalized operating environment.

[ENAMETOOLONG]

Since the file pathname may be constructed by taking elements in the *PATH* variable and putting them together with the filename, the [ENAMETOOLONG] error condition could also be reached this way.

[ETXTBSY] System V returns this error when the executable file is currently open for writing by some process. This volume of POSIX.1-2008 neither requires nor prohibits this behavior.

Other systems (such as System V) may return [EINTR] from *exec*. This is not addressed by this volume of POSIX.1-2008, but implementations may have a window between the call to *exec* and the time that a signal could cause one of the *exec* calls to return with [EINTR].

An explicit statement regarding the floating-point environment (as defined in the **<fenv.h>** header) was added to make it clear that the floating-point environment is set to its default when a call to one of the *exec* functions succeeds. The requirements for inheritance or setting to the default for other process and thread start-up functions is covered by more generic statements in their descriptions and can be summarized as follows:

26463 *posix_spawn()* Set to default.

26464 *fork()* Inherit.

26465 *pthread_create()* Inherit.

26466 The purpose of the *execve()* function is to enable executing a file which has been verified to be
 26467 the intended file. It is possible to actively check the file by reading from the file descriptor and be
 26468 sure that the file is not exchanged for another between the reading and the execution.
 26469 Alternatively, an function like *openat()* can be used to open a file which has been found by
 26470 reading the content of a directory using *readdir()*.

26471 FUTURE DIRECTIONS

26472 None.

26473 SEE ALSO

26474 *alarm()*, *atexit()*, *chmod()*, *close()*, *confstr()*, *exit()*, *fcntl()*, *fork()*, *fstatvfs()*, *getenv()*, *getitimer()*,
 26475 *getrlimit()*, *mknod()*, *mmap()*, *nice()*, *open()*, *posix_spawn()*, *posix_trace_create()*,
 26476 *posix_trace_event()*, *posix_trace_eventid_equal()*, *pthread_atfork()*, *pthread_sigmask()*, *putenv()*,
 26477 *readdir()*, *semop()*, *setlocale()*, *shmat()*, *sigaction()*, *sigaltstack()*, *sigpending()*, *system()*, *times()*,
 26478 *ulimit()*, *umask()*

26479 XBD Chapter 8 (on page 173), [<unistd.h>](#)

26480 CHANGE HISTORY

26481 First released in Issue 1. Derived from Issue 1 of the SVID.

26482 Issue 5

26483 The DESCRIPTION is updated for alignment with the POSIX Realtime Extension and the POSIX
 26484 Threads Extension.

26485 Large File Summit extensions are added.

26486 Issue 6

26487 The following new requirements on POSIX implementations derive from alignment with the
 26488 Single UNIX Specification:

- 26489 • In the DESCRIPTION, behavior is defined for when the process image file is not a valid
 26490 executable.
- 26491 • In this version, `_POSIX_SAVED_IDS` is mandated, thus the effective user ID and effective
 26492 group ID of the new process image shall be saved (as the saved set-user-ID and the saved
 26493 set-group-ID) for use by the *setuid()* function.
- 26494 • The [ELOOP] mandatory error condition is added.
- 26495 • A second [ENAMETOOLONG] is added as an optional error condition.
- 26496 • The [ETXTBSY] optional error condition is added.

26497 The following changes were made to align with the IEEE P1003.1a draft standard:

- 26498 • The [EINVAL] mandatory error condition is added.
- 26499 • The [ELOOP] optional error condition is added.

26500 The description of CPU-time clock semantics is added for alignment with IEEE Std 1003.1d-1999.

26501 The DESCRIPTION is updated for alignment with IEEE Std 1003.1j-2000 by adding semantics
 26502 for typed memory.

26503 The normative text is updated to avoid use of the term “must” for application requirements.

26504	The description of tracing semantics is added for alignment with IEEE Std 1003.1q-2000.
26505	IEEE PASC Interpretation 1003.1 #132 is applied.
26506	The DESCRIPTION is updated to make it explicit that the floating-point environment in the new
26507	process image is set to the default.
26508	The DESCRIPTION and RATIONALE are updated to include clarifications of how the contents
26509	of a process image file affect the behavior of the <i>exec</i> functions.
26510	IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/15 is applied, adding a new paragraph to
26511	the DESCRIPTION and text to the end of the APPLICATION USAGE section. This change
26512	addresses a security concern, where implementations may want to reopen file descriptors 0, 1,
26513	and 2 for programs with the set-user-id or set-group-id file mode bits calling the <i>exec</i> family of
26514	functions.
26515	IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/24 is applied, applying changes to the
26516	DESCRIPTION, addressing which attributes are inherited by threads, and behavioral
26517	requirements for threads attributes.
26518	IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/25 is applied, updating text in the
26519	RATIONALE from “the process signal mask be unchanged across an <i>exec</i> ” to “the new process
26520	image inherits the signal mask of the thread that called <i>exec</i> in the old process image”.
26521	Issue 7
26522	Austin Group Interpretation 1003.1-2001 #047 is applied, adding the description of <code>_CS_V7_ENV</code>
26523	to the APPLICATION USAGE.
26524	Austin Group Interpretation 1003.1-2001 #143 is applied.
26525	The <i>fexecve()</i> function is added from The Open Group Technical Standard, 2006, Extended API
26526	Set Part 2.
26527	Functionality relating to the Asynchronous Input and Output, Memory Mapped Files, Threads,
26528	and Timers options is moved to the Base.
26529	Changes are made related to support for finegrained timestamps.
26530	POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0095 [386], XSH/TC1-2008/0096 [167], +
26531	XSH/TC1-2008/0097 [291], XSH/TC1-2008/0098 [173], XSH/TC1-2008/0099 [296], +
26532	XSH/TC1-2008/00100 [324], XSH/TC1-2008/00101 [296], XSH/TC1-2008/00102 [302], +
26533	XSH/TC1-2008/00103 [167], XSH/TC1-2008/00104 [173], and XSH/TC1-2008/00105 [291,429] +
26534	are applied.

26535 **NAME**

26536 exit — terminate a process

26537 **SYNOPSIS**

26538 #include <stdlib.h>

26539 void exit(int status);

26540 **DESCRIPTION**

26541 CX The functionality described on this reference page is aligned with the ISO C standard. Any
 26542 conflict between the requirements described here and the ISO C standard is unintentional. This
 26543 volume of POSIX.1-2008 defers to the ISO C standard.

26544 CX The value of *status* may be 0, EXIT_SUCCESS, EXIT_FAILURE, or any other value, though only
 26545 the least significant 8 bits (that is, *status* & 0377) shall be available to a waiting parent process.

26546 The *exit()* function shall first call all functions registered by *atexit()*, in the reverse order of their
 26547 registration, except that a function is called after any previously registered functions that had
 26548 already been called at the time it was registered. Each function is called as many times as it was
 26549 registered. If, during the call to any such function, a call to the *longjmp()* function is made that
 26550 would terminate the call to the registered function, the behavior is undefined.

26551 If a function registered by a call to *atexit()* fails to return, the remaining registered functions shall
 26552 not be called and the rest of the *exit()* processing shall not be completed. If *exit()* is called more
 26553 than once, the behavior is undefined.

26554 The *exit()* function shall then flush all open streams with unwritten buffered data and close all
 26555 CX open streams. Finally, the process shall be terminated with the same consequences as described
 26556 in [Consequences of Process Termination](#) (on page 549).

26557 **RETURN VALUE**26558 The *exit()* function does not return.26559 **ERRORS**

26560 No errors are defined.

26561 **EXAMPLES**

26562 None.

26563 **APPLICATION USAGE**

26564 None.

26565 **RATIONALE**26566 See *_Exit()*.26567 **FUTURE DIRECTIONS**

26568 None.

26569 **SEE ALSO**26570 *_Exit()*, *atexit()*, *exec*, *longjmp()*, *tmpfile()*26571 XBD [<stdlib.h>](#)26572 **CHANGE HISTORY**26573 **Issue 7**

26574 Austin Group Interpretation 1003.1-2001 #031 is applied, separating the *_Exit()* and *_exit()*
 26575 functions from the *exit()* function.

26576 Austin Group Interpretation 1003.1-2001 #085 is applied.

26577 **NAME**26578 `exp, expf, expl` — exponential function26579 **SYNOPSIS**

```
26580 #include <math.h>
26581 double exp(double x);
26582 float expf(float x);
26583 long double expl(long double x);
```

26584 **DESCRIPTION**

26585 CX The functionality described on this reference page is aligned with the ISO C standard. Any
 26586 conflict between the requirements described here and the ISO C standard is unintentional. This
 26587 volume of POSIX.1-2008 defers to the ISO C standard.

26588 These functions shall compute the base-*e* exponential of *x*.

26589 An application wishing to check for error situations should set *errno* to zero and call
 26590 *feclearexcept*(FE_ALL_EXCEPT) before calling these functions. On return, if *errno* is non-zero or
 26591 *fetestexcept*(FE_INVALID | FE_DIVBYZERO | FE_OVERFLOW | FE_UNDERFLOW) is non-
 26592 zero, an error has occurred.

26593 **RETURN VALUE**

26594 Upon successful completion, these functions shall return the exponential value of *x*.

26595 If the correct value would cause overflow, a range error shall occur and *exp()*, *expf()*, and *expl()*
 26596 shall return the value of the macro HUGE_VAL, HUGE_VALF, and HUGE_VALL, respectively.

26597 MXX If the correct value would cause underflow, and is not representable, a range error may occur, |
 26598 MXX and *exp()*, *expf()*, and *expl()* shall return 0.0, or (if the IEC 60559 Floating-Point option is not |
 26599 supported) an implementation-defined value no greater in magnitude than DBL_MIN, |
 26600 FLT_MIN, and LDBL_MIN, respectively.

26601 MX If *x* is NaN, a NaN shall be returned.

26602 If *x* is ± 0 , 1 shall be returned.

26603 If *x* is $-\text{Inf}$, +0 shall be returned.

26604 If *x* is $+\text{Inf}$, *x* shall be returned.

26605 MXX If the correct value would cause underflow, and is representable, a range error may occur and
 26606 the correct value shall be returned.

26607 **ERRORS**

26608 These functions shall fail if:

26609 Range Error The result overflows.

26610 If the integer expression (*math_errhandling* & MATH_ERRNO) is non-zero,
 26611 then *errno* shall be set to [ERANGE]. If the integer expression
 26612 (*math_errhandling* & MATH_ERREXCEPT) is non-zero, then the overflow
 26613 floating-point exception shall be raised.

26614 These functions may fail if:

26615 Range Error The result underflows.

26616 If the integer expression (*math_errhandling* & MATH_ERRNO) is non-zero,
 26617 then *errno* shall be set to [ERANGE]. If the integer expression
 26618 (*math_errhandling* & MATH_ERREXCEPT) is non-zero, then the underflow
 26619 floating-point exception shall be raised.

26620 **EXAMPLES**26621 **Computing the Density of the Standard Normal Distribution**

26622 This function shows an implementation for the density of the standard normal distribution
 26623 using `exp()`. This example uses the constant `M_PI` which is part of the XSI option.

```
26624 #include <math.h>
26625 double
26626 normal_density (double x)
26627 {
26628     return exp(-x*x/2) / sqrt (2*M_PI);
26629 }
```

26630 **APPLICATION USAGE**

26631 Note that for IEEE Std 754-1985 **double**, $709.8 < x$ implies `exp(x)` has overflowed. The value
 26632 $x < -708.4$ implies `exp(x)` has underflowed.

26633 On error, the expressions `(math_errhandling & MATH_ERRNO)` and `(math_errhandling &`
 26634 `MATH_ERREXCEPT)` are independent of each other, but at least one of them must be non-zero.

26635 **RATIONALE**

26636 None.

26637 **FUTURE DIRECTIONS**

26638 None.

26639 **SEE ALSO**

26640 [`feclearexcept\(\)`](#), [`fetestexcept\(\)`](#), [`isnan\(\)`](#), [`log\(\)`](#)

26641 XBD [Section 4.19](#) (on page 116), [`<math.h>`](#)

26642 **CHANGE HISTORY**

26643 First released in Issue 1. Derived from Issue 1 of the SVID.

26644 **Issue 5**

26645 The DESCRIPTION is updated to indicate how an application should check for an error. This
 26646 text was previously published in the APPLICATION USAGE section.

26647 **Issue 6**

26648 The `expf()` and `expl()` functions are added for alignment with the ISO/IEC 9899:1999 standard.

26649 The DESCRIPTION, RETURN VALUE, ERRORS, and APPLICATION USAGE sections are
 26650 revised to align with the ISO/IEC 9899:1999 standard.

26651 IEC 60559:1989 standard floating-point extensions over the ISO/IEC 9899:1999 standard are
 26652 marked.

26653 IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/26 is applied, adding the example to the
 26654 EXAMPLES section.

26655 **Issue 7**

26656 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0106 [68] and XSH/TC1-2008/0107 +
 26657 [68] are applied.

26658 **NAME**

26659 exp2, exp2f, exp2l — exponential base 2 functions

26660 **SYNOPSIS**

26661 #include <math.h>

26662 double exp2(double x);

26663 float exp2f(float x);

26664 long double exp2l(long double x);

26665 **DESCRIPTION**

26666 **CX** The functionality described on this reference page is aligned with the ISO C standard. Any
 26667 conflict between the requirements described here and the ISO C standard is unintentional. This
 26668 volume of POSIX.1-2008 defers to the ISO C standard.

26669 These functions shall compute the base-2 exponential of x .

26670 An application wishing to check for error situations should set *errno* to zero and call
 26671 *feclearexcept*(FE_ALL_EXCEPT) before calling these functions. On return, if *errno* is non-zero or
 26672 *fetestexcept*(FE_INVALID | FE_DIVBYZERO | FE_OVERFLOW | FE_UNDERFLOW) is non-
 26673 zero, an error has occurred.

26674 **RETURN VALUE**26675 Upon successful completion, these functions shall return 2^x .

26676 If the correct value would cause overflow, a range error shall occur and *exp2()*, *exp2f()*, and
 26677 *exp2l()* shall return the value of the macro HUGE_VAL, HUGE_VALF, and HUGE_VALL,
 26678 respectively.

26679 **MX** If the correct value would cause underflow, and is not representable, a range error may occur, |
 26680 **MX** and *exp2()*, *exp2f()*, and *exp2l()* shall return 0.0, or (if the IEC 60559 Floating-Point option is not |
 26681 supported) an implementation-defined value no greater in magnitude than DBL_MIN, |
 26682 FLT_MIN, and LDBL_MIN, respectively.

26683 **MX** If x is NaN, a NaN shall be returned.26684 If x is ± 0 , 1 shall be returned.26685 If x is $-\text{Inf}$, $+\text{0}$ shall be returned.26686 If x is $+\text{Inf}$, x shall be returned.

26687 **MX** If the correct value would cause underflow, and is representable, a range error may occur and
 26688 the correct value shall be returned.

26689 **ERRORS**

26690 These functions shall fail if:

26691 Range Error The result overflows.

26692 If the integer expression (*math_errhandling* & MATH_ERRNO) is non-zero,
 26693 then *errno* shall be set to [ERANGE]. If the integer expression
 26694 (*math_errhandling* & MATH_ERREXCEPT) is non-zero, then the overflow
 26695 floating-point exception shall be raised.

26696 These functions may fail if:

26697 Range Error The result underflows.

26698 If the integer expression (*math_errhandling* & MATH_ERRNO) is non-zero,
 26699 then *errno* shall be set to [ERANGE]. If the integer expression
 26700 (*math_errhandling* & MATH_ERREXCEPT) is non-zero, then the underflow

26701 floating-point exception shall be raised.

26702 **EXAMPLES**

26703 None.

26704 **APPLICATION USAGE**

26705 For IEEE Std 754-1985 **double**, $1024 \leq x$ implies $\exp2(x)$ has overflowed. The value $x < -1022$
26706 implies $\exp(x)$ has underflowed.

26707 On error, the expressions $(\text{math\_errhandling} \ \& \ \text{MATH\_ERRNO})$ and $(\text{math\_errhandling} \ \& \ \text{MATH\_ERREXCEPT})$
26708 are independent of each other, but at least one of them must be non-zero.

26709 **RATIONALE**

26710 None.

26711 **FUTURE DIRECTIONS**

26712 None.

26713 **SEE ALSO**

26714 [*exp\(\)*](#), [*feclearexcept\(\)*](#), [*fetestexcept\(\)*](#), [*isnan\(\)*](#), [*log\(\)*](#)

26715 XBD [Section 4.19](#) (on page 116), [**<math.h>**](#)

26716 **CHANGE HISTORY**

26717 First released in Issue 6. Derived from the ISO/IEC 9899: 1999 standard.

26718 **Issue 7**

26719 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0108 [68] and XSH/TC1-2008/0109 +
26720 [68] are applied.

26721 **NAME**

26722 expm1, expm1f, expm1l — compute exponential functions

26723 **SYNOPSIS**

```
26724 #include <math.h>
26725 double expm1(double x);
26726 float expm1f(float x);
26727 long double expm1l(long double x);
```

26728 **DESCRIPTION**

26729 CX The functionality described on this reference page is aligned with the ISO C standard. Any
 26730 conflict between the requirements described here and the ISO C standard is unintentional. This
 26731 volume of POSIX.1-2008 defers to the ISO C standard.

26732 These functions shall compute $e^x - 1.0$.

26733 An application wishing to check for error situations should set *errno* to zero and call
 26734 *feclearexcept*(FE_ALL_EXCEPT) before calling these functions. On return, if *errno* is non-zero or
 26735 *fetestexcept*(FE_INVALID | FE_DIVBYZERO | FE_OVERFLOW | FE_UNDERFLOW) is non-
 26736 zero, an error has occurred.

26737 **RETURN VALUE**26738 Upon successful completion, these functions return $e^x - 1.0$.

26739 If the correct value would cause overflow, a range error shall occur and *expm1()*, *expm1f()*, and
 26740 *expm1l()* shall return the value of the macro HUGE_VAL, HUGE_VALF, and HUGE_VALL,
 26741 respectively.

26742 MX If *x* is NaN, a NaN shall be returned.26743 If *x* is ± 0 , ± 0 shall be returned.26744 If *x* is $-\text{Inf}$, -1 shall be returned.26745 If *x* is $+\text{Inf}$, *x* shall be returned.

26746 If *x* is subnormal, a range error may occur
 26747 MXX and *x* should be returned.

26748 MX If *x* is not returned, *expm1()*, *expm1f()*, and *expm1l()* shall return an implementation-defined
 26749 value no greater in magnitude than DBL_MIN, FLT_MIN, and LDBL_MIN, respectively.

26750 **ERRORS**

26751 These functions shall fail if:

26752 Range Error The result overflows.

26753 If the integer expression (*math_errhandling* & MATH_ERRNO) is non-zero,
 26754 then *errno* shall be set to [ERANGE]. If the integer expression
 26755 (*math_errhandling* & MATH_ERREXCEPT) is non-zero, then the overflow
 26756 floating-point exception shall be raised.

26757 These functions may fail if:

26758 MX Range Error The value of *x* is subnormal.

26759 If the integer expression (*math_errhandling* & MATH_ERRNO) is non-zero,
 26760 then *errno* shall be set to [ERANGE]. If the integer expression
 26761 (*math_errhandling* & MATH_ERREXCEPT) is non-zero, then the underflow
 26762 floating-point exception shall be raised.

EXAMPLES

None.

APPLICATION USAGE

The value of $\text{expm1}(x)$ may be more accurate than $\exp(x)-1.0$ for small values of x .

The $\text{expm1}()$ and $\text{log1p}()$ functions are useful for financial calculations of $((1+x)^n-1)/x$, namely:

```
expm1(n * log1p(x)) / x
```

when x is very small (for example, when calculating small daily interest rates). These functions also simplify writing accurate inverse hyperbolic functions.

For IEEE Std 754-1985 **double**, $709.8 < x$ implies $\text{expm1}(x)$ has overflowed.

On error, the expressions $(\text{math\_errhandling} \ \& \ \text{MATH\_ERRNO})$ and $(\text{math\_errhandling} \ \& \ \text{MATH\_ERREXCEPT})$ are independent of each other, but at least one of them must be non-zero.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

[*exp\(\)*](#), [*feclearexcept\(\)*](#), [*fetestexcept\(\)*](#), [*ilogb\(\)*](#), [*log1p\(\)*](#)

XBD [Section 4.19](#) (on page 116), [**<math.h>**](#)

CHANGE HISTORY

First released in Issue 4, Version 2.

Issue 5

Moved from X/OPEN UNIX extension to BASE.

Issue 6

The $\text{expm1f}()$ and $\text{expm1l}()$ functions are added for alignment with the ISO/IEC 9899:1999 standard.

The $\text{expm1}()$ function is no longer marked as an extension.

The DESCRIPTION, RETURN VALUE, ERRORS, and APPLICATION USAGE sections are revised to align with the ISO/IEC 9899:1999 standard.

IEC 60559:1989 standard floating-point extensions over the ISO/IEC 9899:1999 standard are marked.

Issue 7

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0110 [68] is applied.

+

26795 **NAME**

26796 fabs, fabsf, fabsl — absolute value function

26797 **SYNOPSIS**

```

26798       #include <math.h>

26799       double fabs(double x);
26800       float fabsf(float x);
26801       long double fabsl(long double x);

```

26802 **DESCRIPTION**

26803 CX The functionality described on this reference page is aligned with the ISO C standard. Any
 26804 conflict between the requirements described here and the ISO C standard is unintentional. This
 26805 volume of POSIX.1-2008 defers to the ISO C standard.

26806 These functions shall compute the absolute value of their argument x , $|x|$.26807 **RETURN VALUE**26808 Upon successful completion, these functions shall return the absolute value of x .26809 MX If x is NaN, a NaN shall be returned.26810 If x is ± 0 , $+0$ shall be returned.26811 If x is $\pm \text{Inf}$, $+\text{Inf}$ shall be returned.26812 **ERRORS**

26813 No errors are defined.

26814 **EXAMPLES**26815 **Computing the 1-Norm of a Floating-Point Vector**26816 This example shows the use of *fabs()* to compute the 1-norm of a vector defined as follows:26817 norm1(v) = $|v[0]| + |v[1]| + \dots + |v[n-1]|$

26818 where $|x|$ denotes the absolute value of x , n denotes the vector's dimension $v[i]$ denotes the i -th
 26819 component of v ($0 \leq i < n$).

```

26820       #include <math.h>

26821       double
26822       norm1(const double v[], const int n)
26823       {
26824           int       i;
26825           double   nl_v; /* 1-norm of v */

26826           nl_v = 0;
26827           for (i=0; i<n; i++) {
26828               nl_v += fabs (v[i]);
26829           }

26830           return nl_v;
26831       }

```

26832 **APPLICATION USAGE**

26833 None.

26834 **RATIONALE**

26835 None.

26836 **FUTURE DIRECTIONS**

26837 None.

26838 **SEE ALSO**26839 *isnan()*

26840 XBD <math.h>

26841 **CHANGE HISTORY**

26842 First released in Issue 1. Derived from Issue 1 of the SVID.

26843 **Issue 5**26844 The DESCRIPTION is updated to indicate how an application should check for an error. This
26845 text was previously published in the APPLICATION USAGE section.26846 **Issue 6**26847 The *fabsf()* and *fabsl()* functions are added for alignment with the ISO/IEC 9899:1999 standard.26848 The DESCRIPTION, RETURN VALUE, ERRORS, and APPLICATION USAGE sections are
26849 revised to align with the ISO/IEC 9899:1999 standard.26850 IEC 60559:1989 standard floating-point extensions over the ISO/IEC 9899:1999 standard are
26851 marked.26852 IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/27 is applied, adding the example to the
26853 EXAMPLES section.

26854 **NAME**

26855 faccessat — determine accessibility of a file relative to directory file descriptor

26856 **SYNOPSIS**

26857 #include <unistd.h>

26858 int faccessat(int *fd*, const char **path*, int *amode*, int *flag*);26859 **DESCRIPTION**26860 Refer to *access()*.

NAME

fattach — attach a STREAMS-based file descriptor to a file in the file system name space (STREAMS)

SYNOPSIS

```
#include <stropts.h>

int fattach(int fildes, const char *path);
```

DESCRIPTION

The *fattach()* function shall attach a STREAMS-based file descriptor to a file, effectively associating a pathname with *fildes*. The application shall ensure that the *fildes* argument is a valid open file descriptor associated with a STREAMS file. The *path* argument points to a pathname of an existing file. The application shall have appropriate privileges or be the owner of the file named by *path* and have write permission. A successful call to *fattach()* shall cause all pathnames that name the file named by *path* to name the STREAMS file associated with *fildes*, until the STREAMS file is detached from the file. A STREAMS file can be attached to more than one file and can have several pathnames associated with it.

The attributes of the named STREAMS file shall be initialized as follows: the permissions, user ID, group ID, and times are set to those of the file named by *path*, the number of links is set to 1, and the size and device identifier are set to those of the STREAMS file associated with *fildes*. If any attributes of the named STREAMS file are subsequently changed (for example, by *chmod()*), neither the attributes of the underlying file nor the attributes of the STREAMS file to which *fildes* refers shall be affected.

File descriptors referring to the underlying file, opened prior to an *fattach()* call, shall continue to refer to the underlying file.

RETURN VALUE

Upon successful completion, *fattach()* shall return 0. Otherwise, -1 shall be returned and *errno* set to indicate the error.

ERRORS

The *fattach()* function shall fail if:

[EACCES] Search permission is denied for a component of the path prefix, or the process is the owner of *path* but does not have write permissions on the file named by *path*.

[EBADF] The *fildes* argument is not a valid open file descriptor.

[EBUSY] The file named by *path* is currently a mount point or has a STREAMS file attached to it.

[ELOOP] A loop exists in symbolic links encountered during resolution of the *path* argument.

[ENAMETOOLONG]

The length of a component of a pathname is longer than {NAME_MAX}.

[ENOENT] A component of *path* does not name an existing file or *path* is an empty string.

[ENOTDIR] A component of the path prefix names an existing file that is neither a directory nor a symbolic link to a directory, or the *path* argument contains at least one non-`<slash>` character and ends with one or more trailing `<slash>` characters.

26904	[EPERM]	The effective user ID of the process is not the owner of the file named by <i>path</i>
26905		and the process does not have appropriate privileges.
26906		The <i>fattach()</i> function may fail if:
26907	[EINVAL]	The <i>fildev</i> argument does not refer to a STREAMS file.
26908	[ELOOP]	More than {SYMLOOP_MAX} symbolic links were encountered during
26909		resolution of the <i>path</i> argument.
26910	[ENAMETOOLONG]	
26911		The length of a pathname exceeds {PATH_MAX}, or pathname resolution of a
26912		symbolic link produced an intermediate result with a length that exceeds
26913		{PATH_MAX}.
26914	[EXDEV]	A link to a file on another file system was attempted.

26915 EXAMPLES

26916 Attaching a File Descriptor to a File

26917 In the following example, *fd* refers to an open STREAMS file. The call to *fattach()* associates this
 26918 STREAM with the file **/tmp/named-STREAM**, such that any future calls to open **/tmp/named-**
 26919 **STREAM**, prior to breaking the attachment via a call to *fdetach()*, will instead create a new file
 26920 handle referring to the STREAMS file associated with *fd*.

```
26921 #include <stropts.h>
26922 ...
26923     int fd;
26924     char *pathname = "/tmp/named-STREAM";
26925     int ret;
26926
26927     ret = fattach(fd, pathname);
```

26927 APPLICATION USAGE

26928 The *fattach()* function behaves similarly to the traditional *mount()* function in the way a file is
 26929 temporarily replaced by the root directory of the mounted file system. In the case of *fattach()*, the
 26930 replaced file need not be a directory and the replacing file is a STREAMS file.

26931 RATIONALE

26932 The file attributes of a file which has been the subject of an *fattach()* call are specifically set
 26933 because of an artifact of the original implementation. The internal mechanism was the same as
 26934 for the *mount()* function. Since *mount()* is typically only applied to directories, the effects when
 26935 applied to a regular file are a little surprising, especially as regards the link count which rigidly
 26936 remains one, even if there were several links originally and despite the fact that all original links
 26937 refer to the STREAM as long as the *fattach()* remains in effect.

26938 FUTURE DIRECTIONS

26939 The *fattach()* function may be removed in a future version.

26940 SEE ALSO

26941 *fdetach()*, *isastream()*

26942 XBD **<stropts.h>**

26943 CHANGE HISTORY

26944 First released in Issue 4, Version 2.

Issue 5

- 26945 Moved from X/OPEN UNIX extension to BASE.
26946
26947 The [EXDEV] error is added to the list of optional errors in the ERRORS section.

Issue 6

- 26948 This function is marked as part of the XSI STREAMS Option Group.
26949
26950 The normative text is updated to avoid use of the term “must” for application requirements.
26951 The wording of the mandatory [ELOOP] error condition is updated, and a second optional
26952 [ELOOP] error condition is added.

Issue 7

- 26953 Austin Group Interpretation 1003.1-2001 #143 is applied.
26954
26955 The *fattach()* function is marked obsolescent.
26956 The [ENOTDIR] error condition is clarified to cover the condition where the last component of a
26957 pathname exists but is not a directory or a symbolic link to a directory.
26958 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0111 [146,324] and +
26959 XSH/TC1-2008/0112 [291] are applied.

26960 **NAME**

26961 fchdir — change working directory

26962 **SYNOPSIS**

26963 #include <unistd.h>

26964 int fchdir(int *fildev*);26965 **DESCRIPTION**26966 The *fchdir()* function shall be equivalent to *chdir()* except that the directory that is to be the new
26967 current working directory is specified by the file descriptor *fildev*.26968 A conforming application can obtain a file descriptor for a file of type directory using *open()*,
26969 provided that the file status flags and access modes do not contain O_WRONLY or O_RDWR.26970 **RETURN VALUE**26971 Upon successful completion, *fchdir()* shall return 0. Otherwise, it shall return -1 and set *errno* to
26972 indicate the error. On failure the current working directory shall remain unchanged.26973 **ERRORS**26974 The *fchdir()* function shall fail if:26975 [EACCES] Search permission is denied for the directory referenced by *fildev*.26976 [EBADF] The *fildev* argument is not an open file descriptor.26977 [ENOTDIR] The open file descriptor *fildev* does not refer to a directory.26978 The *fchdir()* may fail if:26979 [EINTR] A signal was caught during the execution of *fchdir()*.

26980 [EIO] An I/O error occurred while reading from or writing to the file system.

26981 **EXAMPLES**

26982 None.

26983 **APPLICATION USAGE**

26984 None.

26985 **RATIONALE**

26986 None.

26987 **FUTURE DIRECTIONS**

26988 None.

26989 **SEE ALSO**26990 *chdir()*, *dirfd()*

26991 XBD <unistd.h>

26992 **CHANGE HISTORY**

26993 First released in Issue 4, Version 2.

26994 **Issue 5**

26995 Moved from X/OPEN UNIX extension to BASE.

26996 **Issue 7**26997 The *fchdir()* function is moved from the XSI option to the Base.

26998 **NAME**

26999 fchmod — change mode of a file

27000 **SYNOPSIS**

27001 #include <sys/stat.h>

27002 int fchmod(int *fildes*, mode_t *mode*);27003 **DESCRIPTION**27004 The *fchmod()* function shall be equivalent to *chmod()* except that the file whose permissions are
27005 changed is specified by the file descriptor *fildes*.27006 SHM If *fildes* references a shared memory object, the *fchmod()* function need only affect the S_IRUSR,
27007 S_IWUSR, S_IRGRP, S_IWGRP, S_IROTH, and S_IWOTH file permission bits.27008 TYM If *fildes* references a typed memory object, the behavior of *fchmod()* is unspecified.27009 If *fildes* refers to a socket, the behavior of *fchmod()* is unspecified.27010 OB XSR If *fildes* refers to a STREAM (which is *fattach()*-ed into the file system name space) the call
27011 returns successfully, doing nothing.27012 **RETURN VALUE**27013 Upon successful completion, *fchmod()* shall return 0. Otherwise, it shall return -1 and set *errno* to
27014 indicate the error.27015 **ERRORS**27016 The *fchmod()* function shall fail if:27017 [EBADF] The *fildes* argument is not an open file descriptor.27018 [EPERM] The effective user ID does not match the owner of the file and the process does
27019 not have appropriate privileges.27020 [EROFS] The file referred to by *fildes* resides on a read-only file system.27021 The *fchmod()* function may fail if:27022 XSI [EINTR] The *fchmod()* function was interrupted by a signal.27023 XSI [EINVAL] The value of the *mode* argument is invalid.27024 [EINVAL] The *fildes* argument refers to a pipe and the implementation disallows
27025 execution of *fchmod()* on a pipe.27026 **EXAMPLES**27027 **Changing the Current Permissions for a File**27028 The following example shows how to change the permissions for a file named **/home/cnd/mod1**
27029 so that the owner and group have read/write/execute permissions, but the world only has
27030 read/write permissions.

27031 #include <sys/stat.h>

27032 #include <fcntl.h>

27033 mode_t mode;

27034 int fildes;

27035 ...

27036 fildes = open("/home/cnd/mod1", O_RDWR);

27037 fchmod(fildes, S_IRWXU | S_IRWXG | S_IROTH | S_IWOTH);

27038 APPLICATION USAGE

27039 None.

27040 RATIONALE

27041 None.

27042 FUTURE DIRECTIONS

27043 None.

27044 SEE ALSO

27045 *chmod()*, *chown()*, *creat()*, *fcntl()*, *fstatat()*, *fstatvfs()*, *mknod()*, *open()*, *read()*, *write()*

27046 XBD <sys/stat.h>

27047 CHANGE HISTORY

27048 First released in Issue 4, Version 2.

27049 Issue 5

27050 Moved from X/OPEN UNIX extension to BASE and aligned with *fchmod()* in the POSIX
27051 Realtime Extension. Specifically, the second paragraph of the DESCRIPTION is added and a
27052 second instance of [EINVAL] is defined in the list of optional errors.

27053 Issue 6

27054 The DESCRIPTION is updated for alignment with IEEE Std 1003.1j-2000 by stating that *fchmod()*
27055 behavior is unspecified for typed memory objects.

27056 **NAME**

27057 fchmodat — change mode of a file relative to directory file descriptor

27058 **SYNOPSIS**

27059 #include <sys/stat.h>

27060 int fchmodat(int *fd*, const char **path*, mode_t *mode*, int *flag*);27061 **DESCRIPTION**27062 Refer to *chmod()*.

27063 **NAME**27064 **fchown** — change owner and group of a file27065 **SYNOPSIS**27066 `#include <unistd.h>`27067 `int fchown(int fildes, uid_t owner, gid_t group);`27068 **DESCRIPTION**

27069 The *fchown()* function shall be equivalent to *chown()* except that the file whose owner and group
 27070 are changed is specified by the file descriptor *fil*des.

27071 **RETURN VALUE**

27072 Upon successful completion, *fchown()* shall return 0. Otherwise, it shall return -1 and set *errno* to
 27073 indicate the error.

27074 **ERRORS**27075 The *fchown()* function shall fail if:

27076 [EBADF] The *fil*des argument is not an open file descriptor.

27077 [EPERM] The effective user ID does not match the owner of the file or the process does
 27078 not have appropriate privileges and `_POSIX_CHOWN_RESTRICTED`
 27079 indicates that such privilege is required.

27080 [EROFS] The file referred to by *fil*des resides on a read-only file system.

27081 The *fchown()* function may fail if:

27082 [EINVAL] The owner or group ID is not a value supported by the implementation. The
 27083 OB XSR *fil*des argument refers to a pipe or socket or an *fattach()*-ed STREAM and the
 27084 implementation disallows execution of *fchown()* on a pipe.

27085 [EIO] A physical I/O error has occurred.

27086 [EINTR] The *fchown()* function was interrupted by a signal which was caught.

27087 **EXAMPLES**27088 **Changing the Current Owner of a File**

27089 The following example shows how to change the owner of a file named `/home/cnd/mod1` to
 27090 “jones” and the group to “cnd”.

27091 The numeric value for the user ID is obtained by extracting the user ID from the user database
 27092 entry associated with “jones”. Similarly, the numeric value for the group ID is obtained by
 27093 extracting the group ID from the group database entry associated with “cnd”. This example
 27094 assumes the calling program has appropriate privileges.

27095 `#include <sys/types.h>`27096 `#include <unistd.h>`27097 `#include <fcntl.h>`27098 `#include <pwd.h>`27099 `#include <grp.h>`27100 `struct passwd *pwd;`27101 `struct group *grp;`27102 `int fil`des;27103 `...`27104 `fil`des = open("/home/cnd/mod1", O_RDWR);27105 `pwd = getpwnam("jones");`

```
27106         grp = getgrnam("cnd");
27107         fchown(fildes, pwd->pw_uid, grp->gr_gid);
```

27108 APPLICATION USAGE

27109 None.

27110 RATIONALE

27111 None.

27112 FUTURE DIRECTIONS

27113 None.

27114 SEE ALSO

27115 *chown()*

27116 XBD <unistd.h>

27117 CHANGE HISTORY

27118 First released in Issue 4, Version 2.

27119 Issue 5

27120 Moved from X/OPEN UNIX extension to BASE.

27121 Issue 6

27122 The following changes were made to align with the IEEE P1003.1a draft standard:

- 27123 • Clarification is added that a call to *fchown()* may not be allowed on a pipe.

27124 The *fchown()* function is defined as mandatory.

27125 Issue 7

27126 Functionality relating to XSI STREAMS is marked obsolescent.

27127 NAME

27128 fchownat — change owner and group of a file relative to directory file descriptor

27129 SYNOPSIS

27130 #include <unistd.h>

27131 int fchownat(int *fd*, const char **path*, uid_t *owner*, gid_t *group*,
27132 int *flag*);

27133 DESCRIPTION

27134 Refer to *chown()*.

27135 **NAME**

27136 `fclose` — close a stream

27137 **SYNOPSIS**

27138 `#include <stdio.h>`

27139 `int fclose(FILE *stream);`

27140 **DESCRIPTION**

27141 CX The functionality described on this reference page is aligned with the ISO C standard. Any
27142 conflict between the requirements described here and the ISO C standard is unintentional. This
27143 volume of POSIX.1-2008 defers to the ISO C standard.

27144 The `fclose()` function shall cause the stream pointed to by *stream* to be flushed and the associated
27145 file to be closed. Any unwritten buffered data for the stream shall be written to the file; any
27146 unread buffered data shall be discarded. Whether or not the call succeeds, the stream shall be
27147 disassociated from the file and any buffer set by the `setbuf()` or `setvbuf()` function shall be
27148 disassociated from the stream. If the associated buffer was automatically allocated, it shall be
27149 deallocated.

27150 CX If the file is not already at EOF, and the file is one capable of seeking, the file offset of the
27151 underlying open file description shall be set to the file position of the stream if the stream is the
27152 active handle to the underlying file description.

27153 The `fclose()` function shall mark for update the last data modification and last file status change
27154 timestamps of the underlying file, if the stream was writable, and if buffered data remains that
27155 has not yet been written to the file. The `fclose()` function shall perform the equivalent of a `close()`
27156 on the file descriptor that is associated with the stream pointed to by *stream*.

27157 After the call to `fclose()`, any use of *stream* results in undefined behavior.

27158 **RETURN VALUE**

27159 CX Upon successful completion, `fclose()` shall return 0; otherwise, it shall return EOF and set *errno*
27160 to indicate the error.

27161 **ERRORS**

27162 The `fclose()` function shall fail if:

27163 CX [EAGAIN] The O_NONBLOCK flag is set for the file descriptor underlying *stream* and
27164 the thread would be delayed in the write operation.

27165 CX [EBADF] The file descriptor underlying stream is not valid.

27166 CX [EFBIG] An attempt was made to write a file that exceeds the maximum file size.

27167 XSI [EFBIG] An attempt was made to write a file that exceeds the file size limit of the
27168 process.

27169 CX [EFBIG] The file is a regular file and an attempt was made to write at or beyond the
27170 offset maximum associated with the corresponding stream.

27171 CX [EINTR] The `fclose()` function was interrupted by a signal.

27172 CX [EIO] The process is a member of a background process group attempting to write to
27173 its controlling terminal, TOSTOP is set, the calling thread is not blocking
27174 SIGTTOU, the process is not ignoring SIGTTOU, and the process group of the
27175 process is orphaned. This error may also be returned under implementation-
27176 defined conditions.

27177 CX [ENOMEM] The underlying stream was created by *open_memstream()* or
 27178 *open_wmemstream()* and insufficient memory is available.

27179 CX [ENOSPC] There was no free space remaining on the device containing the file or in the
 27180 buffer used by the *fmemopen()* function.

27181 CX [EPIPE] An attempt is made to write to a pipe or FIFO that is not open for reading by
 27182 any process. A SIGPIPE signal shall also be sent to the thread.

27183 The *fclose()* function may fail if:

27184 CX [ENXIO] A request was made of a nonexistent device, or the request was outside the
 27185 capabilities of the device.

27186 EXAMPLES

27187 None.

27188 APPLICATION USAGE

27189 None.

27190 RATIONALE

27191 None.

27192 FUTURE DIRECTIONS

27193 None.

27194 SEE ALSO

27195 [Section 2.5](#) (on page 495), *close()*, *fmemopen()*, *fopen()*, *getrlimit()*, *open_memstream()*, *ulimit()*

27196 XBD [<stdio.h>](#)

27197 CHANGE HISTORY

27198 First released in Issue 1. Derived from Issue 1 of the SVID.

27199 Issue 5

27200 Large File Summit extensions are added.

27201 Issue 6

27202 Extensions beyond the ISO C standard are marked.

27203 The following new requirements on POSIX implementations derive from alignment with the
 27204 Single UNIX Specification:

- 27205 • The [EFBIG] error is added as part of the large file support extensions.
- 27206 • The [ENXIO] optional error condition is added.

27207 The DESCRIPTION is updated to note that the stream and any buffer are disassociated whether
 27208 or not the call succeeds. This is for alignment with the ISO/IEC 9899:1999 standard.

27209 IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/28 is applied, updating the [EAGAIN]
 27210 error in the ERRORS section from “the process would be delayed” to “the thread would be
 27211 delayed”.

27212 Issue 7

27213 Austin Group Interpretation 1003.1-2001 #002 is applied, clarifying the interaction of file
 27214 descriptors and streams.

27215 The [ENOSPC] error condition is updated and the [ENOMEM] error is added from The Open
 27216 Group Technical Standard, 2006, Extended API Set Part 1.

27217 Changes are made related to support for finegrained timestamps.

27218
27219

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0113 [87], XSH/TC1-2008/0114 [79], +
and XSH/TC1-2008/0115 [14] are applied.

27220 **NAME**

27221 fcntl — file control

27222 **SYNOPSIS**

27223 #include <fcntl.h>

27224 int fcntl(int *fildes*, int *cmd*, ...);27225 **DESCRIPTION**

27226 The *fcntl()* function shall perform the operations described below on open files. The *fildes*
 27227 argument is a file descriptor.

27228 The available values for *cmd* are defined in <fcntl.h> and are as follows:

27229 **F_DUPFD** Return a new file descriptor which shall be the lowest numbered
 27230 available (that is, not already open) file descriptor greater than or equal to
 27231 the third argument, *arg*, taken as an integer of type **int**. The new file
 27232 descriptor shall refer to the same open file description as the original file
 27233 descriptor, and shall share any locks. The FD_CLOEXEC flag associated
 27234 with the new file descriptor shall be cleared to keep the file open across
 27235 calls to one of the *exec* functions.

27236 **F_DUPFD_CLOEXEC**

27237 Like F_DUPFD, but the FD_CLOEXEC flag associated with the new file
 27238 descriptor shall be set.

27239 **F_GETFD** Get the file descriptor flags defined in <fcntl.h> that are associated with
 27240 the file descriptor *fildes*. File descriptor flags are associated with a single
 27241 file descriptor and do not affect other file descriptors that refer to the
 27242 same file.

27243 **F_SETFD** Set the file descriptor flags defined in <fcntl.h>, that are associated with
 27244 *fildes*, to the third argument, *arg*, taken as type **int**. If the FD_CLOEXEC
 27245 flag in the third argument is 0, the file descriptor shall remain open across
 27246 the *exec* functions; otherwise, the file descriptor shall be closed upon
 27247 successful execution of one of the *exec* functions.

27248 **F_GETFL** Get the file status flags and file access modes, defined in <fcntl.h>, for the
 27249 file description associated with *fildes*. The file access modes can be
 27250 extracted from the return value using the mask O_ACCMODE, which is
 27251 defined in <fcntl.h>. File status flags and file access modes are associated
 27252 with the file description and do not affect other file descriptors that refer
 27253 to the same file with different open file descriptions. The flags returned
 27254 may include non-standard file status flags which the application did not
 27255 set, provided that these additional flags do not alter the behavior of a
 27256 conforming application.

27257 **F_SETFL** Set the file status flags, defined in <fcntl.h>, for the file description
 27258 associated with *fildes* from the corresponding bits in the third argument,
 27259 *arg*, taken as type **int**. Bits corresponding to the file access mode and the
 27260 file creation flags, as defined in <fcntl.h>, that are set in *arg* shall be
 27261 ignored. If any bits in *arg* other than those mentioned here are changed by
 27262 the application, the result is unspecified. If *fildes* does not support non-
 27263 blocking operations, it is unspecified whether the O_NONBLOCK flag
 27264 will be ignored.

27265	F_GETOWN	If <i>fildev</i> refers to a socket, get the process or process group ID specified to receive SIGURG signals when out-of-band data is available. Positive values indicate a process ID; negative values, other than -1, indicate a process group ID. If <i>fildev</i> does not refer to a socket, the results are unspecified.
27266		
27267		
27268		
27269		
27270	F_SETOWN	If <i>fildev</i> refers to a socket, set the process or process group ID specified to receive SIGURG signals when out-of-band data is available, using the value of the third argument, <i>arg</i> , taken as type int . Positive values indicate a process ID; negative values, other than -1, indicate a process group ID. If <i>fildev</i> does not refer to a socket, the results are unspecified.
27271		
27272		
27273		
27274		
27275	The following values for <i>cmd</i> are available for advisory record locking. Record locking shall be supported for regular files, and may be supported for other files.	
27276		
27277	F_GETLK	Get the first lock which blocks the lock description pointed to by the third argument, <i>arg</i> , taken as a pointer to type struct flock , defined in <fcntl.h> . The information retrieved shall overwrite the information passed to <i>fcntl()</i> in the structure flock . If no lock is found that would prevent this lock from being created, then the structure shall be left unchanged except for the lock type which shall be set to F_UNLCK.
27278		
27279		
27280		
27281		
27282		
27283	F_SETLK	Set or clear a file segment lock according to the lock description pointed to by the third argument, <i>arg</i> , taken as a pointer to type struct flock , defined in <fcntl.h> . F_SETLK can establish shared (or read) locks (F_RDLCK) or exclusive (or write) locks (F_WRLCK), as well as to remove either type of lock (F_UNLCK). F_RDLCK, F_WRLCK, and F_UNLCK are defined in <fcntl.h> . If a shared or exclusive lock cannot be set, <i>fcntl()</i> shall return immediately with a return value of -1.
27284		
27285		
27286		
27287		
27288		
27289		
27290	F_SETLKW	This command shall be equivalent to F_SETLK except that if a shared or exclusive lock is blocked by other locks, the thread shall wait until the request can be satisfied. If a signal that is to be caught is received while <i>fcntl()</i> is waiting for a region, <i>fcntl()</i> shall be interrupted. Upon return from the signal handler, <i>fcntl()</i> shall return -1 with <i>errno</i> set to [EINTR], and the lock operation shall not be done.
27291		
27292		
27293		
27294		
27295		
27296	Additional implementation-defined values for <i>cmd</i> may be defined in <fcntl.h> . Their names shall start with F_.	
27297		
27298	When a shared lock is set on a segment of a file, other processes shall be able to set shared locks on that segment or a portion of it. A shared lock prevents any other process from setting an exclusive lock on any portion of the protected area. A request for a shared lock shall fail if the file descriptor was not opened with read access.	
27299		
27300		
27301		
27302	An exclusive lock shall prevent any other process from setting a shared lock or an exclusive lock on any portion of the protected area. A request for an exclusive lock shall fail if the file descriptor was not opened with write access.	
27303		
27304		
27305	The structure flock describes the type (<i>l_type</i>), starting offset (<i>l_whence</i>), relative offset (<i>l_start</i>), size (<i>l_len</i>), and process ID (<i>l_pid</i>) of the segment of the file to be affected.	
27306		
27307	The value of <i>l_whence</i> is SEEK_SET, SEEK_CUR, or SEEK_END, to indicate that the relative offset <i>l_start</i> bytes shall be measured from the start of the file, current position, or end of the file, respectively. The value of <i>l_len</i> is the number of consecutive bytes to be locked. The value of <i>l_len</i> may be negative (where the definition of off_t permits negative values of <i>l_len</i>). The <i>l_pid</i> field is only used with F_GETLK to return the process ID of the process holding a blocking lock. After	
27308		
27309		
27310		
27311		

27312 a successful F_GETLK request, when a blocking lock is found, the values returned in the **lock**
 27313 structure shall be as follows:

27314 *l_type* Type of blocking lock found.
 27315 *l_whence* SEEK_SET.
 27316 *l_start* Start of the blocking lock.
 27317 *l_len* Length of the blocking lock.
 27318 *l_pid* Process ID of the process that holds the blocking lock.

27319 If the command is F_SETLK and the process must wait for another process to release a lock,
 27320 then the range of bytes to be locked shall be determined before the *fcntl()* function blocks. If the
 27321 file size or file descriptor seek offset change while *fcntl()* is blocked, this shall not affect the
 27322 range of bytes locked.

27323 If *l_len* is positive, the area affected shall start at *l_start* and end at *l_start+l_len-1*. If *l_len* is
 27324 negative, the area affected shall start at *l_start+l_len* and end at *l_start-1*. Locks may start and
 27325 extend beyond the current end of a file, but shall not extend before the beginning of the file. A
 27326 lock shall be set to extend to the largest possible value of the file offset for that file by setting
 27327 *l_len* to 0. If such a lock also has *l_start* set to 0 and *l_whence* is set to SEEK_SET, the whole file
 27328 shall be locked.

27329 There shall be at most one type of lock set for each byte in the file. Before a successful return
 27330 from an F_SETLK or an F_SETLKW request when the calling process has previously existing
 27331 locks on bytes in the region specified by the request, the previous lock type for each byte in the
 27332 specified region shall be replaced by the new lock type. As specified above under the
 27333 descriptions of shared locks and exclusive locks, an F_SETLK or an F_SETLKW request
 27334 (respectively) shall fail or block when another process has existing locks on bytes in the specified
 27335 region and the type of any of those locks conflicts with the type specified in the request.

27336 All locks associated with a file for a given process shall be removed when a file descriptor for
 27337 that file is closed by that process or the process holding that file descriptor terminates. Locks are
 27338 not inherited by a child process.

27339 A potential for deadlock occurs if a process controlling a locked region is put to sleep by
 27340 attempting to lock the locked region of another process. If the system detects that sleeping until
 27341 a locked region is unlocked would cause a deadlock, *fcntl()* shall fail with an [EDEADLK] error.

27342 An unlock (F_UNLCK) request in which *l_len* is non-zero and the offset of the last byte of the
 27343 requested segment is the maximum value for an object of type **off_t**, when the process has an
 27344 existing lock in which *l_len* is 0 and which includes the last byte of the requested segment, shall
 27345 be treated as a request to unlock from the start of the requested segment with an *l_len* equal to 0.
 27346 Otherwise, an unlock (F_UNLCK) request shall attempt to unlock only the requested segment.

27347 SHM When the file descriptor *fildev* refers to a shared memory object, the behavior of *fcntl()* shall be
 27348 the same as for a regular file except the effect of the following values for the argument *cmd* shall
 27349 be unspecified: F_SETFL, F_GETLK, F_SETLK, and F_SETLKW.

27350 TYM If *fildev* refers to a typed memory object, the result of the *fcntl()* function is unspecified.

27351 RETURN VALUE

27352 Upon successful completion, the value returned shall depend on *cmd* as follows:

27353 F_DUPFD A new file descriptor.

27354	F_DUPFD_CLOEXEC	
27355		A new file descriptor.
27356	F_GETFD	Value of flags defined in <code><fcntl.h></code> . The return value shall not be negative.
27357	F_SETFD	Value other than <code>-1</code> .
27358	F_GETFL	Value of file status flags and access modes. The return value is not negative.
27359	F_SETFL	Value other than <code>-1</code> .
27360	F_GETLK	Value other than <code>-1</code> .
27361	F_SETLK	Value other than <code>-1</code> .
27362	F_SETLKW	Value other than <code>-1</code> .
27363	F_GETOWN	Value of the socket owner process or process group; this will not be <code>-1</code> .
27364	F_SETOWN	Value other than <code>-1</code> .
27365		Otherwise, <code>-1</code> shall be returned and <i>errno</i> set to indicate the error.

ERRORS

27366	The <i>fcntl()</i> function shall fail if:	
27367		
27368	[EACCES] or [EAGAIN]	
27369		The <i>cmd</i> argument is <code>F_SETLK</code> ; the type of lock (<i>l_type</i>) is a shared (<code>F_RDLCK</code>) or exclusive (<code>F_WRLCK</code>) lock and the segment of a file to be locked is already exclusive-locked by another process, or the type is an exclusive lock and some portion of the segment of a file to be locked is already shared-locked or exclusive-locked by another process.
27370		
27371		
27372		
27373		
27374	[EBADF]	The <i>fildest</i> argument is not a valid open file descriptor, or the argument <i>cmd</i> is <code>F_SETLK</code> or <code>F_SETLKW</code> , the type of lock, <i>l_type</i> , is a shared lock (<code>F_RDLCK</code>), and <i>fildest</i> is not a valid file descriptor open for reading, or the type of lock, <i>l_type</i> , is an exclusive lock (<code>F_WRLCK</code>), and <i>fildest</i> is not a valid file descriptor open for writing.
27375		
27376		
27377		
27378		
27379	[EINTR]	The <i>cmd</i> argument is <code>F_SETLKW</code> and the function was interrupted by a signal.
27380	[EINVAL]	The <i>cmd</i> argument is invalid, or the <i>cmd</i> argument is <code>F_DUPFD</code> or <code>F_DUPFD_CLOEXEC</code> and <i>arg</i> is negative or greater than or equal to <code>{OPEN_MAX}</code> , or the <i>cmd</i> argument is <code>F_GETLK</code> , <code>F_SETLK</code> , or <code>F_SETLKW</code> and the data pointed to by <i>arg</i> is not valid, or <i>fildest</i> refers to a file that does not support locking.
27381		
27382		
27383		
27384		
27385	[EMFILE]	The argument <i>cmd</i> is <code>F_DUPFD</code> or <code>F_DUPFD_CLOEXEC</code> and all file descriptors available to the process are currently open, or no file descriptors greater than or equal to <i>arg</i> are available.
27386		
27387		
27388	[ENOLCK]	The argument <i>cmd</i> is <code>F_SETLK</code> or <code>F_SETLKW</code> and satisfying the lock or unlock request would result in the number of locked regions in the system exceeding a system-imposed limit.
27389		
27390		
27391	[EOVERFLOW]	One of the values to be returned cannot be represented correctly.
27392	[EOVERFLOW]	The <i>cmd</i> argument is <code>F_GETLK</code> , <code>F_SETLK</code> , or <code>F_SETLKW</code> and the smallest or, if <i>l_len</i> is non-zero, the largest offset of any byte in the requested segment cannot be represented correctly in an object of type <code>off_t</code> .
27393		
27394		

27395 The *fcntl()* function may fail if:

27396 [EDEADLK] The *cmd* argument is F_SETLKW, the lock is blocked by a lock from another
 27397 process, and putting the calling process to sleep to wait for that lock to become
 27398 free would cause a deadlock.

27399 EXAMPLES

27400 Locking and Unlocking a File

27401 The following example demonstrates how to place a lock on bytes 100 to 109 of a file and then
 27402 later remove it. F_SETLKW is used to perform a non-blocking lock request so that the process does
 27403 not have to wait if an incompatible lock is held by another process; instead the process can take
 27404 some other action.

```

27405 #include <stdlib.h>
27406 #include <unistd.h>
27407 #include <fcntl.h>
27408 #include <errno.h>
27409 #include <stdio.h>

27410 int
27411 main(int argc, char *argv[])
27412 {
27413     int fd;
27414     struct flock fl;

27415     fd = open("testfile", O_RDWR);
27416     if (fd == -1)
27417         /* Handle error */;

27418     /* Make a non-blocking request to place a write lock
27419        on bytes 100-109 of testfile */

27420     fl.l_type = F_WRLCK;
27421     fl.l_whence = SEEK_SET;
27422     fl.l_start = 100;
27423     fl.l_len = 10;

27424     if (fcntl(fd, F_SETLKW, &fl) == -1) {
27425         if (errno == EACCES || errno == EAGAIN) {
27426             printf("Already locked by another process\n");
27427             /* We can't get the lock at the moment */

27428         } else {
27429             /* Handle unexpected error */;
27430         }
27431     } else { /* Lock was granted... */

27432         /* Perform I/O on bytes 100 to 109 of file */

27433         /* Unlock the locked bytes */

27434         fl.l_type = F_UNLCK;
27435         fl.l_whence = SEEK_SET;
27436         fl.l_start = 100;
27437         fl.l_len = 10;
27438         if (fcntl(fd, F_SETLKW, &fl) == -1)
  
```

```

27439             /* Handle error */;
27440         }
27441         exit(EXIT_SUCCESS);
27442     } /* main */

```

Setting the Close-on-Exec Flag

The following example demonstrates how to set the close-on-exec flag for the file descriptor *fd*.

```

27445 #include <unistd.h>
27446 #include <fcntl.h>
27447 ...
27448     int flags;
27449
27449     flags = fcntl(fd, F_GETFD);
27450     if (flags == -1)
27451         /* Handle error */;
27452     flags |= FD_CLOEXEC;
27453     if (fcntl(fd, F_SETFD, flags) == -1)
27454         /* Handle error */;

```

APPLICATION USAGE

The *arg* values to `F_GETFD`, `F_SETFD`, `F_GETFL`, and `F_SETFL` all represent flag values to allow for future growth. Applications using these functions should do a read-modify-write operation on them, rather than assuming that only the values defined by this volume of POSIX.1-2008 are valid. It is a common error to forget this, particularly in the case of `F_SETFD`. Some implementations set additional file status flags to advise the application of default behavior, even though the application did not request these flags.

RATIONALE

The ellipsis in the SYNOPSIS is the syntax specified by the ISO C standard for a variable number of arguments. It is used because System V uses pointers for the implementation of file locking functions.

This volume of POSIX.1-2008 permits concurrent read and write access to file data using the `fcntl()` function; this is a change from the 1984 `/usr/group` standard and early proposals. Without concurrency controls, this feature may not be fully utilized without occasional loss of data.

Data losses occur in several ways. One case occurs when several processes try to update the same record, without sequencing controls; several updates may occur in parallel and the last writer “wins”. Another case is a bit-tree or other internal list-based database that is undergoing reorganization. Without exclusive use to the tree segment by the updating process, other reading processes chance getting lost in the database when the index blocks are split, condensed, inserted, or deleted. While `fcntl()` is useful for many applications, it is not intended to be overly general and does not handle the bit-tree example well.

This facility is only required for regular files because it is not appropriate for many devices such as terminals and network connections.

Since `fcntl()` works with “any file descriptor associated with that file, however it is obtained”, the file descriptor may have been inherited through a `fork()` or `exec` operation and thus may affect a file that another process also has open.

The use of the open file description to identify what to lock requires extra calls and presents problems if several processes are sharing an open file description, but there are too many

implementations of the existing mechanism for this volume of POSIX.1-2008 to use different specifications.

Another consequence of this model is that closing any file descriptor for a given file (whether or not it is the same open file description that created the lock) causes the locks on that file to be relinquished for that process. Equivalently, any close for any file/process pair relinquishes the locks owned on that file for that process. But note that while an open file description may be shared through *fork()*, locks are not inherited through *fork()*. Yet locks may be inherited through one of the *exec* functions.

The identification of a machine in a network environment is outside the scope of this volume of POSIX.1-2008. Thus, an *l_sysid* member, such as found in System V, is not included in the locking structure.

Changing of lock types can result in a previously locked region being split into smaller regions.

Mandatory locking was a major feature of the 1984 /usr/group standard.

For advisory file record locking to be effective, all processes that have access to a file must cooperate and use the advisory mechanism before doing I/O on the file. Enforcement-mode record locking is important when it cannot be assumed that all processes are cooperating. For example, if one user uses an editor to update a file at the same time that a second user executes another process that updates the same file and if only one of the two processes is using advisory locking, the processes are not cooperating. Enforcement-mode record locking would protect against accidental collisions.

Secondly, advisory record locking requires a process using locking to bracket each I/O operation with lock (or test) and unlock operations. With enforcement-mode file and record locking, a process can lock the file once and unlock when all I/O operations have been completed. Enforcement-mode record locking provides a base that can be enhanced; for example, with sharable locks. That is, the mechanism could be enhanced to allow a process to lock a file so other processes could read it, but none of them could write it.

Mandatory locks were omitted for several reasons:

1. Mandatory lock setting was done by multiplexing the set-group-ID bit in most implementations; this was confusing, at best.
2. The relationship to file truncation as supported in 4.2 BSD was not well specified.
3. Any publicly readable file could be locked by anyone. Many historical implementations keep the password database in a publicly readable file. A malicious user could thus prohibit logins. Another possibility would be to hold open a long-distance telephone line.
4. Some demand-paged historical implementations offer memory mapped files, and enforcement cannot be done on that type of file.

Since sleeping on a region is interrupted with any signal, *alarm()* may be used to provide a timeout facility in applications requiring it. This is useful in deadlock detection. Since implementation of full deadlock detection is not always feasible, the [EDEADLK] error was made optional.

FUTURE DIRECTIONS

None.

SEE ALSO

alarm(), *close()*, *exec*, *open()*, *sigaction()*

XBD **<fcntl.h>**, **<signal.h>**

CHANGE HISTORY

First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 5

The DESCRIPTION is updated for alignment with the POSIX Realtime Extension and the POSIX Threads Extension.

Large File Summit extensions are added.

Issue 6

In the SYNOPSIS, the optional include of the **<sys/types.h>** header is removed.

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The requirement to include **<sys/types.h>** has been removed. Although **<sys/types.h>** was required for conforming implementations of previous POSIX specifications, it was not required for UNIX applications.
- In the DESCRIPTION, sentences describing behavior when *l_len* is negative are now mandated, and the description of unlock (F_UNLOCK) when *l_len* is non-negative is mandated.
- In the ERRORS section, the [EINVAL] error condition has the case mandated when the *cmd* is invalid, and two [EOVERFLOW] error conditions are added.

The F_GETOWN and F_SETOWN values are added for sockets.

The following changes were made to align with the IEEE P1003.1a draft standard:

- Clarification is added that the extent of the bytes locked is determined prior to the blocking action.

The DESCRIPTION is updated for alignment with IEEE Std 1003.1j-2000 by specifying that *fcntl()* results are unspecified for typed memory objects.

The normative text is updated to avoid use of the term “must” for application requirements.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/29 is applied, adding the example to the EXAMPLES section.

Issue 7

Austin Group Interpretation 1003.1-2001 #150 is applied, clarifying the file status flags returned when *cmd* is F_GETFL.

Austin Group Interpretation 1003.1-2001 #171 is applied, adding support to set the FD_CLOEXEC flag atomically at *open()*, and adding the F_DUPFD_CLOEXEC flag.

The optional **<unistd.h>** header is removed from this function, since **<fcntl.h>** now defines SEEK_SET, SEEK_CUR, and SEEK_END as part of the Base.

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0116 [141] is applied.

+

27563 **NAME**27564 `fdatasync` — synchronize the data of a file (**REALTIME**)27565 **SYNOPSIS**

```
27566 SIO      #include <unistd.h>
27567          int fdatasync(int fildes);
```

27568 **DESCRIPTION**

27569 The `fdatasync()` function shall force all currently queued I/O operations associated with the file
 27570 indicated by file descriptor *fildes* to the synchronized I/O completion state.

27571 The functionality shall be equivalent to `fsync()` with the symbol `_POSIX_SYNCHRONIZED_IO`
 27572 defined, with the exception that all I/O operations shall be completed as defined for
 27573 synchronized I/O data integrity completion.

27574 **RETURN VALUE**

27575 If successful, the `fdatasync()` function shall return the value 0; otherwise, the function shall return
 27576 the value `-1` and set *errno* to indicate the error. If the `fdatasync()` function fails, outstanding I/O
 27577 operations are not guaranteed to have been completed.

27578 **ERRORS**

27579 The `fdatasync()` function shall fail if:

27580 [EBADF] The *fildes* argument is not a valid file descriptor open for writing.

27581 [EINVAL] This implementation does not support synchronized I/O for this file.

27582 In the event that any of the queued I/O operations fail, `fdatasync()` shall return the error
 27583 conditions defined for `read()` and `write()`.

27584 **EXAMPLES**

27585 None.

27586 **APPLICATION USAGE**

27587 None.

27588 **RATIONALE**

27589 None.

27590 **FUTURE DIRECTIONS**

27591 None.

27592 **SEE ALSO**

27593 [*aio_fsync\(\)*](#), [*fcntl\(\)*](#), [*fsync\(\)*](#), [*open\(\)*](#), [*read\(\)*](#), [*write\(\)*](#)

27594 XBD [*<unistd.h>*](#)

27595 **CHANGE HISTORY**

27596 First released in Issue 5. Included for alignment with the POSIX Realtime Extension.

27597 **Issue 6**

27598 The [ENOSYS] error condition has been removed as stubs need not be provided if an
 27599 implementation does not support the Synchronized Input and Output option.

27600 The `fdatasync()` function is marked as part of the Synchronized Input and Output option.

27601 **NAME**27602 fdetach — detach a name from a STREAMS-based file descriptor (**STREAMS**)27603 **SYNOPSIS**

```
27604 OB XSR #include <stropts.h>
27605 int fdetach(const char *path);
```

27606 **DESCRIPTION**

27607 The *fdetach()* function shall detach a STREAMS-based file from the file to which it was attached
 27608 by a previous call to *fattach()*. The *path* argument points to the pathname of the attached
 27609 STREAMS file. The process shall have appropriate privileges or be the owner of the file. A
 27610 successful call to *fdetach()* shall cause all pathnames that named the attached STREAMS file to
 27611 again name the file to which the STREAMS file was attached. All subsequent operations on *path*
 27612 shall operate on the underlying file and not on the STREAMS file.

27613 All open file descriptions established while the STREAMS file was attached to the file referenced
 27614 by *path* shall still refer to the STREAMS file after the *fdetach()* has taken effect.

27615 If there are no open file descriptors or other references to the STREAMS file, then a successful
 27616 call to *fdetach()* shall be equivalent to performing the last *close()* on the attached file.

27617 **RETURN VALUE**

27618 Upon successful completion, *fdetach()* shall return 0; otherwise, it shall return -1 and set *errno* to
 27619 indicate the error.

27620 **ERRORS**

27621 The *fdetach()* function shall fail if:

27622	[EACCES]	Search permission is denied on a component of the path prefix.
27623	[EINVAL]	The <i>path</i> argument names a file that is not currently attached.
27624	[ELOOP]	A loop exists in symbolic links encountered during resolution of the <i>path</i> argument.
27625		
27626	[ENAMETOOLONG]	
27627		The length of a component of a pathname is longer than {NAME_MAX}.
27628	[ENOENT]	A component of <i>path</i> does not name an existing file or <i>path</i> is an empty string.
27629	[ENOTDIR]	A component of the path prefix names an existing file that is neither a directory nor a symbolic link to a directory, or the <i>path</i> argument contains at least one non- <i><slash></i> character and ends with one or more trailing <i><slash></i> characters and the last pathname component names an existing file that is neither a directory nor a symbolic link to a directory.
27630		
27631		
27632		
27633		
27634	[EPERM]	The effective user ID is not the owner of <i>path</i> and the process does not have appropriate privileges.
27635		

27636 The *fdetach()* function may fail if:

27637	[ELOOP]	More than {SYMLOOP_MAX} symbolic links were encountered during resolution of the <i>path</i> argument.
27638		
27639	[ENAMETOOLONG]	
27640		The length of a pathname exceeds {PATH_MAX}, or pathname resolution of a symbolic link produced an intermediate result with a length that exceeds {PATH_MAX}.
27641		
27642		

27643 **EXAMPLES**27644 **Detaching a File**

27645 The following example detaches the STREAMS-based file **/tmp/named-STREAM** from the file to
 27646 which it was attached by a previous, successful call to *fattach()*. Subsequent calls to open this
 27647 file refer to the underlying file, not to the STREAMS file.

```
27648 #include <stropts.h>
27649 ...
27650 char *pathname = "/tmp/named-STREAM";
27651 int ret;
27652 ret = fdetach(pathname);
```

27653 **APPLICATION USAGE**

27654 None.

27655 **RATIONALE**

27656 None.

27657 **FUTURE DIRECTIONS**

27658 The *fdetach()* function may be removed in a future version.

27659 **SEE ALSO**

27660 *fattach()*

27661 XBD **<stropts.h>**

27662 **CHANGE HISTORY**

27663 First released in Issue 4, Version 2.

27664 **Issue 5**

27665 Moved from X/OPEN UNIX extension to BASE.

27666 **Issue 6**

27667 The normative text is updated to avoid use of the term “must” for application requirements.

27668 The wording of the mandatory [ELOOP] error condition is updated, and a second optional
 27669 [ELOOP] error condition is added.

27670 **Issue 7**

27671 Austin Group Interpretation 1003.1-2001 #143 is applied.

27672 The *fdetach()* function is marked obsolescent.

27673 The [ENOTDIR] error condition is clarified to cover the condition where the last component of a
 27674 pathname exists but is not a directory or a symbolic link to a directory.

27675 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0117 [324] and XSH/TC1-2008/0118 +
 27676 [291] are applied.

27677 **NAME**

27678 fdim, fdimf, fdiml — compute positive difference between two floating-point numbers

27679 **SYNOPSIS**

```
27680 #include <math.h>
27681 double fdim(double x, double y);
27682 float fdimf(float x, float y);
27683 long double fdiml(long double x, long double y);
```

27684 **DESCRIPTION**

27685 CX The functionality described on this reference page is aligned with the ISO C standard. Any
 27686 conflict between the requirements described here and the ISO C standard is unintentional. This
 27687 volume of POSIX.1-2008 defers to the ISO C standard.

27688 These functions shall determine the positive difference between their arguments. If x is greater
 27689 than y , $x-y$ is returned. If x is less than or equal to y , +0 is returned.

27690 An application wishing to check for error situations should set *errno* to zero and call
 27691 *feclearexcept*(FE_ALL_EXCEPT) before calling these functions. On return, if *errno* is non-zero or
 27692 *fetestexcept*(FE_INVALID | FE_DIVBYZERO | FE_OVERFLOW | FE_UNDERFLOW) is non-
 27693 zero, an error has occurred.

27694 **RETURN VALUE**

27695 Upon successful completion, these functions shall return the positive difference value.

27696 If $x-y$ is positive and overflows, a range error shall occur and *fdim*(), *fdimf*(), and *fdiml*() shall
 27697 return the value of the macro HUGE_VAL, HUGE_VALF, and HUGE_VALL, respectively.

27698 If the correct value would cause underflow, a range error may occur, and *fdim*(), *fdimf*(), and
 27699 MXX *fdiml*() shall return the correct value, or (if the IEC 60559 Floating-Point option is not
 27700 supported) an implementation-defined value no greater in magnitude than DBL_MIN,
 27701 FLT_MIN, and LDBL_MIN, respectively.

27702 MX If x or y is NaN, a NaN shall be returned.

27703 **ERRORS**

27704 The *fdim*() function shall fail if:

27705 Range Error The result overflows.

27706 If the integer expression (*math_errhandling* & MATH_ERRNO) is non-zero,
 27707 then *errno* shall be set to [ERANGE]. If the integer expression
 27708 (*math_errhandling* & MATH_ERREXCEPT) is non-zero, then the overflow
 27709 floating-point exception shall be raised.

27710 The *fdim*() function may fail if:

27711 Range Error The result underflows.

27712 If the integer expression (*math_errhandling* & MATH_ERRNO) is non-zero,
 27713 then *errno* shall be set to [ERANGE]. If the integer expression
 27714 (*math_errhandling* & MATH_ERREXCEPT) is non-zero, then the underflow
 27715 floating-point exception shall be raised.

27716 **EXAMPLES**

27717 None.

27718 **APPLICATION USAGE**

27719 On error, the expressions (*math_errhandling* & MATH_ERRNO) and (*math_errhandling* & -
27720 MATH_ERREXCEPT) are independent of each other, but at least one of them must be non-zero.

27721 **RATIONALE**

27722 None.

27723 **FUTURE DIRECTIONS**

27724 None.

27725 **SEE ALSO**27726 *feclearexcept()*, *fetestexcept()*, *fmax()*, *fmin()*

27727 Section 4.19 (on page 116), <math.h>

27728 **CHANGE HISTORY**

27729 First released in Issue 6. Derived from the ISO/IEC 9899: 1999 standard.

27730 **Issue 7**

27731 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0119 [68,428] and +
27732 XSH/TC1-2008/0120 [68,428] are applied.

27733 **NAME**

27734 fdopen — associate a stream with a file descriptor

27735 **SYNOPSIS**

```
27736 CX      #include <stdio.h>
27737 FILE *fdopen(int fil-des, const char *mode);
```

27738 **DESCRIPTION**27739 The *fdopen()* function shall associate a stream with a file descriptor.27740 The *mode* argument is a character string having one of the following values:27741 *r* or *rb* Open a file for reading.27742 *w* or *wb* Open a file for writing.27743 *a* or *ab* Open a file for writing at end-of-file.27744 *r+* or *rb+* or *r+b* Open a file for update (reading and writing).27745 *w+* or *wb+* or *w+b* Open a file for update (reading and writing).27746 *a+* or *ab+* or *a+b* Open a file for update (reading and writing) at end-of-file.27747 The meaning of these flags is exactly as specified in *fopen()*, except that modes beginning with *w*
27748 shall not cause truncation of the file.27749 Additional values for the *mode* argument may be supported by an implementation.

27750 The application shall ensure that the mode of the stream as expressed by the *mode* argument is
27751 allowed by the file access mode of the open file description to which *fil-des* refers. The file
27752 position indicator associated with the new stream is set to the position indicated by the file offset
27753 associated with the file descriptor.

27754 The error and end-of-file indicators for the stream shall be cleared. The *fdopen()* function may
27755 cause the last data access timestamp of the underlying file to be marked for update.

27756 SHM If *fil-des* refers to a shared memory object, the result of the *fdopen()* function is unspecified.27757 TYM If *fil-des* refers to a typed memory object, the result of the *fdopen()* function is unspecified.

27758 The *fdopen()* function shall preserve the offset maximum previously set for the open file
27759 description corresponding to *fil-des*.

27760 **RETURN VALUE**

27761 Upon successful completion, *fdopen()* shall return a pointer to a stream; otherwise, a null pointer
27762 shall be returned and *errno* set to indicate the error.

27763 **ERRORS**27764 The *fdopen()* function shall fail if:

27765 [EMFILE] {STREAM_MAX} streams are currently open in the calling process.

27766 The *fdopen()* function may fail if:27767 [EBADF] The *fil-des* argument is not a valid file descriptor.27768 [EINVAL] The *mode* argument is not a valid mode.

27769 [EMFILE] {FOPEN_MAX} streams are currently open in the calling process.

27770 [ENOMEM] Insufficient space to allocate a buffer.

27771 EXAMPLES

27772 None.

27773 APPLICATION USAGE

27774 File descriptors are obtained from calls like *open()*, *dup()*, *creat()*, or *pipe()*, which open files but
27775 do not return streams.

27776 RATIONALE

27777 The file descriptor may have been obtained from *open()*, *creat()*, *pipe()*, *dup()*, *fcntl()*, or *socket()*;
27778 inherited through *fork()*, *posix_spawn()*, or *exec*; or perhaps obtained by other means.

27779 The meanings of the *mode* arguments of *fdopen()* and *fopen()* differ. With *fdopen()*, open for write
27780 (*w* or *w+*) does not truncate, and append (*a* or *a+*) cannot create for writing. The *mode* argument
27781 formats that include a *b* are allowed for consistency with the ISO C standard function *fopen()*.
27782 The *b* has no effect on the resulting stream. Although not explicitly required by this volume of
27783 POSIX.1-2008, a good implementation of append (*a*) mode would cause the O_APPEND flag to
27784 be set.

27785 FUTURE DIRECTIONS

27786 None.

27787 SEE ALSO

27788 [Section 2.5.1](#) (on page 496), [fclose\(\)](#), [fnemopen\(\)](#), [fopen\(\)](#), [open\(\)](#), [open_memstream\(\)](#),
27789 [posix_spawn\(\)](#), [socket\(\)](#)

27790 XBD [<stdio.h>](#)

27791 CHANGE HISTORY

27792 First released in Issue 1. Derived from Issue 1 of the SVID.

27793 Issue 5

27794 The DESCRIPTION is updated for alignment with the POSIX Realtime Extension.

27795 Large File Summit extensions are added.

27796 Issue 6

27797 The following new requirements on POSIX implementations derive from alignment with the
27798 Single UNIX Specification:

- 27799 • In the DESCRIPTION, the use and setting of the *mode* argument are changed to include
27800 binary streams.
- 27801 • In the DESCRIPTION, text is added for large file support to indicate setting of the offset
27802 maximum in the open file description.
- 27803 • All errors identified in the ERRORS section are added.
- 27804 • In the DESCRIPTION, text is added that the *fdopen()* function may cause *st_atime* to be
27805 updated.

27806 The following changes were made to align with the IEEE P1003.1a draft standard:

- 27807 • Clarification is added that it is the responsibility of the application to ensure that the mode
27808 is compatible with the open file descriptor.

27809 The DESCRIPTION is updated for alignment with IEEE Std 1003.1j-2000 by specifying that
27810 *fdopen()* results are unspecified for typed memory objects.

27811	IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/30 is applied, making corrections to the	
27812	RATIONALE.	
27813	Issue 7	
27814	SD5-XSH-ERN-149 is applied, adding the {STREAM_MAX} [EMFILE] error condition.	
27815	Changes are made related to support for finegrained timestamps.	
27816	POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0121 [409] is applied.	+

27817 **NAME**

27818 fdopendir, opendir — open directory associated with file descriptor

27819 **SYNOPSIS**

27820 #include <dirent.h>

27821 DIR *fdopendir(int *fd*);27822 DIR *opendir(const char **dirname*);27823 **DESCRIPTION**

27824 The *fdopendir()* function shall be equivalent to the *opendir()* function except that the directory is
 27825 specified by a file descriptor rather than by a name. The file offset associated with the file
 27826 descriptor at the time of the call determines which entries are returned.

27827 Upon successful return from *fdopendir()*, the file descriptor is under the control of the system,
 27828 and if any attempt is made to close the file descriptor, or to modify the state of the associated |
 27829 XSI description, other than by means of *closedir()*, *readdir()*, *readdir_r()*, *rewinddir()*, or *seekdir()*, the +
 27830 behavior is undefined. Upon calling *closedir()* the file descriptor shall be closed.

27831 It is unspecified whether the FD_CLOEXEC flag will be set on the file descriptor by a successful
 27832 call to *fdopendir()*.

27833 The *opendir()* function shall open a directory stream corresponding to the directory named by
 27834 the *dirname* argument. The directory stream is positioned at the first entry. If the type **DIR** is
 27835 implemented using a file descriptor, applications shall only be able to open up to a total of
 27836 {OPEN_MAX} files and directories.

27837 If the type **DIR** is implemented using a file descriptor, the descriptor shall be obtained as if the
 27838 O_DIRECTORY flag was passed to *open()*.

27839 **RETURN VALUE**

27840 Upon successful completion, these functions shall return a pointer to an object of type **DIR**.
 27841 Otherwise, these functions shall return a null pointer and set *errno* to indicate the error.

27842 **ERRORS**27843 The *fdopendir()* function shall fail if:27844 [EBADF] The *fd* argument is not a valid file descriptor open for reading.27845 [ENOTDIR] The descriptor *fd* is not associated with a directory.27846 The *opendir()* function shall fail if:

27847 [EACCES] Search permission is denied for the component of the path prefix of *dirname* or
 27848 read permission is denied for *dirname*.

27849 [ELOOP] A loop exists in symbolic links encountered during resolution of the *dirname*
 27850 argument.

27851 [ENAMETOOLONG]
 27852 The length of a component of a pathname is longer than {NAME_MAX}.

27853 [ENOENT] A component of *dirname* does not name an existing directory or *dirname* is an
 27854 empty string.

27855 [ENOTDIR] A component of *dirname* names an existing file that is neither a directory nor a |
 27856 symbolic link to a directory.

27857 The *opendir()* function may fail if:

27858 [ELOOP] More than {SYMLOOP_MAX} symbolic links were encountered during
27859 resolution of the *dirname* argument.

27860 [EMFILE] All file descriptors available to the process are currently open.

27861 [ENAMETOOLONG]
27862 The length of a pathname exceeds {PATH_MAX}, or pathname resolution of a
27863 symbolic link produced an intermediate result with a length that exceeds
27864 {PATH_MAX}.

27865 [ENFILE] Too many files are currently open in the system.

27866 EXAMPLES

27867 Open a Directory Stream

27868 The following program fragment demonstrates how the *opendir()* function is used.

```
27869 #include <dirent.h>
27870 ...
27871     DIR *dir;
27872     struct dirent *dp;
27873 ...
27874     if ((dir = opendir(".")) == NULL) {
27875         perror("Cannot open .");
27876         exit(1);
27877     }
27878     while ((dp = readdir(dir)) != NULL) {
27879 ...
```

27880 Find And Open a File

27881 The following program searches through a given directory looking for files whose name does
27882 not begin with a dot and whose size is larger than 1 MiB.

```
27883 #include <stdio.h>
27884 #include <dirent.h>
27885 #include <fcntl.h>
27886 #include <sys/stat.h>
27887 #include <stdint.h>
27888 #include <stdlib.h>
27889 #include <unistd.h>
27890
27891 int
27892 main(int argc, char *argv[])
27893 {
27894     struct stat statbuf;
27895     DIR *d;
27896     struct dirent *dp;
27897     int dfd, ffd;
27898
27899     if ((d = fdopendir((dfd = open("./tmp", O_RDONLY)))) == NULL) {
27900         fprintf(stderr, "Cannot open ./tmp directory\n");
27901         exit(1);
27902     }
```

```

27900     }
27901     while ((dp = readdir(d)) != NULL) {
27902         if (dp->d_name[0] == '.')
27903             continue;
27904         /* there is a possible race condition here as the file
27905          * could be renamed between the readdir and the open */
27906         if ((ffd = openat(dfd, dp->d_name, O_RDONLY)) == -1) {
27907             perror(dp->d_name);
27908             continue;
27909         }
27910         if (fstat(ffd, &statbuf) == 0 && statbuf.st_size > (1024*1024)) {
27911             /* found it ... */
27912             printf("%s: %jdK\n", dp->d_name,
27913                 (intmax_t)(statbuf.st_size / 1024));
27914         }
27915         close(ffd);
27916     }
27917     closedir(d); // note this implicitly closes dfd
27918     return 0;
27919 }

```

APPLICATION USAGE

The *opendir()* function should be used in conjunction with *readdir()*, *closedir()*, and *rewinddir()* to examine the contents of the directory (see the EXAMPLES section in *readdir()*). This method is recommended for portability.

RATIONALE

The purpose of the *fdopendir()* function is to enable opening files in directories other than the current working directory without exposure to race conditions. Any part of the path of a file could be changed in parallel to a call to *opendir()*, resulting in unspecified behavior.

Based on historical implementations, the rules about file descriptors apply to directory streams as well. However, this volume of POSIX.1-2008 does not mandate that the directory stream be implemented using file descriptors. The description of *closedir()* clarifies that if a file descriptor is used for the directory stream, it is mandatory that *closedir()* deallocate the file descriptor. When a file descriptor is used to implement the directory stream, it behaves as if the *FD_CLOEXEC* had been set for the file descriptor.

The directory entries for dot and dot-dot are optional. This volume of POSIX.1-2008 does not provide a way to test *a priori* for their existence because an application that is portable must be written to look for (and usually ignore) those entries. Writing code that presumes that they are the first two entries does not always work, as many implementations permit them to be other than the first two entries, with a “normal” entry preceding them. There is negligible value in providing a way to determine what the implementation does because the code to deal with dot and dot-dot must be written in any case and because such a flag would add to the list of those flags (which has proven in itself to be objectionable) and might be abused.

Since the structure and buffer allocation, if any, for directory operations are defined by the implementation, this volume of POSIX.1-2008 imposes no portability requirements for erroneous program constructs, erroneous data, or the use of unspecified values such as the use or referencing of a *dirp* value or a **dirent** structure value after a directory stream has been closed or after a *fork()* or one of the *exec* function calls.

27947 **FUTURE DIRECTIONS**

27948 None.

27949 **SEE ALSO**27950 *closedir()*, *dirfd()*, *fstatat()*, *open()*, *readdir()*, *rewinddir()*, *symlink()*27951 XBD **<dirent.h>**, **<sys/types.h>**27952 **CHANGE HISTORY**

27953 First released in Issue 2.

27954 **Issue 6**27955 In the SYNOPSIS, the optional include of the **<sys/types.h>** header is removed.27956 The following new requirements on POSIX implementations derive from alignment with the
27957 Single UNIX Specification:27958 • The requirement to include **<sys/types.h>** has been removed. Although **<sys/types.h>** was
27959 required for conforming implementations of previous POSIX specifications, it was not
27960 required for UNIX applications.

27961 • The [ELOOP] mandatory error condition is added.

27962 • A second [ENAMETOOLONG] is added as an optional error condition.

27963 The following changes were made to align with the IEEE P1003.1a draft standard:

27964 • The [ELOOP] optional error condition is added.

27965 **Issue 7**

27966 Austin Group Interpretation 1003.1-2001 #143 is applied.

27967 SD5-XBD-ERN-4 is applied, changing the definition of the [EMFILE] error.

27968 The *fdopendir()* function is added from The Open Group Technical Standard, 2006, Extended API
27969 Set Part 2.

27970 An additional example is added.

27971 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0122 [422] and XSH/TC1-2008/0123 +
27972 [324] are applied.

27973 **NAME**27974 `feclearexcept` — clear floating-point exception27975 **SYNOPSIS**27976 `#include <fenv.h>`27977 `int feclearexcept(int excepts);`27978 **DESCRIPTION**

27979 CX The functionality described on this reference page is aligned with the ISO C standard. Any
27980 conflict between the requirements described here and the ISO C standard is unintentional. This
27981 volume of POSIX.1-2008 defers to the ISO C standard.

27982 The `feclearexcept()` function shall attempt to clear the supported floating-point exceptions
27983 represented by *excepts*.

27984 **RETURN VALUE**

27985 If the argument is zero or if all the specified exceptions were successfully cleared, `feclearexcept()`
27986 shall return zero. Otherwise, it shall return a non-zero value.

27987 **ERRORS**

27988 No errors are defined.

27989 **EXAMPLES**

27990 None.

27991 **APPLICATION USAGE**

27992 None.

27993 **RATIONALE**

27994 None.

27995 **FUTURE DIRECTIONS**

27996 None.

27997 **SEE ALSO**27998 [*fegetexceptflag\(\)*](#), [*feraiseexcept\(\)*](#), [*fetestexcept\(\)*](#)27999 XBD [**<fenv.h>**](#)28000 **CHANGE HISTORY**

28001 First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

28002 ISO/IEC 9899:1999 standard, Technical Corrigendum 1 is incorporated.

28003 **NAME**

28004 fegetenv, fesetenv — get and set current floating-point environment

28005 **SYNOPSIS**

```
28006 #include <fenv.h>
28007 int fegetenv(fenv_t *envp);
28008 int fesetenv(const fenv_t *envp);
```

28009 **DESCRIPTION**

28010 CX The functionality described on this reference page is aligned with the ISO C standard. Any
 28011 conflict between the requirements described here and the ISO C standard is unintentional. This
 28012 volume of POSIX.1-2008 defers to the ISO C standard.

28013 The *fegetenv()* function shall attempt to store the current floating-point environment in the object
 28014 pointed to by *envp*.

28015 The *fesetenv()* function shall attempt to establish the floating-point environment represented by
 28016 the object pointed to by *envp*. The argument *envp* shall point to an object set by a call to
 28017 *fegetenv()* or *fehldexcept()*, or equal a floating-point environment macro. The *fesetenv()* function
 28018 does not raise floating-point exceptions, but only installs the state of the floating-point status
 28019 flags represented through its argument.

28020 **RETURN VALUE**

28021 If the representation was successfully stored, *fegetenv()* shall return zero. Otherwise, it shall
 28022 return a non-zero value. If the environment was successfully established, *fesetenv()* shall return
 28023 zero. Otherwise, it shall return a non-zero value.

28024 **ERRORS**

28025 No errors are defined.

28026 **EXAMPLES**

28027 None.

28028 **APPLICATION USAGE**

28029 None.

28030 **RATIONALE**

28031 None.

28032 **FUTURE DIRECTIONS**

28033 None.

28034 **SEE ALSO**28035 *fehldexcept()*, *feupdateenv()*28036 XBD **<fenv.h>**28037 **CHANGE HISTORY**

28038 First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

28039 ISO/IEC 9899:1999 standard, Technical Corrigendum 1 is incorporated.

28040 **NAME**

28041 fegetexceptflag, fesetexceptflag — get and set floating-point status flags

28042 **SYNOPSIS**

28043 #include <fenv.h>

28044 int fegetexceptflag(fexcept_t *flagp, int excepts);

28045 int fesetexceptflag(const fexcept_t *flagp, int excepts);

28046 **DESCRIPTION**

28047 CX The functionality described on this reference page is aligned with the ISO C standard. Any
 28048 conflict between the requirements described here and the ISO C standard is unintentional. This
 28049 volume of POSIX.1-2008 defers to the ISO C standard.

28050 The *fegetexceptflag()* function shall attempt to store an implementation-defined representation of
 28051 the states of the floating-point status flags indicated by the argument *excepts* in the object
 28052 pointed to by the argument *flagp*.

28053 The *fesetexceptflag()* function shall attempt to set the floating-point status flags indicated by the
 28054 argument *excepts* to the states stored in the object pointed to by *flagp*. The value pointed to by
 28055 *flagp* shall have been set by a previous call to *fegetexceptflag()* whose second argument
 28056 represented at least those floating-point exceptions represented by the argument *excepts*. This
 28057 function does not raise floating-point exceptions, but only sets the state of the flags.

28058 **RETURN VALUE**

28059 If the representation was successfully stored, *fegetexceptflag()* shall return zero. Otherwise, it
 28060 shall return a non-zero value. If the *excepts* argument is zero or if all the specified exceptions
 28061 were successfully set, *fesetexceptflag()* shall return zero. Otherwise, it shall return a non-zero
 28062 value.

28063 **ERRORS**

28064 No errors are defined.

28065 **EXAMPLES**

28066 None.

28067 **APPLICATION USAGE**

28068 None.

28069 **RATIONALE**

28070 None.

28071 **FUTURE DIRECTIONS**

28072 None.

28073 **SEE ALSO**28074 *feclearexcept()*, *feraiseexcept()*, *fetestexcept()*

28075 XBD <fenv.h>

28076 **CHANGE HISTORY**

28077 First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

28078 ISO/IEC 9899:1999 standard, Technical Corrigendum 1 is incorporated.

NAME

fegetround, fesetround — get and set current rounding direction

SYNOPSIS

```
#include <fenv.h>

int fegetround(void);
int fesetround(int round);
```

DESCRIPTION

CX The functionality described on this reference page is aligned with the ISO C standard. Any conflict between the requirements described here and the ISO C standard is unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

The *fegetround()* function shall get the current rounding direction.

The *fesetround()* function shall establish the rounding direction represented by its argument *round*. If the argument is not equal to the value of a rounding direction macro, the rounding direction is not changed.

RETURN VALUE

The *fegetround()* function shall return the value of the rounding direction macro representing the current rounding direction or a negative value if there is no such rounding direction macro or the current rounding direction is not determinable.

The *fesetround()* function shall return a zero value if and only if the requested rounding direction was established.

ERRORS

No errors are defined.

EXAMPLES

The following example saves, sets, and restores the rounding direction, reporting an error and aborting if setting the rounding direction fails:

```
#include <fenv.h>
#include <assert.h>
void f(int round_dir)
{
    #pragma STDC FENV_ACCESS ON
    int save_round;
    int setround_ok;
    save_round = fegetround();
    setround_ok = fesetround(round_dir);
    assert(setround_ok == 0);
    /* ... */
    fesetround(save_round);
    /* ... */
}
```

APPLICATION USAGE

None.

RATIONALE

None.

28122 FUTURE DIRECTIONS

28123 None.

28124 SEE ALSO

28125 XBD [<fenv.h>](#)

28126 CHANGE HISTORY

28127 First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

28128 ISO/IEC 9899:1999 standard, Technical Corrigendum 1 is incorporated.

28129 **NAME**

28130 feholdexcept — save current floating-point environment

28131 **SYNOPSIS**

28132 #include <fenv.h>

28133 int feholdexcept(fenv_t *envp);

28134 **DESCRIPTION**

28135 CX The functionality described on this reference page is aligned with the ISO C standard. Any
28136 conflict between the requirements described here and the ISO C standard is unintentional. This
28137 volume of POSIX.1-2008 defers to the ISO C standard.

28138 The *feholdexcept()* function shall save the current floating-point environment in the object
28139 pointed to by *envp*, clear the floating-point status flags, and then install a non-stop (continue on
28140 floating-point exceptions) mode, if available, for all floating-point exceptions.

28141 **RETURN VALUE**

28142 The *feholdexcept()* function shall return zero if and only if non-stop floating-point exception
28143 handling was successfully installed.

28144 **ERRORS**

28145 No errors are defined.

28146 **EXAMPLES**

28147 None.

28148 **APPLICATION USAGE**

28149 None.

28150 **RATIONALE**

28151 The *feholdexcept()* function should be effective on typical IEC 60559:1989 standard
28152 implementations which have the default non-stop mode and at least one other mode for trap
28153 handling or aborting. If the implementation provides only the non-stop mode, then installing the
28154 non-stop mode is trivial.

28155 **FUTURE DIRECTIONS**

28156 None.

28157 **SEE ALSO**28158 *fegetenv()*, *feupdateenv()*

28159 XBD <fenv.h>

28160 **CHANGE HISTORY**

28161 First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

28162 **NAME**

28163 feof — test end-of-file indicator on a stream

28164 **SYNOPSIS**

28165 #include <stdio.h>

28166 int feof(FILE *stream);

28167 **DESCRIPTION**

28168 CX The functionality described on this reference page is aligned with the ISO C standard. Any
 28169 conflict between the requirements described here and the ISO C standard is unintentional. This
 28170 volume of POSIX.1-2008 defers to the ISO C standard.

28171 The *feof()* function shall test the end-of-file indicator for the stream pointed to by *stream*.28172 CX The *feof()* function shall not change the setting of *errno* if *stream* is valid.

+

28173 **RETURN VALUE**28174 The *feof()* function shall return non-zero if and only if the end-of-file indicator is set for *stream*.28175 **ERRORS**

28176 No errors are defined.

28177 **EXAMPLES**

28178 None.

28179 **APPLICATION USAGE**

28180 None.

28181 **RATIONALE**

28182 None.

28183 **FUTURE DIRECTIONS**

28184 None.

28185 **SEE ALSO**28186 *clearerr()*, *ferror()*, *fopen()*

28187 XBD <stdio.h>

28188 **CHANGE HISTORY**

28189 First released in Issue 1. Derived from Issue 1 of the SVID.

28190 **Issue 7**

28191 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0124 [401] is applied.

+

28192 **NAME**28193 `feraiseexcept` — raise floating-point exception28194 **SYNOPSIS**28195 `#include <fenv.h>`28196 `int fraiseexcept(int excepts);`28197 **DESCRIPTION**

28198 CX The functionality described on this reference page is aligned with the ISO C standard. Any
 28199 conflict between the requirements described here and the ISO C standard is unintentional. This
 28200 volume of POSIX.1-2008 defers to the ISO C standard.

28201 The `feraiseexcept()` function shall attempt to raise the supported floating-point exceptions
 28202 represented by the argument *excepts*. The order in which these floating-point exceptions are
 28203 raised is unspecified. Whether the `feraiseexcept()` function additionally raises the inexact floating-
 28204 point exception whenever it raises the overflow or underflow floating-point exception is
 28205 implementation-defined.

28206 **RETURN VALUE**

28207 If the argument is zero or if all the specified exceptions were successfully raised, `feraiseexcept()`
 28208 shall return zero. Otherwise, it shall return a non-zero value.

28209 **ERRORS**

28210 No errors are defined.

28211 **EXAMPLES**

28212 None.

28213 **APPLICATION USAGE**

28214 The effect is intended to be similar to that of floating-point exceptions raised by arithmetic
 28215 operations. Hence, enabled traps for floating-point exceptions raised by this function are taken.

28216 **RATIONALE**

28217 Raising overflow or underflow is allowed to also raise inexact because on some architectures the
 28218 only practical way to raise an exception is to execute an instruction that has the exception as a
 28219 side-effect. The function is not restricted to accept only valid coincident expressions for atomic
 28220 operations, so the function can be used to raise exceptions accrued over several operations.

28221 **FUTURE DIRECTIONS**

28222 None.

28223 **SEE ALSO**28224 `feclearexcept()`, `fesetexceptflag()`, `fetestexcept()`28225 XBD `<fenv.h>`28226 **CHANGE HISTORY**

28227 First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

28228 ISO/IEC 9899:1999 standard, Technical Corrigendum 1 is incorporated.

28229 **NAME**

28230 ferror — test error indicator on a stream

28231 **SYNOPSIS**

28232 #include <stdio.h>

28233 int ferror(FILE **stream*);28234 **DESCRIPTION**

28235 CX The functionality described on this reference page is aligned with the ISO C standard. Any
 28236 conflict between the requirements described here and the ISO C standard is unintentional. This
 28237 volume of POSIX.1-2008 defers to the ISO C standard.

28238 The *ferror()* function shall test the error indicator for the stream pointed to by *stream*.28239 CX The *ferror()* function shall not change the setting of *errno* if *stream* is valid.

+

28240 **RETURN VALUE**28241 The *ferror()* function shall return non-zero if and only if the error indicator is set for *stream*.28242 **ERRORS**

28243 No errors are defined.

28244 **EXAMPLES**

28245 None.

28246 **APPLICATION USAGE**

28247 None.

28248 **RATIONALE**

28249 None.

28250 **FUTURE DIRECTIONS**

28251 None.

28252 **SEE ALSO**28253 *clearerr()*, *feof()*, *fopen()*

28254 XBD <stdio.h>

28255 **CHANGE HISTORY**

28256 First released in Issue 1. Derived from Issue 1 of the SVID.

28257 **Issue 7**

28258 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0125 [401] is applied.

+

28259 **NAME**

28260 fesetenv — set current floating-point environment

28261 **SYNOPSIS**

28262 #include <fenv.h>

28263 int fesetenv(const fenv_t *envp);

28264 **DESCRIPTION**

28265 Refer to *fegetenv()*.

28266 **NAME**

28267 fesetexceptflag — set floating-point status flags

28268 **SYNOPSIS**

28269 #include <fenv.h>

28270 int fesetexceptflag(const fexcept_t *flagp, int excepts);

28271 **DESCRIPTION**28272 Refer to *fegetexceptflag()*.

28273 **NAME**

28274 fesetround — set current rounding direction

28275 **SYNOPSIS**

28276 #include <fenv.h>

28277 int fesetround(int *round*);

28278 **DESCRIPTION**

28279 Refer to *fegetround()*.

28280 **NAME**

28281 fetestexcept — test floating-point exception flags

28282 **SYNOPSIS**

28283 #include <fenv.h>

28284 int fetestexcept(int *excepts*);28285 **DESCRIPTION**

28286 CX The functionality described on this reference page is aligned with the ISO C standard. Any
 28287 conflict between the requirements described here and the ISO C standard is unintentional. This
 28288 volume of POSIX.1-2008 defers to the ISO C standard.

28289 The *fetestexcept()* function shall determine which of a specified subset of the floating-point
 28290 exception flags are currently set. The *excepts* argument specifies the floating-point status flags to
 28291 be queried.

28292 **RETURN VALUE**

28293 The *fetestexcept()* function shall return the value of the bitwise-inclusive OR of the floating-point
 28294 exception macros corresponding to the currently set floating-point exceptions included in
 28295 *excepts*.

28296 **ERRORS**

28297 No errors are defined.

28298 **EXAMPLES**

28299 The following example calls function *f()* if an invalid exception is set, and then function *g()* if an
 28300 overflow exception is set:

```
28301 #include <fenv.h>
28302 /* ... */
28303 {
28304     #pragma STDC FENV_ACCESS ON
28305     int set_excepts;
28306     feclearexcept(FE_INVALID | FE_OVERFLOW);
28307     // maybe raise exceptions
28308     set_excepts = fetestexcept(FE_INVALID | FE_OVERFLOW);
28309     if (set_excepts & FE_INVALID) f();
28310     if (set_excepts & FE_OVERFLOW) g();
28311     /* ... */
28312 }
```

28313 **APPLICATION USAGE**

28314 None.

28315 **RATIONALE**

28316 None.

28317 **FUTURE DIRECTIONS**

28318 None.

28319 **SEE ALSO**28320 *feclearexcept()*, *fegetexceptflag()*, *feraiseexcept()*

28321 XBD <fenv.h>

28322	CHANGE HISTORY
28323	First released in Issue 6. Derived from the ISO/IEC 9899: 1999 standard.

28324 **NAME**

28325 feupdateenv — update floating-point environment

28326 **SYNOPSIS**

28327 #include <fenv.h>

28328 int feupdateenv(const fenv_t *envp);

28329 **DESCRIPTION**

28330 CX The functionality described on this reference page is aligned with the ISO C standard. Any
 28331 conflict between the requirements described here and the ISO C standard is unintentional. This
 28332 volume of POSIX.1-2008 defers to the ISO C standard.

28333 The *feupdateenv()* function shall attempt to save the currently raised floating-point exceptions in
 28334 its automatic storage, attempt to install the floating-point environment represented by the object
 28335 pointed to by *envp*, and then attempt to raise the saved floating-point exceptions. The argument
 28336 *envp* shall point to an object set by a call to *feholdexcept()* or *fegetenv()*, or equal a floating-point
 28337 environment macro.

28338 **RETURN VALUE**

28339 The *feupdateenv()* function shall return a zero value if and only if all the required actions were
 28340 successfully carried out.

28341 **ERRORS**

28342 No errors are defined.

28343 **EXAMPLES**

28344 The following example shows sample code to hide spurious underflow floating-point
 28345 exceptions:

```
28346 #include <fenv.h>
28347 double f(double x)
28348 {
28349     #pragma STDC FENV_ACCESS ON
28350     double result;
28351     fenv_t save_env;
28352     feholdexcept(&save_env);
28353     // compute result
28354     if (/* test spurious underflow */)
28355         feclearexcept(FE_UNDERFLOW);
28356     feupdateenv(&save_env);
28357     return result;
28358 }
```

28359 **APPLICATION USAGE**

28360 None.

28361 **RATIONALE**

28362 None.

28363 **FUTURE DIRECTIONS**

28364 None.

28365 **SEE ALSO**28366 *fegetenv()*, *feholdexcept()*

28367 XBD <fenv.h>

CHANGE HISTORY

- 28368
- 28369 First released in Issue 6. Derived from the ISO/IEC 9899: 1999 standard.
- 28370 ISO/IEC 9899: 1999 standard, Technical Corrigendum 1 is incorporated.

28371 **NAME**

28372 fexecve — execute a file

28373 **SYNOPSIS**

28374 #include <unistd.h>

28375 int fexecve(int *fd*, char *const *argv*[], char *const *envp*[]);28376 **DESCRIPTION**28377 Refer to *exec*.

28378 **NAME**

28379 fflush — flush a stream

28380 **SYNOPSIS**

28381 #include <stdio.h>

28382 int fflush(FILE *stream);

28383 **DESCRIPTION**

28384 CX The functionality described on this reference page is aligned with the ISO C standard. Any
 28385 conflict between the requirements described here and the ISO C standard is unintentional. This
 28386 volume of POSIX.1-2008 defers to the ISO C standard.

28387 If *stream* points to an output stream or an update stream in which the most recent operation was
 28388 CX not input, *fflush()* shall cause any unwritten data for that stream to be written to the file, and the
 28389 last data modification and last file status change timestamps of the underlying file shall be
 28390 marked for update.

28391 If *stream* is a null pointer, *fflush()* shall perform this flushing action on all streams for which the
 28392 behavior is defined above.

28393 CX For a stream open for reading, if the file is not already at EOF, and the file is one capable of
 28394 seeking, the file offset of the underlying open file description shall be set to the file position of
 28395 the stream, and any characters pushed back onto the stream by *ungetc()* or *ungetwc()* that have
 28396 not subsequently been read from the stream shall be discarded (without further changing the file
 28397 offset).

28398 **RETURN VALUE**

28399 Upon successful completion, *fflush()* shall return 0; otherwise, it shall set the error indicator for
 28400 CX the stream, return EOF, and set *errno* to indicate the error.

28401 **ERRORS**28402 The *fflush()* function shall fail if:

28403 CX [EAGAIN] The O_NONBLOCK flag is set for the file descriptor underlying *stream* and
 28404 the thread would be delayed in the write operation.

28405 CX [EBADF] The file descriptor underlying *stream* is not valid.

28406 CX [EFBIG] An attempt was made to write a file that exceeds the maximum file size.

28407 XSI [EFBIG] An attempt was made to write a file that exceeds the file size limit of the
 28408 process.

28409 CX [EFBIG] The file is a regular file and an attempt was made to write at or beyond the
 28410 offset maximum associated with the corresponding stream.

28411 CX [EINTR] The *fflush()* function was interrupted by a signal.

28412 CX [EIO] The process is a member of a background process group attempting to write to
 28413 its controlling terminal, TOSTOP is set, the calling thread is not blocking
 28414 SIGTTOU, the process is not ignoring SIGTTOU, and the process group of the
 28415 process is orphaned. This error may also be returned under implementation-
 28416 defined conditions.

28417 CX [ENOMEM] The underlying stream was created by *open_memstream()* or
 28418 *open_wmemstream()* and insufficient memory is available.

28419 CX [ENOSPC] There was no free space remaining on the device containing the file or in the
 28420 buffer used by the *fmemopen()* function.

28421 CX [EPIPE] An attempt is made to write to a pipe or FIFO that is not open for reading by
 28422 any process. A SIGPIPE signal shall also be sent to the thread.

28423 The *fflush()* function may fail if:

28424 CX [ENXIO] A request was made of a nonexistent device, or the request was outside the
 28425 capabilities of the device.

28426 EXAMPLES

28427 Sending Prompts to Standard Output

28428 The following example uses *printf()* calls to print a series of prompts for information the user
 28429 must enter from standard input. The *fflush()* calls force the output to standard output. The
 28430 *fflush()* function is used because standard output is usually buffered and the prompt may not
 28431 immediately be printed on the output or terminal. The *getline()* function calls read strings from
 28432 standard input and place the results in variables, for use later in the program.

```

28433 char *user;
28434 char *oldpasswd;
28435 char *newpasswd;
28436 ssize_t llen;
28437 size_t blen;
28438 struct termios term;
28439 tcflag_t saveflag;

28440 printf("User name: ");
28441 fflush(stdout);
28442 blen = 0;
28443 llen = getline(&user, &blen, stdin);
28444 user[llen-1] = 0;
28445 tcgetattr(fileno(stdin), &term);
28446 saveflag = term.c_lflag;
28447 term.c_lflag &= ~ECHO;
28448 tcsetattr(fileno(stdin), TCSANOW, &term);
28449 printf("Old password: ");
28450 fflush(stdout);
28451 blen = 0;
28452 llen = getline(&oldpasswd, &blen, stdin);
28453 oldpasswd[llen-1] = 0;

28454 printf("\nNew password: ");
28455 fflush(stdout);
28456 blen = 0;
28457 llen = getline(&newpasswd, &blen, stdin);
28458 newpasswd[llen-1] = 0;
28459 term.c_lflag = saveflag;
28460 tcsetattr(fileno(stdin), TCSANOW, &term);
28461 free(user);
28462 free(oldpasswd);
28463 free(newpasswd);

```

28464 **APPLICATION USAGE**

28465 None.

28466 **RATIONALE**

28467 Data buffered by the system may make determining the validity of the position of the current
 28468 file descriptor impractical. Thus, enforcing the repositioning of the file descriptor after *fflush()*
 28469 on streams open for *read()* is not mandated by POSIX.1-2008.

28470 **FUTURE DIRECTIONS**

28471 None.

28472 **SEE ALSO**28473 [Section 2.5](#) (on page 495), *fmemopen()*, *getrlimit()*, *open_memstream()*, *ulimit()*28474 XBD [<stdio.h>](#)28475 **CHANGE HISTORY**

28476 First released in Issue 1. Derived from Issue 1 of the SVID.

28477 **Issue 5**

28478 Large File Summit extensions are added.

28479 **Issue 6**

28480 Extensions beyond the ISO C standard are marked.

28481 The following new requirements on POSIX implementations derive from alignment with the
 28482 Single UNIX Specification:

- 28483 • The [EFBIG] error is added as part of the large file support extensions.
- 28484 • The [ENXIO] optional error condition is added.

28485 The RETURN VALUE section is updated to note that the error indicator shall be set for the
 28486 stream. This is for alignment with the ISO/IEC 9899:1999 standard.

28487 IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/31 is applied, updating the [EAGAIN]
 28488 error in the ERRORS section from “the process would be delayed” to “the thread would be
 28489 delayed”.

28490 **Issue 7**

28491 Austin Group Interpretation 1003.1-2001 #002 is applied, clarifying the interaction of file
 28492 descriptors and streams.

28493 The [ENOSPC] error condition is updated and the [ENOMEM] error is added from The Open
 28494 Group Technical Standard, 2006, Extended API Set Part 1.

28495 The EXAMPLES section is revised.

28496 Changes are made related to support for finegrained timestamps.

28497 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0126 [87], XSH/TC1-2008/0127 [79], +
 28498 and XSH/TC1-2008/0128 [14] are applied.

28499 NAME

28500 ffs — find first set bit

28501 SYNOPSIS

```
28502 XSI        #include <strings.h>  
28503        int ffs(int i);
```

28504 DESCRIPTION

28505 The *ffs()* function shall find the first bit set (beginning with the least significant bit) in *i*, and
28506 return the index of that bit. Bits are numbered starting at one (the least significant bit).

28507 RETURN VALUE

28508 The *ffs()* function shall return the index of the first bit set. If *i* is 0, then *ffs()* shall return 0.

28509 ERRORS

28510 No errors are defined.

28511 EXAMPLES

28512 None.

28513 APPLICATION USAGE

28514 None.

28515 RATIONALE

28516 None.

28517 FUTURE DIRECTIONS

28518 None.

28519 SEE ALSO

28520 XBD [<strings.h>](#)

28521 CHANGE HISTORY

28522 First released in Issue 4, Version 2.

28523 Issue 5

28524 Moved from X/OPEN UNIX extension to BASE.

28525 **NAME**

28526 fgetc — get a byte from a stream

28527 **SYNOPSIS**

28528 #include <stdio.h>

28529 int fgetc(FILE *stream);

28530 **DESCRIPTION**

28531 CX The functionality described on this reference page is aligned with the ISO C standard. Any
 28532 conflict between the requirements described here and the ISO C standard is unintentional. This
 28533 volume of POSIX.1-2008 defers to the ISO C standard.

28534 If the end-of-file indicator for the input stream pointed to by *stream* is not set and a next byte is
 28535 present, the *fgetc()* function shall obtain the next byte as an **unsigned char** converted to an **int**,
 28536 from the input stream pointed to by *stream*, and advance the associated file position indicator for
 28537 the stream (if defined). Since *fgetc()* operates on bytes, reading a character consisting of multiple
 28538 bytes (or “a multi-byte character”) may require multiple calls to *fgetc()*.

28539 CX The *fgetc()* function may mark the last data access timestamp of the file associated with *stream*
 28540 for update. The last data access timestamp shall be marked for update by the first successful
 28541 execution of *fgetc()*, *fgets()*, *fread()*, *fscanf()*, *getc()*, *getchar()*, *getdelim()*, *getline()*, *gets()*, or
 28542 *scanf()* using *stream* that returns data not supplied by a prior call to *ungetc()*.

28543 **RETURN VALUE**

28544 Upon successful completion, *fgetc()* shall return the next byte from the input stream pointed to
 28545 by *stream*. If the end-of-file indicator for the stream is set, or if the stream is at end-of-file, the
 28546 end-of-file indicator for the stream shall be set and *fgetc()* shall return EOF. If a read error occurs,
 28547 CX the error indicator for the stream shall be set, *fgetc()* shall return EOF, and shall set *errno* to
 28548 indicate the error.

28549 **ERRORS**28550 The *fgetc()* function shall fail if data needs to be read and:

28551 CX [EAGAIN] The O_NONBLOCK flag is set for the file descriptor underlying *stream* and
 28552 the thread would be delayed in the *fgetc()* operation.

28553 CX [EBADF] The file descriptor underlying *stream* is not a valid file descriptor open for
 28554 reading.

28555 CX [EINTR] The read operation was terminated due to the receipt of a signal, and no data
 28556 was transferred.

28557 CX [EIO] A physical I/O error has occurred, or the process is in a background process
 28558 group attempting to read from its controlling terminal, and either the calling
 28559 thread is blocking SIGTTIN or the process is ignoring SIGTTIN or the process
 28560 group of the process is orphaned. This error may also be generated for
 28561 implementation-defined reasons.

28562 CX [EOVERFLOW] The file is a regular file and an attempt was made to read at or beyond the
 28563 offset maximum associated with the corresponding stream.

28564 The *fgetc()* function may fail if:

28565 CX [ENOMEM] Insufficient storage space is available.

28566 CX [ENXIO] A request was made of a nonexistent device, or the request was outside the
 28567 capabilities of the device.

EXAMPLES

None.

APPLICATION USAGE

If the integer value returned by *fgetc()* is stored into a variable of type **char** and then compared against the integer constant EOF, the comparison may never succeed, because sign-extension of a variable of type **char** on widening to integer is implementation-defined.

The *ferror()* or *feof()* functions must be used to distinguish between an error condition and an end-of-file condition.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

Section 2.5 (on page 495), *feof()*, *ferror()*, *fgets()*, *fread()*, *fscanf()*, *getchar()*, *getc()*, *gets()*, *ungetc()*

XBD <stdio.h>

CHANGE HISTORY

First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 5

Large File Summit extensions are added.

Issue 6

Extensions beyond the ISO C standard are marked.

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The [EIO] and [EOVERFLOW] mandatory error conditions are added.
- The [ENOMEM] and [ENXIO] optional error conditions are added.

The following changes are made for alignment with the ISO/IEC 9899:1999 standard:

- The DESCRIPTION is updated to clarify the behavior when the end-of-file indicator for the input stream is not set.
- The RETURN VALUE section is updated to note that the error indicator shall be set for the stream.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/32 is applied, updating the [EAGAIN] error in the ERRORS section from “the process would be delayed” to “the thread would be delayed”.

Issue 7

Austin Group Interpretation 1003.1-2001 #051 is applied, updating the list of functions that mark the last data access timestamp for update.

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0129 [79] and XSH/TC1-2008/0130 + [14] are applied.

28606 **NAME**28607 `fgetpos` — get current file position information28608 **SYNOPSIS**28609 `#include <stdio.h>`28610 `int fgetpos(FILE *restrict stream, fpos_t *restrict pos);`28611 **DESCRIPTION**

28612 CX The functionality described on this reference page is aligned with the ISO C standard. Any
 28613 conflict between the requirements described here and the ISO C standard is unintentional. This
 28614 volume of POSIX.1-2008 defers to the ISO C standard.

28615 The `fgetpos()` function shall store the current values of the parse state (if any) and file position
 28616 indicator for the stream pointed to by *stream* in the object pointed to by *pos*. The value stored
 28617 contains unspecified information usable by `fsetpos()` for repositioning the stream to its position
 28618 at the time of the call to `fgetpos()`.

28619 The `fgetpos()` function shall not change the setting of *errno* if successful.

+

28620 **RETURN VALUE**

28621 Upon successful completion, `fgetpos()` shall return 0; otherwise, it shall return a non-zero value
 28622 and set *errno* to indicate the error.

28623 **ERRORS**28624 The `fgetpos()` function shall fail if:

28625 CX [EBADF] The file descriptor underlying *stream* is not valid.

-

28626 CX [EOVERFLOW] The current value of the file position cannot be represented correctly in an
 28627 object of type `fpos_t`.

+

28628 CX [ESPIPE] The file descriptor underlying *stream* is associated with a pipe, FIFO, or socket.

28629 **EXAMPLES**

28630 None.

28631 **APPLICATION USAGE**

28632 None.

28633 **RATIONALE**

28634 None.

28635 **FUTURE DIRECTIONS**

28636 None.

28637 **SEE ALSO**28638 [Section 2.5](#) (on page 495), `fopen()`, `ftell()`, `rewind()`, `ungetc()`28639 XBD `<stdio.h>`28640 **CHANGE HISTORY**

28641 First released in Issue 4. Derived from the ISO C standard.

28642 **Issue 5**

28643 Large File Summit extensions are added.

Issue 6

Extensions beyond the ISO C standard are marked.

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The [EBADF] and [ESPIPE] optional error conditions are added.

An additional [ESPIPE] error condition is added for sockets.

The prototype for *fgetpos()* is changed for alignment with the ISO/IEC 9899:1999 standard.

Issue 7

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0131 [105], XSH/TC1-2008/0132 [122], + and XSH/TC1-2008/0133 [14] are applied.

28654 **NAME**28655 `fgets` — get a string from a stream28656 **SYNOPSIS**28657 `#include <stdio.h>`28658 `char *fgets(char *restrict s, int n, FILE *restrict stream);`28659 **DESCRIPTION**

28660 CX The functionality described on this reference page is aligned with the ISO C standard. Any
 28661 conflict between the requirements described here and the ISO C standard is unintentional. This
 28662 volume of POSIX.1-2008 defers to the ISO C standard.

28663 The `fgets()` function shall read bytes from *stream* into the array pointed to by *s*, until *n*−1 bytes
 28664 are read, or a <newline> is read and transferred to *s*, or an end-of-file condition is encountered.
 28665 The string is then terminated with a null byte.

28666 CX The `fgets()` function may mark the last data access timestamp of the file associated with *stream*
 28667 for update. The last data access timestamp shall be marked for update by the first successful
 28668 execution of `fgetc()`, `fgets()`, `fread()`, `fscanf()`, `getc()`, `getchar()`, `getdelim()`, `getline()`, `gets()`, or
 28669 `scanf()` using *stream* that returns data not supplied by a prior call to `ungetc()`.

28670 **RETURN VALUE**

28671 Upon successful completion, `fgets()` shall return *s*. If the stream is at end-of-file, the end-of-file
 28672 indicator for the stream shall be set and `fgets()` shall return a null pointer. If a read error occurs,
 28673 CX the error indicator for the stream shall be set, `fgets()` shall return a null pointer, and shall set
 28674 `errno` to indicate the error.

28675 **ERRORS**28676 Refer to `fgetc()`.28677 **EXAMPLES**28678 **Reading Input**

28679 The following example uses `fgets()` to read lines of input. It assumes that the file it is reading is a
 28680 text file and that lines in this text file are no longer than 16384 (or {LINE_MAX} if it is less than
 28681 16384 on the implementation where it is running) bytes long. (Note that the standard utilities
 28682 have no line length limit if `sysconf(_SC_LINE_MAX)` returns −1 without setting `errno`. This
 28683 example assumes that `sysconf(_SC_LINE_MAX)` will not fail.)

```

28684 #include <limits.h>
28685 #include <stdio.h>
28686 #include <unistd.h>
28687 #define MYLIMIT 16384
28688
28688 char *line;
28689 int line_max;
28690 if (LINE_MAX >= MYLIMIT) {
28691     // Use maximum line size of MYLIMIT. If LINE_MAX is
28692     // bigger than our limit, sysconf() can't report a
28693     // smaller limit.
28694     line_max = MYLIMIT;
28695 } else {
28696     long limit = sysconf(_SC_LINE_MAX);
28697     line_max = (limit < 0 || limit > MYLIMIT) ? MYLIMIT : (int)limit;
28698 }
```



```

28699 // line_max + 1 leaves room for the null byte added by fgets().
28700 line = malloc(line_max + 1);
28701 if (line == NULL) {
28702     // out of space
28703     ...
28704     return error;
28705 }
28706 while (fgets(line, line_max + 1, fp) != NULL) {
28707     // Verify that a full line has been read ...
28708     // If not, report an error or prepare to treat the
28709     // next time through the loop as a read of a
28710     // continuation of the current line.
28711     ...
28712     // Process line ...
28713     ...
28714 }
28715 free(line);
28716 ...

```

28717 **APPLICATION USAGE**
28718 None.

28719 **RATIONALE**
28720 None.

28721 **FUTURE DIRECTIONS**
28722 None.

28723 **SEE ALSO**
28724 Section 2.5 (on page 495), *fgetc()*, *fopen()*, *fread()*, *fscanf()*, *getc()*, *getchar()*, *getdelim()*, *gets()*,
28725 *ungetc()*
28726 XBD <stdio.h>

28727 **CHANGE HISTORY**
28728 First released in Issue 1. Derived from Issue 1 of the SVID.

28729 **Issue 6**
28730 Extensions beyond the ISO C standard are marked.
28731 The prototype for *fgets()* is changed for alignment with the ISO/IEC 9899:1999 standard.

28732 **Issue 7**
28733 Austin Group Interpretation 1003.1-2001 #051 is applied, updating the list of functions that mark
28734 the last data access timestamp for update.
28735 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0134 [182] and XSH/TC1-2008/0135 +
28736 [14] are applied.

28737 **NAME**

28738 fgetwc — get a wide-character code from a stream

28739 **SYNOPSIS**

28740 #include <stdio.h>

28741 #include <wchar.h>

28742 wint_t fgetwc(FILE *stream);

28743 **DESCRIPTION**

28744 CX The functionality described on this reference page is aligned with the ISO C standard. Any
 28745 conflict between the requirements described here and the ISO C standard is unintentional. This
 28746 volume of POSIX.1-2008 defers to the ISO C standard.

28747 The *fgetwc()* function shall obtain the next character (if present) from the input stream pointed to
 28748 by *stream*, convert that to the corresponding wide-character code, and advance the associated file
 28749 position indicator for the stream (if defined).

28750 If an error occurs, the resulting value of the file position indicator for the stream is unspecified.

28751 CX The *fgetwc()* function may mark the last data access timestamp of the file associated with *stream*
 28752 for update. The last data access timestamp shall be marked for update by the first successful
 28753 execution of *fgetwc()*, *fgetws()*, *fwscanf()*, *getwc()*, *getwchar()*, *vfwscanf()*, *vfwscanf()*, or *wscanf()*
 28754 using *stream* that returns data not supplied by a prior call to *ungetwc()*.

28755 The *fgetwc()* function shall not change the setting of *errno* if successful.

+

28756 **RETURN VALUE**

28757 Upon successful completion, the *fgetwc()* function shall return the wide-character code of the
 28758 character read from the input stream pointed to by *stream* converted to a type **wint_t**. If the end-
 28759 of-file indicator for the stream is set, or if the stream is at end-of-file, the end-of-file indicator for
 28760 the stream shall be set and *fgetwc()* shall return WEOF. If a read error occurs, the error indicator
 28761 CX for the stream shall be set, *fgetwc()* shall return WEOF, and shall set *errno* to indicate the error. If
 28762 an encoding error occurs, the error indicator for the stream shall be set, *fgetwc()* shall return
 28763 WEOF, and shall set *errno* to indicate the error.

28764 **ERRORS**

28765 The *fgetwc()* function shall fail if data needs to be read and:

28766 CX [EAGAIN] The O_NONBLOCK flag is set for the file descriptor underlying *stream* and
 28767 the thread would be delayed in the *fgetwc()* operation.

28768 CX [EBADF] The file descriptor underlying *stream* is not a valid file descriptor open for
 28769 reading.

28770 [EILSEQ] The data obtained from the input stream does not form a valid character.

28771 CX [EINTR] The read operation was terminated due to the receipt of a signal, and no data
 28772 was transferred.

28773 CX [EIO] A physical I/O error has occurred, or the process is in a background process
 28774 group attempting to read from its controlling terminal, and either the calling
 28775 thread is blocking SIGTTIN or the process is ignoring SIGTTIN or the process
 28776 group of the process is orphaned. This error may also be generated for
 28777 implementation-defined reasons.

28778 CX [EOVERFLOW] The file is a regular file and an attempt was made to read at or beyond the
 28779 offset maximum associated with the corresponding stream.

28780 The *fgetwc()* function may fail if:

28781 CX [ENOMEM] Insufficient storage space is available.

28782 CX [ENXIO] A request was made of a nonexistent device, or the request was outside the
28783 capabilities of the device.

28784 EXAMPLES

28785 None.

28786 APPLICATION USAGE

28787 The *ferror()* or *feof()* functions must be used to distinguish between an error condition and an
28788 end-of-file condition.

28789 RATIONALE

28790 None.

28791 FUTURE DIRECTIONS

28792 None.

28793 SEE ALSO

28794 [Section 2.5](#) (on page 495), *feof()*, *ferror()*, *fopen()*

28795 XBD [<stdio.h>](#), [<wchar.h>](#)

28796 CHANGE HISTORY

28797 First released in Issue 4. Derived from the MSE working draft.

28798 Issue 5

28799 The Optional Header (OH) marking is removed from [<stdio.h>](#).

28800 Large File Summit extensions are added.

28801 Issue 6

28802 Extensions beyond the ISO C standard are marked.

28803 The following new requirements on POSIX implementations derive from alignment with the
28804 Single UNIX Specification:

- 28805 • The [EIO] and [EOVERFLOW] mandatory error conditions are added.
- 28806 • The [ENOMEM] and [ENXIO] optional error conditions are added.

28807 IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/33 is applied, updating the [EAGAIN]
28808 error in the ERRORS section from “the process would be delayed” to “the thread would be
28809 delayed”.

28810 Issue 7

28811 Austin Group Interpretation 1003.1-2001 #051 is applied, clarifying the RETURN VALUE section.

28812 Changes are made related to support for finegrained timestamps.

28813 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0136 [105], XSH/TC1-2008/0137 [79], +
28814 and XSH/TC1-2008/0138 [14] are applied.

28815 **NAME**

28816 fgetws — get a wide-character string from a stream

28817 **SYNOPSIS**

28818 #include <stdio.h>

28819 #include <wchar.h>

28820 wchar_t *fgetws(wchar_t *restrict ws, int n,

28821 FILE *restrict stream);

28822 **DESCRIPTION**

28823 CX The functionality described on this reference page is aligned with the ISO C standard. Any
 28824 conflict between the requirements described here and the ISO C standard is unintentional. This
 28825 volume of POSIX.1-2008 defers to the ISO C standard.

28826 The *fgetws()* function shall read characters from the *stream*, convert these to the corresponding
 28827 wide-character codes, place them in the **wchar_t** array pointed to by *ws*, until *n*–1 characters are
 28828 read, or a <newline> is read, converted, and transferred to *ws*, or an end-of-file condition is
 28829 encountered. The wide-character string, *ws*, shall then be terminated with a null wide-character
 28830 code.

28831 If an error occurs, the resulting value of the file position indicator for the stream is unspecified.

28832 CX The *fgetws()* function may mark the last data access timestamp of the file associated with *stream*
 28833 for update. The last data access timestamp shall be marked for update by the first successful
 28834 execution of *fgetwc()*, *fgetws()*, *fwscanf()*, *getwc()*, *getwchar()*, *vfwscanf()*, *vwscanf()*, or *wscanf()*
 28835 using *stream* that returns data not supplied by a prior call to *ungetwc()*.

28836 **RETURN VALUE**

28837 Upon successful completion, *fgetws()* shall return *ws*. If the end-of-file indicator for the stream is
 28838 set, or if the stream is at end-of-file, the end-of-file indicator for the stream shall be set and
 28839 *fgetws()* shall return a null pointer. If a read error occurs, the error indicator for the stream shall
 28840 CX be set, *fgetws()* shall return a null pointer, and shall set *errno* to indicate the error.

28841 **ERRORS**28842 Refer to *fgetwc()*.28843 **EXAMPLES**

28844 None.

28845 **APPLICATION USAGE**

28846 None.

28847 **RATIONALE**

28848 None.

28849 **FUTURE DIRECTIONS**

28850 None.

28851 **SEE ALSO**28852 Section 2.5 (on page 495), *fopen()*, *fread()*

28853 XBD <stdio.h>, <wchar.h>

28854 **CHANGE HISTORY**

28855 First released in Issue 4. Derived from the MSE working draft.

28856 **Issue 5**

28857 The Optional Header (OH) marking is removed from `<stdio.h>`.

28858 **Issue 6**

28859 Extensions beyond the ISO C standard are marked.

28860 The prototype for `fgetws()` is changed for alignment with the ISO/IEC 9899:1999 standard.

28861 **Issue 7**

28862 Austin Group Interpretation 1003.1-2001 #051 is applied, clarifying the RETURN VALUE section.

28863 Changes are made related to support for finegrained timestamps.

28864 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0139 [14] is applied.

+

28865 **NAME**

28866 `fileno` — map a stream pointer to a file descriptor

28867 **SYNOPSIS**

```
28868 CX      #include <stdio.h>
28869          int fileno(FILE *stream);
```

28870 **DESCRIPTION**

28871 The `fileno()` function shall return the integer file descriptor associated with the stream pointed to
28872 by `stream`.

28873 **RETURN VALUE**

28874 Upon successful completion, `fileno()` shall return the integer value of the file descriptor
28875 associated with `stream`. Otherwise, the value `-1` shall be returned and `errno` set to indicate the
28876 error.

28877 **ERRORS**

28878 The `fileno()` function may fail if:

28879	[EBADF]	The <code>stream</code> argument is not a valid stream, or the stream is not associated with
28880		a file.

28881 **EXAMPLES**

28882 None.

28883 **APPLICATION USAGE**

28884 None.

28885 **RATIONALE**

28886 Without some specification of which file descriptors are associated with these streams, it is
28887 impossible for an application to set up the streams for another application it starts with `fork()`
28888 and `exec`. In particular, it would not be possible to write a portable version of the `sh` command
28889 interpreter (although there may be other constraints that would prevent that portability).

28890 **FUTURE DIRECTIONS**

28891 None.

28892 **SEE ALSO**

28893 [Section 2.5.1](#) (on page 496), [dirfd\(\)](#), [fdopen\(\)](#), [fopen\(\)](#), [stdin](#)
28894 XBD [<stdio.h>](#)

28895 **CHANGE HISTORY**

28896 First released in Issue 1. Derived from Issue 1 of the SVID.

28897 **Issue 6**

28898 The following new requirements on POSIX implementations derive from alignment with the
28899 Single UNIX Specification:

- 28900 • The [EBADF] optional error condition is added.

28901 **Issue 7**

28902 SD5-XBD-ERN-99 is applied, changing the definition of the [EBADF] error.

28903 **NAME**

28904 flockfile, ftrylockfile, funlockfile — stdio locking functions

28905 **SYNOPSIS**

```
28906 CX      #include <stdio.h>
28907          void flockfile(FILE *file);
28908          int ftrylockfile(FILE *file);
28909          void funlockfile(FILE *file);
```

28910 **DESCRIPTION**

28911 These functions shall provide for explicit application-level locking of stdio (**FILE ***) objects.
 28912 These functions can be used by a thread to delineate a sequence of I/O statements that are
 28913 executed as a unit.

28914 The *flockfile()* function shall acquire for a thread ownership of a (**FILE ***) object.

28915 The *ftrylockfile()* function shall acquire for a thread ownership of a (**FILE ***) object if the object is
 28916 available; *ftrylockfile()* is a non-blocking version of *flockfile()*.

28917 The *funlockfile()* function shall relinquish the ownership granted to the thread. The behavior is
 28918 undefined if a thread other than the current owner calls the *funlockfile()* function.

28919 The functions shall behave as if there is a lock count associated with each (**FILE ***) object. This
 28920 count is implicitly initialized to zero when the (**FILE ***) object is created. The (**FILE ***) object is
 28921 unlocked when the count is zero. When the count is positive, a single thread owns the (**FILE ***)
 28922 object. When the *flockfile()* function is called, if the count is zero or if the count is positive and
 28923 the caller owns the (**FILE ***) object, the count shall be incremented. Otherwise, the calling thread
 28924 shall be suspended, waiting for the count to return to zero. Each call to *funlockfile()* shall
 28925 decrement the count. This allows matching calls to *flockfile()* (or successful calls to *ftrylockfile()*)
 28926 and *funlockfile()* to be nested.

28927 All functions that reference (**FILE ***) objects, except those with names ending in *_unlocked*, shall
 28928 behave as if they use *flockfile()* and *funlockfile()* internally to obtain ownership of these (**FILE ***)
 28929 objects.

28930 **RETURN VALUE**

28931 None for *flockfile()* and *funlockfile()*.

28932 The *ftrylockfile()* function shall return zero for success and non-zero to indicate that the lock
 28933 cannot be acquired.

28934 **ERRORS**

28935 No errors are defined.

28936 **EXAMPLES**

28937 None.

28938 **APPLICATION USAGE**

28939 Applications using these functions may be subject to priority inversion, as discussed in XBD
 28940 [Section 3.287](#) (on page 79).

28941 **RATIONALE**

28942 The *flockfile()* and *funlockfile()* functions provide an orthogonal mutual-exclusion lock for each
 28943 **FILE**. The *ftrylockfile()* function provides a non-blocking attempt to acquire a file lock,
 28944 analogous to *pthread_mutex_trylock()*.

28945 These locks behave as if they are the same as those used internally by *stdio* for thread-safety.
 28946 This both provides thread-safety of these functions without requiring a second level of internal

locking and allows functions in *stdio* to be implemented in terms of other *stdio* functions.

Application developers and implementors should be aware that there are potential deadlock problems on **FILE** objects. For example, the line-buffered flushing semantics of *stdio* (requested via `{_IOLBF}`) require that certain input operations sometimes cause the buffered contents of implementation-defined line-buffered output streams to be flushed. If two threads each hold the lock on the other's **FILE**, deadlock ensues. This type of deadlock can be avoided by acquiring **FILE** locks in a consistent order. In particular, the line-buffered output stream deadlock can typically be avoided by acquiring locks on input streams before locks on output streams if a thread would be acquiring both.

In summary, threads sharing *stdio* streams with other threads can use *flockfile()* and *funlockfile()* to cause sequences of I/O performed by a single thread to be kept bundled. The only case where the use of *flockfile()* and *funlockfile()* is required is to provide a scope protecting uses of the **_unlocked* functions/macros. This moves the cost/performance tradeoff to the optimal point.

FUTURE DIRECTIONS

None.

SEE ALSO

[*getc_unlocked\(\)*](#)

XBD [Section 3.287](#) (on page 79), [`<stdio.h>`](#)

CHANGE HISTORY

First released in Issue 5. Included for alignment with the POSIX Threads Extension.

Issue 6

These functions are marked as part of the Thread-Safe Functions option.

Issue 7

The *flockfile()*, *ftrylockfile()*, and *funlockfile()* functions are moved from the Thread-Safe Functions option to the Base.

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0140 [118] is applied.

+

28973 **NAME**

28974 floor, floorf, floorl — floor function

28975 **SYNOPSIS**

```
28976 #include <math.h>
28977 double floor(double x);
28978 float floorf(float x);
28979 long double floorl(long double x);
```

28980 **DESCRIPTION**

28981 CX The functionality described on this reference page is aligned with the ISO C standard. Any
 28982 conflict between the requirements described here and the ISO C standard is unintentional. This
 28983 volume of POSIX.1-2008 defers to the ISO C standard.

28984 These functions shall compute the largest integral value not greater than x . -

28985 **RETURN VALUE**

28986 MX The result shall have the same sign as x . +

28987 Upon successful completion, these functions shall return the largest integral value not greater
 28988 than x , expressed as a **double**, **float**, or **long double**, as appropriate for the return type of the
 28989 function.

28990 MX If x is NaN, a NaN shall be returned.

28991 If x is ± 0 or $\pm \text{Inf}$, x shall be returned. -

28992 **ERRORS**

28993 No errors are defined. |

28994 **EXAMPLES**

28995 None.

28996 **APPLICATION USAGE**

28997 The integral value returned by these functions might not be expressible as an **intmax_t**. The
 28998 return value should be tested before assigning it to an integer type to avoid the undefined
 28999 results of an integer overflow.

29000 These functions may raise the inexact floating-point exception if the result differs in value from |
 29001 the argument.

29002 **RATIONALE**

29003 None.

29004 **FUTURE DIRECTIONS**

29005 None.

29006 **SEE ALSO**

29007 [ceil\(\)](#), [feclearexcept\(\)](#), [fetestexcept\(\)](#), [isnan\(\)](#)

29008 [Section 4.19](#) (on page 116), [<math.h>](#)

29009 **CHANGE HISTORY**

29010 First released in Issue 1. Derived from Issue 1 of the SVID.

29011 **Issue 5**

29012 The DESCRIPTION is updated to indicate how an application should check for an error. This
 29013 text was previously published in the APPLICATION USAGE section.

29014	Issue 6	
29015	The <i>floorf()</i> and <i>floorl()</i> functions are added for alignment with the ISO/IEC 9899:1999 standard.	
29016	The DESCRIPTION, RETURN VALUE, ERRORS, and APPLICATION USAGE sections are	
29017	revised to align with the ISO/IEC 9899:1999 standard.	
29018	IEC 60559:1989 standard floating-point extensions over the ISO/IEC 9899:1999 standard are	
29019	marked.	
29020	Issue 7	
29021	POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0141 [346] is applied.	+

29022 **NAME**29023 `fma, fmaf, fmal` — floating-point multiply-add29024 **SYNOPSIS**

```
29025 #include <math.h>
29026 double fma(double x, double y, double z);
29027 float fmaf(float x, float y, float z);
29028 long double fmal(long double x, long double y, long double z);
```

29029 **DESCRIPTION**

29030 **CX** The functionality described on this reference page is aligned with the ISO C standard. Any
 29031 conflict between the requirements described here and the ISO C standard is unintentional. This
 29032 volume of POSIX.1-2008 defers to the ISO C standard.

29033 These functions shall compute $(x * y) + z$, rounded as one ternary operation: they shall compute
 29034 the value (as if) to infinite precision and round once to the result format, according to the
 29035 rounding mode characterized by the value of `FLT_ROUNDS`.

29036 An application wishing to check for error situations should set `errno` to zero and call
 29037 `feclearexcept(FE_ALL_EXCEPT)` before calling these functions. On return, if `errno` is non-zero or
 29038 `fetestexcept(FE_INVALID | FE_DIVBYZERO | FE_OVERFLOW | FE_UNDERFLOW)` is non-
 29039 zero, an error has occurred.

29040 **RETURN VALUE**

29041 Upon successful completion, these functions shall return $(x * y) + z$, rounded as one ternary
 29042 operation.

29043 **MX** If the result overflows or underflows, a range error may occur. On systems that support the IEC
 29044 60559 Floating-Point option, if the result overflows a range error shall occur.

29045 If x or y are NaN, a NaN shall be returned.

29046 If x multiplied by y is an exact infinity and z is also an infinity but with the opposite sign, a
 29047 domain error shall occur, and either a NaN (if supported), or an implementation-defined value
 29048 shall be returned.

29049 If one of x and y is infinite, the other is zero, and z is not a NaN, a domain error shall occur, and
 29050 either a NaN (if supported), or an implementation-defined value shall be returned.

29051 If one of x and y is infinite, the other is zero, and z is a NaN, a NaN shall be returned and a
 29052 domain error may occur.

29053 If $x*y$ is not $0*\text{Inf}$ nor $\text{Inf}*0$ and z is a NaN, a NaN shall be returned.

29054 **ERRORS**

29055 These functions shall fail if:

29056	MX	Domain Error	The value of $x*y+z$ is invalid, or the value $x*y$ is invalid and z is not a NaN. If the integer expression <code>(math_errhandling & MATH_ERRNO)</code> is non-zero, then <code>errno</code> shall be set to [EDOM]. If the integer expression <code>(math_errhandling & MATH_ERREXCEPT)</code> is non-zero, then the invalid floating-point exception shall be raised.

29061	MX	Range Error	The result overflows. If the integer expression <code>(math_errhandling & MATH_ERRNO)</code> is non-zero, then <code>errno</code> shall be set to [ERANGE]. If the integer expression <code>(math_errhandling & MATH_ERREXCEPT)</code> is non-zero, then the overflow floating-point exception shall be raised.

29066 These functions may fail if:

29067	MX	Domain Error	The value $x*y$ is invalid and z is a NaN.
29068			If the integer expression $(\text{math\_errhandling} \ \& \ \text{MATH\_ERRNO})$ is non-zero,
29069			then <i>errno</i> shall be set to [EDOM]. If the integer expression $(\text{math\_errhandling}$
29070			$\ \& \ \text{MATH\_ERREXCEPT})$ is non-zero, then the invalid floating-point exception
29071			shall be raised.
29072		Range Error	The result underflows.
29073			If the integer expression $(\text{math\_errhandling} \ \& \ \text{MATH\_ERRNO})$ is non-zero,
29074			then <i>errno</i> shall be set to [ERANGE]. If the integer expression
29075			$(\text{math\_errhandling} \ \& \ \text{MATH\_ERREXCEPT})$ is non-zero, then the underflow
29076			floating-point exception shall be raised.
29077		Range Error	The result overflows.
29078			If the integer expression $(\text{math\_errhandling} \ \& \ \text{MATH\_ERRNO})$ is non-zero,
29079			then <i>errno</i> shall be set to [ERANGE]. If the integer expression
29080			$(\text{math\_errhandling} \ \& \ \text{MATH\_ERREXCEPT})$ is non-zero, then the overflow
29081			floating-point exception shall be raised.

29082 EXAMPLES

29083 None.

29084 APPLICATION USAGE

29085 On error, the expressions $(\text{math\_errhandling} \ \& \ \text{MATH\_ERRNO})$ and $(\text{math\_errhandling} \ \& \ \text{MATH\_ERREXCEPT})$ are independent of each other, but at least one of them must be non-zero.

29087 RATIONALE

29088 In many cases, clever use of floating (*fused*) multiply-add leads to much improved code; but its
 29089 unexpected use by the compiler can undermine carefully written code. The FP_CONTRACT
 29090 macro can be used to disallow use of floating multiply-add; and the *fma()* function guarantees
 29091 its use where desired. Many current machines provide hardware floating multiply-add
 29092 instructions; software implementation can be used for others.

29093 FUTURE DIRECTIONS

29094 None.

29095 SEE ALSO

29096 [*feclearexcept\(\)*](#), [*fetestexcept\(\)*](#)

29097 XBD [Section 4.19](#) (on page 116), [**<math.h>**](#)

29098 CHANGE HISTORY

29099 First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

29100 Issue 7

29101 ISO/IEC 9899:1999 standard, Technical Corrigendum 2 #57 (SD5-XSH-ERN-69) is applied,
 29102 adding a “may fail” range error for non-MX systems.

29103 **NAME**

29104 fmax, fmaxf, fmaxl — determine maximum numeric value of two floating-point numbers

29105 **SYNOPSIS**

29106 #include <math.h>

29107 double fmax(double x, double y);

29108 float fmaxf(float x, float y);

29109 long double fmaxl(long double x, long double y);

29110 **DESCRIPTION**

29111 CX The functionality described on this reference page is aligned with the ISO C standard. Any
 29112 conflict between the requirements described here and the ISO C standard is unintentional. This
 29113 volume of POSIX.1-2008 defers to the ISO C standard.

29114 MX These functions shall determine the maximum numeric value of their arguments. NaN
 29115 arguments shall be treated as missing data: if one argument is a NaN and the other numeric,
 29116 then these functions shall choose the numeric value.

29117 **RETURN VALUE**

29118 Upon successful completion, these functions shall return the maximum numeric value of their
 29119 arguments.

29120 MX If just one argument is a NaN, the other argument shall be returned.

29121 If *x* and *y* are NaN, a NaN shall be returned.

29122 **ERRORS**

29123 No errors are defined.

29124 **EXAMPLES**

29125 None.

29126 **APPLICATION USAGE**

29127 None.

29128 **RATIONALE**

29129 None.

29130 **FUTURE DIRECTIONS**

29131 None.

29132 **SEE ALSO**29133 *fdim()*, *fmin()*

29134 XBD <math.h>

29135 **CHANGE HISTORY**

29136 First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

29137 **Issue 7**

29138 Austin Group Interpretation 1003.1-2001 #007 is applied.

29139 **NAME**

29140 fmemopen — open a memory buffer stream

29141 **SYNOPSIS**

```
29142 CX      #include <stdio.h>
29143
29143      FILE *fmemopen(void *restrict buf, size_t size,
29144                     const char *restrict mode);
```

29145 **DESCRIPTION**

29146 The *fmemopen()* function shall associate the buffer given by the *buf* and *size* arguments with a
 29147 stream. The *buf* argument shall be either a null pointer or point to a buffer that is at least *size*
 29148 bytes long.

29149 The *mode* argument points to a string. If the string is one of the following, the stream shall be |
 29150 opened in the indicated mode. Otherwise, the behavior is undefined. |

29151 *r* Open the stream for reading.

29152 *w* Open the stream for writing.

29153 *a* Append; open the stream for writing at the first null byte.

29154 *r+* Open the stream for update (reading and writing).

29155 *w+* Open the stream for update (reading and writing). Truncate the buffer contents.

29156 *a+* Append; open the stream for update (reading and writing); the initial position is at the
 29157 first null byte.

29158 Implementations shall accept all mode strings allowed by *fopen()*, but the use of the character |
 29159 'b' shall produce implementation-defined results, where the resulting **FILE *** need not behave |
 29160 the same as if 'b' were omitted.

29161 If a null pointer is specified as the *buf* argument, *fmemopen()* shall allocate *size* bytes of memory
 29162 as if by a call to *malloc()*. This buffer shall be automatically freed when the stream is closed.
 29163 Because this feature is only useful when the stream is opened for updating (because there is no
 29164 way to get a pointer to the buffer) the *fmemopen()* call may fail if the *mode* argument does not
 29165 include a '+'.

29166 The stream shall maintain a current position in the buffer. This position shall be initially set to |
 29167 either the beginning of the buffer (for *r* and *w* modes) or to the first null byte in the buffer (for *a*
 29168 modes). If no null byte is found in append mode, the initial position shall be set to one byte after |
 29169 the end of the buffer.

29170 If *buf* is a null pointer, the initial position shall always be set to the beginning of the buffer.

29171 The stream shall also maintain the size of the current buffer contents; use of *fseek()* or *fseeko()* on |
 29172 the stream with *SEEK_END* shall seek relative to this size. For modes *r* and *r+* the size shall be |
 29173 set to the value given by the *size* argument. For modes *w* and *w+* the initial size shall be zero and |
 29174 for modes *a* and *a+* the initial size shall be either the position of the first null byte in the buffer or |
 29175 the value of the *size* argument if no null byte is found.

29176 A read operation on the stream shall not advance the current buffer position beyond the current |
 29177 buffer size. Reaching the buffer size in a read operation shall count as “end-of-file”. Null bytes in |
 29178 the buffer shall have no special meaning for reads. The read operation shall start at the current |
 29179 buffer position of the stream.

29180 A write operation shall start either at the current position of the stream (if *mode* has not specified |
 29181 'a' as the first character) or at the current size of the stream (if *mode* had 'a' as the first |

character). If the current position at the end of the write is larger than the current buffer size, the current buffer size shall be set to the current position. A write operation on the stream shall not advance the current buffer size beyond the size given in the *size* argument.

When a stream open for writing is flushed or closed, a null byte shall be written at the current position or at the end of the buffer, depending on the size of the contents. If a stream open for update is flushed or closed and the last write has advanced the current buffer size, a null byte shall be written at the end of the buffer if it fits.

An attempt to seek a memory buffer stream to a negative position or to a position larger than the buffer size given in the *size* argument shall fail.

RETURN VALUE

Upon successful completion, *fmemopen()* shall return a pointer to the object controlling the stream. Otherwise, a null pointer shall be returned, and *errno* shall be set to indicate the error.

ERRORS

The *fmemopen()* function shall fail if:

[EINVAL] The *size* argument specifies a buffer size of zero.

The *fmemopen()* function may fail if:

[EINVAL] The value of the *mode* argument is not valid.

[EINVAL] The *buf* argument is a null pointer and the *mode* argument does not include a '+' character.

[ENOMEM] The *buf* argument is a null pointer and the allocation of a buffer of length *size* has failed.

[EMFILE] {FOPEN_MAX} streams are currently open in the calling process.

EXAMPLES

```
#include <stdio.h>
#include <string.h>

static char buffer[] = "foobar";

int
main (void)
{
    int ch;
    FILE *stream;

    stream = fmemopen(buffer, strlen (buffer), "r");
    if (stream == NULL)
        /* handle error */;

    while ((ch = fgetc(stream)) != EOF)
        printf("Got %c\n", ch);

    fclose(stream);
    return (0);
}
```

This program produces the following output:

```
Got f
Got o
```

29224 Got o
29225 Got b
29226 Got a
29227 Got r

29228 APPLICATION USAGE

29229 None.

29230 RATIONALE

29231 This interface has been introduced to eliminate many of the errors encountered in the
29232 construction of strings, notably overflowing of strings. This interface prevents overflow.

29233 FUTURE DIRECTIONS

29234 A future revision of this standard may mandate specific behavior when the *mode* argument |
29235 includes 'b'.

29236 SEE ALSO

29237 *fdopen()*, *fopen()*, *freopen()*, *fseek()*, *malloc()*, *open_memstream()*

29238 XBD <stdio.h>

29239 CHANGE HISTORY

29240 First released in Issue 7.

29241 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0142 [461], XSH/TC1-2008/0143 [396], +
29242 XSH/TC1-2008/0144 [396], XSH/TC1-2008/0145 [461], XSH/TC1-2008/0146 [461], +
29243 XSH/TC1-2008/0147 [461], XSH/TC1-2008/0148 [461], XSH/TC1-2008/0149 [461], and +
29244 XSH/TC1-2008/0150 [396] are applied.

29245 **NAME**

29246 fmin, fminf, fminl — determine minimum numeric value of two floating-point numbers

29247 **SYNOPSIS**

```
29248 #include <math.h>
29249 double fmin(double x, double y);
29250 float fminf(float x, float y);
29251 long double fminl(long double x, long double y);
```

29252 **DESCRIPTION**

29253 CX The functionality described on this reference page is aligned with the ISO C standard. Any
 29254 conflict between the requirements described here and the ISO C standard is unintentional. This
 29255 volume of POSIX.1-2008 defers to the ISO C standard.

29256 MX These functions shall determine the minimum numeric value of their arguments. NaN
 29257 arguments shall be treated as missing data: if one argument is a NaN and the other numeric,
 29258 then these functions shall choose the numeric value.

29259 **RETURN VALUE**

29260 Upon successful completion, these functions shall return the minimum numeric value of their
 29261 arguments.

29262 MX If just one argument is a NaN, the other argument shall be returned.

29263 If *x* and *y* are NaN, a NaN shall be returned.

29264 **ERRORS**

29265 No errors are defined.

29266 **EXAMPLES**

29267 None.

29268 **APPLICATION USAGE**

29269 None.

29270 **RATIONALE**

29271 None.

29272 **FUTURE DIRECTIONS**

29273 None.

29274 **SEE ALSO**

29275 *fdim()*, *fmax()*

29276 XBD **<math.h>**

29277 **CHANGE HISTORY**

29278 First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

29279 **Issue 7**

29280 Austin Group Interpretation 1003.1-2001 #008 is applied.

29281 **NAME**

29282 fmod, fmodf, fmodl — floating-point remainder value function

29283 **SYNOPSIS**

```
29284 #include <math.h>
29285 double fmod(double x, double y);
29286 float fmodf(float x, float y);
29287 long double fmodl(long double x, long double y);
```

29288 **DESCRIPTION**

29289 CX The functionality described on this reference page is aligned with the ISO C standard. Any
 29290 conflict between the requirements described here and the ISO C standard is unintentional. This
 29291 volume of POSIX.1-2008 defers to the ISO C standard.

29292 These functions shall return the floating-point remainder of the division of x by y .

29293 An application wishing to check for error situations should set *errno* to zero and call
 29294 *feclearexcept*(FE_ALL_EXCEPT) before calling these functions. On return, if *errno* is non-zero or
 29295 *fetestexcept*(FE_INVALID | FE_DIVBYZERO | FE_OVERFLOW | FE_UNDERFLOW) is non-
 29296 zero, an error has occurred.

29297 **RETURN VALUE**

29298 These functions shall return the value $x - i * y$, for some integer i such that, if y is non-zero, the
 29299 result has the same sign as x and magnitude less than the magnitude of y .

29300 MXX If the correct value would cause underflow, and is not representable, a range error may occur, |
 29301 MXX and *fmod*(), *modf*(), and *fmodl*() shall return 0.0, or (if the IEC 60559 Floating-Point option is not |
 29302 supported) an implementation-defined value no greater in magnitude than DBL_MIN, |
 29303 FLT_MIN, and LDBL_MIN, respectively.

29304 MX If x or y is NaN, a NaN shall be returned.

29305 If y is zero, a domain error shall occur, and a NaN shall be returned. |

29306 If x is infinite, a domain error shall occur, and a NaN shall be returned. |

29307 If x is ± 0 and y is not zero, ± 0 shall be returned.

29308 If x is not infinite and y is $\pm \text{Inf}$, x shall be returned.

29309 MXX If the correct value would cause underflow, and is representable, a range error may occur and
 29310 the correct value shall be returned.

29311 **ERRORS**

29312 These functions shall fail if:

29313 MX **Domain Error** The x argument is infinite or y is zero.

29314 If the integer expression (*math_errhandling* & MATH_ERRNO) is non-zero,
 29315 then *errno* shall be set to [EDOM]. If the integer expression (*math_errhandling*
 29316 & MATH_ERREXCEPT) is non-zero, then the invalid floating-point exception
 29317 shall be raised.

29318 These functions may fail if:

29319 **Range Error** The result underflows.

29320 If the integer expression (*math_errhandling* & MATH_ERRNO) is non-zero,
 29321 then *errno* shall be set to [ERANGE]. If the integer expression
 29322 (*math_errhandling* & MATH_ERREXCEPT) is non-zero, then the underflow
 29323 floating-point exception shall be raised.

29324 EXAMPLES

29325 None.

29326 APPLICATION USAGE

29327 On error, the expressions (*math_errhandling* & MATH_ERRNO) and (*math_errhandling* &
29328 MATH_ERREXCEPT) are independent of each other, but at least one of them must be non-zero.

29329 RATIONALE

29330 None.

29331 FUTURE DIRECTIONS

29332 None.

29333 SEE ALSO

29334 *feclearexcept()*, *fetestexcept()*, *isnan()*

29335 Section 4.19 (on page 116), <math.h>

29336 CHANGE HISTORY

29337 First released in Issue 1. Derived from Issue 1 of the SVID.

29338 Issue 5

29339 The DESCRIPTION is updated to indicate how an application should check for an error. This
29340 text was previously published in the APPLICATION USAGE section.

29341 Issue 6

29342 The behavior for when the *y* argument is zero is now defined.

29343 The *fmodf()* and *fmodl()* functions are added for alignment with the ISO/IEC 9899:1999
29344 standard.

29345 The DESCRIPTION, RETURN VALUE, ERRORS, and APPLICATION USAGE sections are
29346 revised to align with the ISO/IEC 9899:1999 standard.

29347 IEC 60559:1989 standard floating-point extensions over the ISO/IEC 9899:1999 standard are
29348 marked.

29349 Issue 7

29350 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0151 [68], XSH/TC1-2008/0152 [320], +
29351 and XSH/TC1-2008/0153 [68] are applied.

NAME

fmtmsg — display a message in the specified format on standard error and/or a system console

SYNOPSIS

```
#include <fmtmsg.h>

int fmtmsg(long classification, const char *label, int severity,
           const char *text, const char *action, const char *tag);
```

DESCRIPTION

The *fmtmsg()* function shall display messages in a specified format instead of the traditional *printf()* function.

Based on a message's classification component, *fmtmsg()* shall write a formatted message either to standard error, to the console, or to both.

A formatted message consists of up to five components as defined below. The component *classification* is not part of a message displayed to the user, but defines the source of the message and directs the display of the formatted message.

classification Contains the sum of identifying values constructed from the constants defined below. Any one identifier from a subclass may be used in combination with a single identifier from a different subclass. Two or more identifiers from the same subclass should not be used together, with the exception of identifiers from the display subclass. (Both display subclass identifiers may be used so that messages can be displayed to both standard error and the system console.)

Major Classifications

Identifies the source of the condition. Identifiers are: MM_HARD (hardware), MM_SOFT (software), and MM_FIRM (firmware).

Message Source Subclassifications

Identifies the type of software in which the problem is detected. Identifiers are: MM_APPL (application), MM_UTIL (utility), and MM_OPSYS (operating system).

Display Subclassifications

Indicates where the message is to be displayed. Identifiers are: MM_PRINT to display the message on the standard error stream, MM_CONSOLE to display the message on the system console. One or both identifiers may be used.

Status Subclassifications

Indicates whether the application can recover from the condition. Identifiers are: MM_RECOVER (recoverable) and MM_NRECOV (non-recoverable).

An additional identifier, MM_NULLMC, indicates that no classification component is supplied for the message.

label Identifies the source of the message. The format is two fields separated by a <colon>. The first field is up to 10 bytes, the second is up to 14 bytes.

severity Indicates the seriousness of the condition. Identifiers for the levels of *severity* are:

29395		MM_HALT	Indicates that the application has encountered a severe fault and is halting. Produces the string "HALT".
29396			
29397		MM_ERROR	Indicates that the application has detected a fault. Produces the string "ERROR".
29398			
29399		MM_WARNING	Indicates a condition that is out of the ordinary, that might be a problem, and should be watched. Produces the string "WARNING".
29400			
29401			
29402		MM_INFO	Provides information about a condition that is not in error. Produces the string "INFO".
29403			
29404		MM_NOSEV	Indicates that no severity level is supplied for the message.
29405	<i>text</i>		Describes the error condition that produced the message. The character string is not limited to a specific size. If the character string is empty, then the text produced is unspecified.
29406			
29407			
29408	<i>action</i>		Describes the first step to be taken in the error-recovery process. The <i>fmtmsg()</i> function precedes the action string with the prefix: "TO FIX:". The <i>action</i> string is not limited to a specific size.
29409			
29410			
29411	<i>tag</i>		An identifier that references on-line documentation for the message. Suggested usage is that <i>tag</i> includes the <i>label</i> and a unique identifying number. A sample <i>tag</i> is "XSI:cat:146".
29412			
29413			

29414 The *MSGVERB* environment variable (for message verbosity) shall determine for *fmtmsg()*
 29415 which message components it is to select when writing messages to standard error. The value of
 29416 *MSGVERB* shall be a <colon>-separated list of optional keywords. Valid keywords are: *label*,
 29417 *severity*, *text*, *action*, and *tag*. If *MSGVERB* contains a keyword for a component and the
 29418 component's value is not the component's null value, *fmtmsg()* shall include that component in
 29419 the message when writing the message to standard error. If *MSGVERB* does not include a
 29420 keyword for a message component, that component shall not be included in the display of the
 29421 message. The keywords may appear in any order. If *MSGVERB* is not defined, if its value is the
 29422 null string, if its value is not of the correct format, or if it contains keywords other than the valid
 29423 ones listed above, *fmtmsg()* shall select all components.

29424 *MSGVERB* shall determine which components are selected for display to standard error. All
 29425 message components shall be included in console messages.

29426 RETURN VALUE

29427 The *fmtmsg()* function shall return one of the following values:

29428	MM_OK	The function succeeded.
29429	MM_NOTOK	The function failed completely.
29430	MM_NOMSG	The function was unable to generate a message on standard error, but otherwise succeeded.
29431		
29432	MM_NOCON	The function was unable to generate a console message, but otherwise succeeded.
29433		

29434 ERRORS

29435 None.

EXAMPLES

1. The following example of *fmtmsg()*:

```
fmtmsg(MM_PRINT, "XSI:cat", MM_ERROR, "illegal option",  
"refer to cat in user's reference manual", "XSI:cat:001")
```

produces a complete message in the specified message format:

```
XSI:cat: ERROR: illegal option  
TO FIX: refer to cat in user's reference manual XSI:cat:001
```

2. When the environment variable *MSGVERB* is set as follows:

```
MSGVERB=severity:text:action
```

and Example 1 is used, *fmtmsg()* produces:

```
ERROR: illegal option  
TO FIX: refer to cat in user's reference manual
```

APPLICATION USAGE

One or more message components may be systematically omitted from messages generated by an application by using the null value of the argument for that component.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

fprintf()

XBD <fmtmsg.h>

CHANGE HISTORY

First released in Issue 4, Version 2.

Issue 5

Moved from X/OPEN UNIX extension to BASE.

NAME

fnmatch — match a filename string or a pathname

SYNOPSIS

```
#include <fnmatch.h>
```

```
int fnmatch(const char *pattern, const char *string, int flags);
```

DESCRIPTION

The *fnmatch()* function shall match patterns as described in XCU [Section 2.13.1](#) (on page 2354) and [Section 2.13.2](#) (on page 2355). It checks the string specified by the *string* argument to see if it matches the pattern specified by the *pattern* argument.

The *flags* argument shall modify the interpretation of *pattern* and *string*. It is the bitwise-inclusive OR of zero or more of the flags defined in **<fnmatch.h>**. If the FNM_PATHNAME flag is set in *flags*, then a <slash> character (' / ') in *string* shall be explicitly matched by a <slash> in *pattern*; it shall not be matched by either the <asterisk> or <question-mark> special characters, nor by a bracket expression. If the FNM_PATHNAME flag is not set, the <slash> character shall be treated as an ordinary character.

If FNM_NOESCAPE is not set in *flags*, a <backslash> character in *pattern* followed by any other character shall match that second character in *string*. In particular, "\\\" shall match a <backslash> in *string*. If FNM_NOESCAPE is set, a <backslash> character shall be treated as an ordinary character.

If FNM_PERIOD is set in *flags*, then a leading <period> (' . ') in *string* shall match a <period> in *pattern*; as described by rule 2 in XCU [Section 2.13.3](#) (on page 2355) where the location of “leading” is indicated by the value of FNM_PATHNAME:

- If FNM_PATHNAME is set, a <period> is “leading” if it is the first character in *string* or if it immediately follows a <slash>.
- If FNM_PATHNAME is not set, a <period> is “leading” only if it is the first character of *string*.

If FNM_PERIOD is not set, then no special restrictions are placed on matching a period.

RETURN VALUE

If *string* matches the pattern specified by *pattern*, then *fnmatch()* shall return 0. If there is no match, *fnmatch()* shall return FNM_NOMATCH, which is defined in **<fnmatch.h>**. If an error occurs, *fnmatch()* shall return another non-zero value.

ERRORS

No errors are defined.

EXAMPLES

None.

APPLICATION USAGE

The *fnmatch()* function has two major uses. It could be used by an application or utility that needs to read a directory and apply a pattern against each entry. The *find* utility is an example of this. It can also be used by the *pax* utility to process its *pattern* operands, or by applications that need to match strings in a similar manner.

The name *fnmatch()* is intended to imply *filename* match, rather than *pathname* match. The default action of this function is to match filename strings, rather than pathnames, since it gives no special significance to the <slash> character. With the FNM_PATHNAME flag, *fnmatch()* does match pathnames, but without tilde expansion, parameter expansion, or special treatment for a <period> at the beginning of a filename.

29507 RATIONALE

29508 This function replaced the REG_FILENAME flag of *regcomp()* in early proposals of this volume
29509 of POSIX.1-2008. It provides virtually the same functionality as the *regcomp()* and *regexexec()*
29510 functions using the REG_FILENAME and REG_FSLASH flags (the REG_FSLASH flag was
29511 proposed for *regcomp()*, and would have had the opposite effect from FNM_PATHNAME), but
29512 with a simpler function and less system overhead.

29513 FUTURE DIRECTIONS

29514 None.

29515 SEE ALSO

29516 *glob()*, [Section 2.6](#)

29517 XBD [<fnmatch.h>](#)

29518 CHANGE HISTORY

29519 First released in Issue 4. Derived from the ISO POSIX-2 standard.

29520 Issue 5

29521 Moved from POSIX2 C-language Binding to BASE.

29522 Issue 7

29523 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0154 [291] and XSH/TC1-2008/0155 +
29524 [291] are applied.

29525 **NAME**29526 `fopen` — open a stream29527 **SYNOPSIS**29528 `#include <stdio.h>`29529 `FILE *fopen(const char *restrict pathname, const char *restrict mode);` |29530 **DESCRIPTION**

29531 CX The functionality described on this reference page is aligned with the ISO C standard. Any
 29532 conflict between the requirements described here and the ISO C standard is unintentional. This
 29533 volume of POSIX.1-2008 defers to the ISO C standard.

29534 The `fopen()` function shall open the file whose pathname is the string pointed to by `pathname`,
 29535 and associates a stream with it.

29536 The `mode` argument points to a string. If the string is one of the following, the file shall be opened
 29537 in the indicated mode. Otherwise, the behavior is undefined.

29538	<code>r</code> or <code>rb</code>	Open file for reading.
29539	<code>w</code> or <code>wb</code>	Truncate to zero length or create file for writing.
29540	<code>a</code> or <code>ab</code>	Append; open or create file for writing at end-of-file.
29541	<code>r+</code> or <code>rb+</code> or <code>r+b</code>	Open file for update (reading and writing).
29542	<code>w+</code> or <code>wb+</code> or <code>w+b</code>	Truncate to zero length or create file for update.
29543	<code>a+</code> or <code>ab+</code> or <code>a+b</code>	Append; open or create file for update, writing at end-of-file.

29544 CX The character '`b`' shall have no effect, but is allowed for ISO C standard conformance. Opening
 29545 a file with read mode (`r` as the first character in the `mode` argument) shall fail if the file does not
 29546 exist or cannot be read.

29547 Opening a file with append mode (`a` as the first character in the `mode` argument) shall cause all
 29548 subsequent writes to the file to be forced to the then current end-of-file, regardless of intervening
 29549 calls to `fseek()`.

29550 When a file is opened with update mode ('+' as the second or third character in the `mode`
 29551 argument), both input and output may be performed on the associated stream. However, the
 29552 application shall ensure that output is not directly followed by input without an intervening call
 29553 to `fflush()` or to a file positioning function (`fseek()`, `fsetpos()`, or `rewind()`), and input is not directly
 29554 followed by output without an intervening call to a file positioning function, unless the input
 29555 operation encounters end-of-file.

29556 When opened, a stream is fully buffered if and only if it can be determined not to refer to an
 29557 interactive device. The error and end-of-file indicators for the stream shall be cleared.

29558 CX If `mode` is `w`, `wb`, `a`, `ab`, `w+`, `wb+`, `w+b`, `a+`, `ab+`, or `a+b`, and the file did not previously exist, upon
 29559 successful completion, `fopen()` shall mark for update the last data access, last data modification,
 29560 and last file status change timestamps of the file and the last file status change and last data
 29561 modification timestamps of the parent directory.

29562 If `mode` is `w`, `wb`, `a`, `ab`, `w+`, `wb+`, `w+b`, `a+`, `ab+`, or `a+b`, and the file did not previously exist, the
 29563 `fopen()` function shall create a file as if it called the `creat()` function with a value appropriate for
 29564 the `path` argument interpreted from `pathname` and a value of `S_IRUSR` | `S_IWUSR` | `S_IRGRP` |
 29565 `S_IWGRP` | `S_IROTH` | `S_IWOTH` for the `mode` argument.

29566 If `mode` is `w`, `wb`, `w+`, `wb+`, or `w+b`, and the file did previously exist, upon successful completion,
 29567 `fopen()` shall mark for update the last data modification and last file status change timestamps of

the file.

After a successful call to the *fopen()* function, the orientation of the stream shall be cleared, the encoding rule shall be cleared, and the associated **mbstate_t** object shall be set to describe an initial conversion state.

The file descriptor associated with the opened stream shall be allocated and opened as if by a call to *open()* with the following flags:

<i>fopen()</i> Mode	<i>open()</i> Flags
<i>r</i> or <i>rb</i>	O_RDONLY
<i>w</i> or <i>wb</i>	O_WRONLY O_CREAT O_TRUNC
<i>a</i> or <i>ab</i>	O_WRONLY O_CREAT O_APPEND
<i>r+</i> or <i>rb+</i> or <i>r+b</i>	O_RDWR
<i>w+</i> or <i>wb+</i> or <i>w+b</i>	O_RDWR O_CREAT O_TRUNC
<i>a+</i> or <i>ab+</i> or <i>a+b</i>	O_RDWR O_CREAT O_APPEND

RETURN VALUE

Upon successful completion, *fopen()* shall return a pointer to the object controlling the stream.

Otherwise, a null pointer shall be returned, and *errno* shall be set to indicate the error.

ERRORS

The *fopen()* function shall fail if:

[EACCES] Search permission is denied on a component of the path prefix, or the file exists and the permissions specified by *mode* are denied, or the file does not exist and write permission is denied for the parent directory of the file to be created.

[EINTR] A signal was caught during *fopen()*.

[EISDIR] The named file is a directory and *mode* requires write access.

[ELOOP] A loop exists in symbolic links encountered during resolution of the *path* argument.

[EMFILE] All file descriptors available to the process are currently open.

[EMFILE] {STREAM_MAX} streams are currently open in the calling process.

[ENAMETOOLONG]

The length of a pathname exceeds {PATH_MAX}, or pathname resolution of a symbolic link produced an intermediate result with a length that exceeds {PATH_MAX}.

[ENFILE] The maximum allowable number of files is currently open in the system.

[ENOENT] The *mode* string begins with 'r' and a component of *pathname* does not name an existing file, or *mode* begins with 'w' or 'a' and a component of the path prefix of *pathname* does not name an existing file, or *pathname* is an empty string.

[ENOENT] or [ENOTDIR]

The *pathname* argument contains at least one non-*<slash>* character and ends with one or more trailing *<slash>* characters. If *pathname* names an existing file, an [ENOENT] error shall not occur.

29609	CX	[ENOSPC]	The directory or file system that would contain the new file cannot be expanded, the file does not exist, and the file was to be created.
29610			
29611	CX	[ENOTDIR]	A component of the path prefix names an existing file that is neither a directory nor a symbolic link to a directory, or the <i>pathname</i> argument contains at least one non- <code><slash></code> character and ends with one or more trailing <code><slash></code> characters and the last <i>pathname</i> component names an existing file that is neither a directory nor a symbolic link to a directory.
29612			
29613			
29614			
29615			
29616	CX	[ENXIO]	The named file is a character special or block special file, and the device associated with this special file does not exist.
29617			
29618	CX	[EOVERFLOW]	The named file is a regular file and the size of the file cannot be represented correctly in an object of type <code>off_t</code> .
29619			
29620	CX	[EROFS]	The named file resides on a read-only file system and <i>mode</i> requires write access.
29621			
29622			The <i>fopen()</i> function may fail if:
29623	CX	[EINVAL]	The value of the <i>mode</i> argument is not valid.
29624	CX	[ELOOP]	More than {SYMLOOP_MAX} symbolic links were encountered during resolution of the <i>path</i> argument.
29625			
29626	CX	[EMFILE]	{FOPEN_MAX} streams are currently open in the calling process.
29627	CX	[ENAMETOOLONG]	
29628			The length of a component of a <i>pathname</i> is longer than {NAME_MAX}.
29629	CX	[ENOMEM]	Insufficient storage space is available.
29630	CX	[ETXTBSY]	The file is a pure procedure (shared text) file that is being executed and <i>mode</i> requires write access.
29631			

29632 EXAMPLES

29633 Opening a File

29634 The following example tries to open the file named **file** for reading. The *fopen()* function returns
 29635 a file pointer that is used in subsequent *fgets()* and *fclose()* calls. If the program cannot open the
 29636 file, it just ignores it.

```

29637 #include <stdio.h>
29638 ...
29639 FILE *fp;
29640 ...
29641 void rgrep(const char *file)
29642 {
29643     ...
29644     if ((fp = fopen(file, "r")) == NULL)
29645         return;
29646     ...
29647 }
```

APPLICATION USAGE

None.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSOSection 2.5 (on page 495), *creat()*, *fclose()*, *fdopen()*, *fmemopen()*, *freopen()*, *open_memstream()*

XBD <stdio.h>

CHANGE HISTORY

First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 5

Large File Summit extensions are added.

Issue 6

Extensions beyond the ISO C standard are marked.

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- In the DESCRIPTION, text is added to indicate setting of the offset maximum in the open file description. This change is to support large files.
- In the ERRORS section, the [Eoverflow] condition is added. This change is to support large files.
- The [ELOOP] mandatory error condition is added.
- The [EINVAL], [EMFILE], [ENAMETOOLONG], [ENOMEM], and [ETXTBSY] optional error conditions are added.

The normative text is updated to avoid use of the term “must” for application requirements.

The following changes are made for alignment with the ISO/IEC 9899:1999 standard:

- The prototype for *fopen()* is updated.
- The DESCRIPTION is updated to note that if the argument *mode* points to a string other than those listed, then the behavior is undefined.

The wording of the mandatory [ELOOP] error condition is updated, and a second optional [ELOOP] error condition is added.

Issue 7

Austin Group Interpretation 1003.1-2001 #025 is applied, clarifying the file creation mode.

Austin Group Interpretation 1003.1-2001 #143 is applied.

Austin Group Interpretation 1003.1-2001 #159 is applied, clarifying requirements for the flags set on the open file description.

SD5-XBD-ERN-4 is applied, changing the definition of the [EMFILE] error.

SD5-XSH-ERN-149 is applied, changing the {STREAM_MAX} [EMFILE] error condition from a “may fail” to a “shall fail”.

Changes are made related to support for finegrained timestamps.

29688
29689

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0156 [291,433], XSH/TC1-2008/0157 + [146,433], XSH/TC1-2008/0158 [324], and XSH/TC1-2008/0159 [14] are applied.

29690 **NAME**

29691 fork — create a new process

29692 **SYNOPSIS**

29693 #include <unistd.h>

29694 pid_t fork(void);

29695 **DESCRIPTION**

29696 The *fork()* function shall create a new process. The new process (child process) shall be an exact
 29697 copy of the calling process (parent process) except as detailed below:

- 29698 • The child process shall have a unique process ID.
- 29699 • The child process ID also shall not match any active process group ID.
- 29700 • The child process shall have a different parent process ID, which shall be the process ID of
 29701 the calling process.
- 29702 • The child process shall have its own copy of the parent's file descriptors. Each of the
 29703 child's file descriptors shall refer to the same open file description with the corresponding
 29704 file descriptor of the parent.
- 29705 • The child process shall have its own copy of the parent's open directory streams. Each
 29706 open directory stream in the child process may share directory stream positioning with the
 29707 corresponding directory stream of the parent.
- 29708 • The child process shall have its own copy of the parent's message catalog descriptors.
- 29709 • The child process values of *tms_utime*, *tms_stime*, *tms_cutime*, and *tms_cstime* shall be set to
 29710 0.
- 29711 • The time left until an alarm clock signal shall be reset to zero, and the alarm, if any, shall be
 29712 canceled; see *alarm()*.
- 29713 XSI • All *semadj* values shall be cleared.
- 29714 • File locks set by the parent process shall not be inherited by the child process.
- 29715 • The set of signals pending for the child process shall be initialized to the empty set.
- 29716 XSI • Interval timers shall be reset in the child process.
- 29717 • Any semaphores that are open in the parent process shall also be open in the child process.
- 29718 ML • The child process shall not inherit any address space memory locks established by the
 29719 parent process via calls to *mlockall()* or *mlock()*.
- 29720 • Memory mappings created in the parent shall be retained in the child process.
 29721 MAP_PRIVATE mappings inherited from the parent shall also be MAP_PRIVATE
 29722 mappings in the child, and any modifications to the data in these mappings made by the
 29723 parent prior to calling *fork()* shall be visible to the child. Any modifications to the data in
 29724 MAP_PRIVATE mappings made by the parent after *fork()* returns shall be visible only to
 29725 the parent. Modifications to the data in MAP_PRIVATE mappings made by the child shall
 29726 be visible only to the child.
- 29727 PS • For the SCHED_FIFO and SCHED_RR scheduling policies, the child process shall inherit
 29728 the policy and priority settings of the parent process during a *fork()* function. For other
 29729 scheduling policies, the policy and priority settings on *fork()* are implementation-defined.

- 29730 • Per-process timers created by the parent shall not be inherited by the child process.
- 29731 MSG • The child process shall have its own copy of the message queue descriptors of the parent.
 29732 Each of the message descriptors of the child shall refer to the same open message queue
 29733 description as the corresponding message descriptor of the parent.
- 29734 • No asynchronous input or asynchronous output operations shall be inherited by the child
 29735 process. Any use of asynchronous control blocks created by the parent produces undefined
 29736 behavior.
- 29737 • A process shall be created with a single thread. If a multi-threaded process calls *fork()*, the
 29738 new process shall contain a replica of the calling thread and its entire address space,
 29739 possibly including the states of mutexes and other resources. Consequently, to avoid
 29740 errors, the child process may only execute async-signal-safe operations until such time as
 29741 one of the *exec* functions is called. Fork handlers may be established by means of the
 29742 *pthread_atfork()* function in order to maintain application invariants across *fork()* calls.
- 29743 When the application calls *fork()* from a signal handler and any of the fork handlers
 29744 registered by *pthread_atfork()* calls a function that is not async-signal-safe, the behavior is
 29745 undefined.
- 29746 OB TRC TRI • If the Trace option and the Trace Inherit option are both supported:
 29747 If the calling process was being traced in a trace stream that had its inheritance policy set
 29748 to *POSIX_TRACE_INHERITED*, the child process shall be traced into that trace stream,
 29749 and the child process shall inherit the parent's mapping of trace event names to trace event
 29750 type identifiers. If the trace stream in which the calling process was being traced had its
 29751 inheritance policy set to *POSIX_TRACE_CLOSE_FOR_CHILD*, the child process shall not
 29752 be traced into that trace stream. The inheritance policy is set by a call to the
 29753 *posix_trace_attr_setinherited()* function.
- 29754 OB TRC • If the Trace option is supported, but the Trace Inherit option is not supported:
 29755 The child process shall not be traced into any of the trace streams of its parent process.
- 29756 OB TRC • If the Trace option is supported, the child process of a trace controller process shall not
 29757 control the trace streams controlled by its parent process.
- 29758 CPT • The initial value of the CPU-time clock of the child process shall be set to zero.
- 29759 TCT • The initial value of the CPU-time clock of the single thread of the child process shall be set
 29760 to zero.
- 29761 All other process characteristics defined by POSIX.1-2008 shall be the same in the parent and
 29762 child processes. The inheritance of process characteristics not defined by POSIX.1-2008 is
 29763 unspecified by POSIX.1-2008.
- 29764 After *fork()*, both the parent and the child processes shall be capable of executing independently
 29765 before either one terminates.
- 29766 **RETURN VALUE**
 29767 Upon successful completion, *fork()* shall return 0 to the child process and shall return the
 29768 process ID of the child process to the parent process. Both processes shall continue to execute
 29769 from the *fork()* function. Otherwise, -1 shall be returned to the parent process, no child process
 29770 shall be created, and *errno* shall be set to indicate the error.

ERRORS

The *fork()* function shall fail if:

[EAGAIN] The system lacked the necessary resources to create another process, or the system-imposed limit on the total number of processes under execution system-wide or by a single user {CHILD_MAX} would be exceeded.

The *fork()* function may fail if:

[ENOMEM] Insufficient storage space is available.

EXAMPLES

None.

APPLICATION USAGE

None.

RATIONALE

Many historical implementations have timing windows where a signal sent to a process group (for example, an interactive SIGINT) just prior to or during execution of *fork()* is delivered to the parent following the *fork()* but not to the child because the *fork()* code clears the child's set of pending signals. This volume of POSIX.1-2008 does not require, or even permit, this behavior. However, it is pragmatic to expect that problems of this nature may continue to exist in implementations that appear to conform to this volume of POSIX.1-2008 and pass available verification suites. This behavior is only a consequence of the implementation failing to make the interval between signal generation and delivery totally invisible. From the application's perspective, a *fork()* call should appear atomic. A signal that is generated prior to the *fork()* should be delivered prior to the *fork()*. A signal sent to the process group after the *fork()* should be delivered to both parent and child. The implementation may actually initialize internal data structures corresponding to the child's set of pending signals to include signals sent to the process group during the *fork()*. Since the *fork()* call can be considered as atomic from the application's perspective, the set would be initialized as empty and such signals would have arrived after the *fork()*; see also <signal.h>.

One approach that has been suggested to address the problem of signal inheritance across *fork()* is to add an [EINTR] error, which would be returned when a signal is detected during the call. While this is preferable to losing signals, it was not considered an optimal solution. Although it is not recommended for this purpose, such an error would be an allowable extension for an implementation.

The [ENOMEM] error value is reserved for those implementations that detect and distinguish such a condition. This condition occurs when an implementation detects that there is not enough memory to create the process. This is intended to be returned when [EAGAIN] is inappropriate because there can never be enough memory (either primary or secondary storage) to perform the operation. Since *fork()* duplicates an existing process, this must be a condition where there is sufficient memory for one such process, but not for two. Many historical implementations actually return [ENOMEM] due to temporary lack of memory, a case that is not generally distinct from [EAGAIN] from the perspective of a conforming application.

Part of the reason for including the optional error [ENOMEM] is because the SVID specifies it and it should be reserved for the error condition specified there. The condition is not applicable on many implementations.

IEEE Std 1003.1-1988 neglected to require concurrent execution of the parent and child of *fork()*. A system that single-threads processes was clearly not intended and is considered an unacceptable "toy implementation" of this volume of POSIX.1-2008. The only objection anticipated to the phrase "executing independently" is testability, but this assertion should be

testable. Such tests require that both the parent and child can block on a detectable action of the other, such as a write to a pipe or a signal. An interactive exchange of such actions should be possible for the system to conform to the intent of this volume of POSIX.1-2008.

The [EAGAIN] error exists to warn applications that such a condition might occur. Whether it occurs or not is not in any practical sense under the control of the application because the condition is usually a consequence of the user's use of the system, not of the application's code. Thus, no application can or should rely upon its occurrence under any circumstances, nor should the exact semantics of what concept of "user" is used be of concern to the application developer. Validation writers should be cognizant of this limitation.

There are two reasons why POSIX programmers call *fork()*. One reason is to create a new thread of control within the same program (which was originally only possible in POSIX by creating a new process); the other is to create a new process running a different program. In the latter case, the call to *fork()* is soon followed by a call to one of the *exec* functions.

The general problem with making *fork()* work in a multi-threaded world is what to do with all of the threads. There are two alternatives. One is to copy all of the threads into the new process. This causes the programmer or implementation to deal with threads that are suspended on system calls or that might be about to execute system calls that should not be executed in the new process. The other alternative is to copy only the thread that calls *fork()*. This creates the difficulty that the state of process-local resources is usually held in process memory. If a thread that is not calling *fork()* holds a resource, that resource is never released in the child process because the thread whose job it is to release the resource does not exist in the child process.

When a programmer is writing a multi-threaded program, the first described use of *fork()*, creating new threads in the same program, is provided by the *pthread_create()* function. The *fork()* function is thus used only to run new programs, and the effects of calling functions that require certain resources between the call to *fork()* and the call to an *exec* function are undefined.

The addition of the *forkall()* function to the standard was considered and rejected. The *forkall()* function lets all the threads in the parent be duplicated in the child. This essentially duplicates the state of the parent in the child. This allows threads in the child to continue processing and allows locks and the state to be preserved without explicit *pthread_atfork()* code. The calling process has to ensure that the threads processing state that is shared between the parent and child (that is, file descriptors or MAP_SHARED memory) behaves properly after *forkall()*. For example, if a thread is reading a file descriptor in the parent when *forkall()* is called, then two threads (one in the parent and one in the child) are reading the file descriptor after the *forkall()*. If this is not desired behavior, the parent process has to synchronize with such threads before calling *forkall()*.

While the *fork()* function is async-signal-safe, there is no way for an implementation to determine whether the fork handlers established by *pthread_atfork()* are async-signal-safe. The fork handlers may attempt to execute portions of the implementation that are not async-signal-safe, such as those that are protected by mutexes, leading to a deadlock condition. It is therefore undefined for the fork handlers to execute functions that are not async-signal-safe when *fork()* is called from a signal handler.

When *forkall()* is called, threads, other than the calling thread, that are in functions that can return with an [EINTR] error may have those functions return [EINTR] if the implementation cannot ensure that the function behaves correctly in the parent and child. In particular, *pthread_cond_wait()* and *pthread_cond_timedwait()* need to return in order to ensure that the condition has not changed. These functions can be awakened by a spurious condition wakeup rather than returning [EINTR].

29865 **FUTURE DIRECTIONS**

29866 None.

29867 **SEE ALSO**29868 *alarm()*, *exec*, *fcntl()*, *posix_trace_attr_getinherited()*, *posix_trace_eventid_equal()*, *pthread_atfork()*,
29869 *semop()*, *signal()*, *times()*29870 XBD [Section 4.11](#) (on page 110), [<sys/types.h>](#), [<unistd.h>](#)29871 **CHANGE HISTORY**

29872 First released in Issue 1. Derived from Issue 1 of the SVID.

29873 **Issue 5**29874 The DESCRIPTION is changed for alignment with the POSIX Realtime Extension and the POSIX
29875 Threads Extension.29876 **Issue 6**29877 The following new requirements on POSIX implementations derive from alignment with the
29878 Single UNIX Specification:

- 29879
- The requirement to include [<sys/types.h>](#) has been removed. Although [<sys/types.h>](#) was
29880 required for conforming implementations of previous POSIX specifications, it was not
29881 required for UNIX applications.

29882 The following changes were made to align with the IEEE P1003.1a draft standard:

- 29883
- The effect of *fork()* on a pending alarm call in the child process is clarified.

29884 The description of CPU-time clock semantics is added for alignment with IEEE Std 1003.1d-1999.

29885 The description of tracing semantics is added for alignment with IEEE Std 1003.1q-2000.

29886 IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/17 is applied, adding text to the
29887 DESCRIPTION and RATIONALE relating to fork handlers registered by the *pthread_atfork()*
29888 function and async-signal safety.29889 **Issue 7**29890 Austin Group Interpretation 1003.1-2001 #080 is applied, clarifying the status of asynchronous
29891 input and asynchronous output operations and asynchronous control lists in the DESCRIPTION.29892 Functionality relating to the Asynchronous Input and Output, Memory Mapped Files, Timers,
29893 and Threads options is moved to the Base.

29894 Functionality relating to message catalog descriptors is moved from the XSI option to the Base.

29895 **NAME**

29896 fpathconf, pathconf — get configurable pathname variables

29897 **SYNOPSIS**

29898 #include <unistd.h>

29899 long fpathconf(int *fildes*, int *name*);29900 long pathconf(const char **path*, int *name*);29901 **DESCRIPTION**29902 The *fpathconf()* and *pathconf()* functions shall determine the current value of a configurable limit
29903 or option (*variable*) that is associated with a file or directory.29904 For *pathconf()*, the *path* argument points to the pathname of a file or directory.29905 For *fpathconf()*, the *fildes* argument is an open file descriptor.29906 The *name* argument represents the variable to be queried relative to that file or directory.
29907 Implementations shall support all of the variables listed in the following table and may support
29908 others. The variables in the following table come from <limits.h> or <unistd.h> and the
29909 symbolic constants, defined in <unistd.h>, are the corresponding values used for *name*.

Variable	Value of <i>name</i>	Requirements
{FILESIZEBITS}	_PC_FILESIZEBITS	4,7
{LINK_MAX}	_PC_LINK_MAX	1
{MAX_CANON}	_PC_MAX_CANON	2
{MAX_INPUT}	_PC_MAX_INPUT	2
{NAME_MAX}	_PC_NAME_MAX	3,4
{PATH_MAX}	_PC_PATH_MAX	4,5
{PIPE_BUF}	_PC_PIPE_BUF	6
{POSIX2_SYMLINKS}	_PC_2_SYMLINKS	4
{POSIX_ALLOC_SIZE_MIN}	_PC_ALLOC_SIZE_MIN	10
{POSIX_REC_INCR_XFER_SIZE}	_PC_REC_INCR_XFER_SIZE	10
{POSIX_REC_MAX_XFER_SIZE}	_PC_REC_MAX_XFER_SIZE	10
{POSIX_REC_MIN_XFER_SIZE}	_PC_REC_MIN_XFER_SIZE	10
{POSIX_REC_XFER_ALIGN}	_PC_REC_XFER_ALIGN	10
{SYMLINK_MAX}	_PC_SYMLINK_MAX	4,9
_POSIX_CHOWN_RESTRICTED	_PC_CHOWN_RESTRICTED	7
_POSIX_NO_TRUNC	_PC_NO_TRUNC	3,4
_POSIX_VDISABLE	_PC_VDISABLE	2
_POSIX_ASYNC_IO	_PC_ASYNC_IO	8
_POSIX_PRIO_IO	_PC_PRIO_IO	8
_POSIX_SYNC_IO	_PC_SYNC_IO	8
_POSIX_TIMESTAMP_RESOLUTION	_PC_TIMESTAMP_RESOLUTION	1

29932 **Requirements**

- 29933 1. If *path* or *fildes* refers to a directory, the value returned shall apply to the directory itself.
- 29934 2. If *path* or *fildes* does not refer to a terminal file, it is unspecified whether an
29935 implementation supports an association of the variable name with the specified file.
- 29936 3. If *path* or *fildes* refers to a directory, the value returned shall apply to filenames within the
29937 directory.

4. If *path* or *fildes* does not refer to a directory, it is unspecified whether an implementation supports an association of the variable name with the specified file.
5. If *path* or *fildes* refers to a directory, the value returned shall be the maximum length of a relative pathname that would not cross any mount points when the specified directory is the working directory.
6. If *path* refers to a FIFO, or *fildes* refers to a pipe or FIFO, the value returned shall apply to the referenced object. If *path* or *fildes* refers to a directory, the value returned shall apply to any FIFO that exists or can be created within the directory. If *path* or *fildes* refers to any other type of file, it is unspecified whether an implementation supports an association of the variable name with the specified file.
7. If *path* or *fildes* refers to a directory, the value returned shall apply to any files, other than directories, that exist or can be created within the directory.
8. If *path* or *fildes* refers to a directory, it is unspecified whether an implementation supports an association of the variable name with the specified file.
9. If *path* or *fildes* refers to a directory, the value returned shall be the maximum length of the string that a symbolic link in that directory can contain.
10. If *path* or *fildes* does not refer to a regular file, it is unspecified whether an implementation supports an association of the variable name with the specified file. If an implementation supports such an association for other than a regular file, the value returned is unspecified.

RETURN VALUE

If *name* is an invalid value, both *pathconf()* and *fpathconf()* shall return `-1` and set *errno* to indicate the error.

If the variable corresponding to *name* is described in `<limits.h>` as a maximum or minimum value and the variable has no limit for the path or file descriptor, both *pathconf()* and *fpathconf()* shall return `-1` without changing *errno*. Note that indefinite limits do not imply infinite limits; see `<limits.h>`.

If the implementation needs to use *path* to determine the value of *name* and the implementation does not support the association of *name* with the file specified by *path*, or if the process did not have appropriate privileges to query the file specified by *path*, or *path* does not exist, *pathconf()* shall return `-1` and set *errno* to indicate the error.

If the implementation needs to use *fildes* to determine the value of *name* and the implementation does not support the association of *name* with the file specified by *fildes*, or if *fildes* is an invalid file descriptor, *fpathconf()* shall return `-1` and set *errno* to indicate the error.

Otherwise, *pathconf()* or *fpathconf()* shall return the current variable value for the file or directory without changing *errno*. The value returned shall not be more restrictive than the corresponding value available to the application when it was compiled with the implementation's `<limits.h>` or `<unistd.h>`.

If the variable corresponding to *name* is dependent on an unsupported option, the results are unspecified.

ERRORS

The *pathconf()* function shall fail if:

[EINVAL] The value of *name* is not valid.

29981	[ELOOP]	A loop exists in symbolic links encountered during resolution of the <i>path</i> argument.
29982		
29983	[EOVERFLOW]	The value of <i>name</i> is <code>_PC_TIMESTAMP_RESOLUTION</code> and the resolution is larger than <code>{LONG_MAX}</code> .
29984		
29985	The <i>fpathconf()</i> function may fail if:	
29986	[EACCES]	Search permission is denied for a component of the path prefix.
29987	[EINVAL]	The implementation does not support an association of the variable <i>name</i> with the specified file.
29988		
29989	[ELOOP]	More than <code>{SYMLOOP_MAX}</code> symbolic links were encountered during resolution of the <i>path</i> argument.
29990		
29991	[ENAMETOOLONG]	
29992		The length of a component of a pathname is longer than <code>{NAME_MAX}</code> .
29993	[ENAMETOOLONG]	
29994		The length of a pathname exceeds <code>{PATH_MAX}</code> , or pathname resolution of a symbolic link produced an intermediate result with a length that exceeds <code>{PATH_MAX}</code> .
29995		
29996		
29997	[ENOENT]	A component of <i>path</i> does not name an existing file or <i>path</i> is an empty string.
29998	[ENOTDIR]	A component of the path prefix names an existing file that is neither a directory nor a symbolic link to a directory, or the <i>path</i> argument contains at least one non- <code><slash></code> character and ends with one or more trailing <code><slash></code> characters and the last pathname component names an existing file that is neither a directory nor a symbolic link to a directory.
29999		
30000		
30001		
30002		
30003	The <i>fpathconf()</i> function shall fail if:	
30004	[EINVAL]	The value of <i>name</i> is not valid.
30005	[EOVERFLOW]	The value of <i>name</i> is <code>_PC_TIMESTAMP_RESOLUTION</code> and the resolution is larger than <code>{LONG_MAX}</code> .
30006		
30007	The <i>fpathconf()</i> function may fail if:	
30008	[EBADF]	The <i>fildev</i> argument is not a valid file descriptor.
30009	[EINVAL]	The implementation does not support an association of the variable <i>name</i> with the specified file.
30010		

EXAMPLES

None.

APPLICATION USAGE

Application developers should check whether an option, such as `_POSIX_ADVISORY_INFO`, is supported prior to obtaining and using values for related variables such as `{POSIX_ALLOC_SIZE_MIN}`.

RATIONALE

The *fpathconf()* function was proposed immediately after the *sysconf()* function when it was realized that some configurable values may differ across file system, directory, or device boundaries.

For example, `{NAME_MAX}` frequently changes between System V and BSD-based file systems; System V uses a maximum of 14, BSD 255. On an implementation that provides both types of file

systems, an application would be forced to limit all pathname components to 14 bytes, as this would be the value specified in `<limits.h>` on such a system.

Therefore, various useful values can be queried on any pathname or file descriptor, assuming that appropriate privileges are in place.

The value returned for the variable `{PATH_MAX}` indicates the longest relative pathname that could be given if the specified directory is the current working directory of the process. A process may not always be able to generate a name that long and use it if a subdirectory in the pathname crosses into a more restrictive file system. Note that implementations are allowed to accept pathnames longer than `{PATH_MAX}` bytes long, but are not allowed to return pathnames longer than this unless the user specifies a larger buffer using a function that provides a buffer size argument.

The value returned for the variable `_POSIX_CHOWN_RESTRICTED` also applies to directories that do not have file systems mounted on them. The value may change when crossing a mount point, so applications that need to know should check for each directory. (An even easier check is to try the `chown()` function and look for an error in case it happens.)

Unlike the values returned by `sysconf()`, the pathname-oriented variables are potentially more volatile and are not guaranteed to remain constant throughout the lifetime of the process. For example, in between two calls to `pathconf()`, the file system in question may have been unmounted and remounted with different characteristics.

Also note that most of the errors are optional. If one of the variables always has the same value on an implementation, the implementation need not look at *path* or *fildev* to return that value and is, therefore, not required to detect any of the errors except the meaning of `[EINVAL]` that indicates that the value of *name* is not valid for that variable.

If the value of any of the limits is unspecified (logically infinite), they will not be defined in `<limits.h>` and the `pathconf()` and `fpathconf()` functions return `-1` without changing *errno*. This can be distinguished from the case of giving an unrecognized *name* argument because *errno* is set to `[EINVAL]` in this case.

Since `-1` is a valid return value for the `pathconf()` and `fpathconf()` functions, applications should set *errno* to zero before calling them and check *errno* only if the return value is `-1`.

For the case of `{SYMLINK_MAX}`, since both `pathconf()` and `open()` follow symbolic links, there is no way that *path* or *fildev* could refer to a symbolic link.

It was the intention of IEEE Std 1003.1d-1999 that the following variables:

```
{POSIX_ALLOC_SIZE_MIN}
{POSIX_REC_INCR_XFER_SIZE}
{POSIX_REC_MAX_XFER_SIZE}
{POSIX_REC_MIN_XFER_SIZE}
{POSIX_REC_XFER_ALIGN}
```

only applied to regular files, but Note 10 also permits implementation of the advisory semantics on other file types unique to an implementation (for example, a character special device).

The `[EOVERFLOW]` error for `_PC_TIMESTAMP_RESOLUTION` cannot occur on POSIX-compliant file systems because POSIX requires a timestamp resolution no larger than one second. Even on 32-bit systems, this can be represented without overflow.

FUTURE DIRECTIONS

None.

SEE ALSO

chown(), *confstr()*, *sysconf()*

XBD [**<limits.h>**](#), [**<unistd.h>**](#)

XCU *getconf*

CHANGE HISTORY

First released in Issue 3. Included for alignment with the POSIX.1-1988 standard.

Issue 5

The DESCRIPTION is updated for alignment with the POSIX Realtime Extension.

Large File Summit extensions are added.

Issue 6

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The DESCRIPTION is updated to include {FILESIZEBITS}.
- The [ELOOP] mandatory error condition is added.
- A second [ENAMETOOLONG] is added as an optional error condition.

The following changes were made to align with the IEEE P1003.1a draft standard:

- The _PC_SYMLINK_MAX entry is added to the table in the DESCRIPTION.

The following *pathconf()* variables and their associated names are added for alignment with IEEE Std 1003.1d-1999:

```
{POSIX_ALLOC_SIZE_MIN}
{POSIX_REC_INCR_XFER_SIZE}
{POSIX_REC_MAX_XFER_SIZE}
{POSIX_REC_MIN_XFER_SIZE}
{POSIX_REC_XFER_ALIGN}
```

IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/18 is applied, changing the fourth paragraph of the DESCRIPTION and removing shading and margin markers from the table. This change is needed since implementations are required to support all of these symbols.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/34 is applied, adding the table entry for POSIX2_SYMLINKS in the DESCRIPTION.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/35 is applied, updating the DESCRIPTION and RATIONALE sections to clarify behavior for the following variables:

```
{POSIX_ALLOC_SIZE_MIN}
{POSIX_REC_INCR_XFER_SIZE}
{POSIX_REC_MAX_XFER_SIZE}
{POSIX_REC_MIN_XFER_SIZE}
{POSIX_REC_XFER_ALIGN}
```

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/36 is applied, updating the RETURN VALUE and APPLICATION USAGE sections to state that the results are unspecified if a variable is dependent on an unsupported option, and advising application developers to check for

- 30106 supported options prior to obtaining and using such values.
- 30107 **Issue 7**
- 30108 Austin Group Interpretations 1003.1-2001 #143 and #160 are applied.
- 30109 Changes are made related to support for finegrained timestamps.
- 30110 The [ENOTDIR] error condition is clarified to cover the condition where the last component of a
30111 pathname exists but is not a directory or a symbolic link to a directory.
- 30112 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0160 [256,428], XSH/TC1-2008/0161 +
30113 [256,428], and XSH/TC1-2008/0162 [324] are applied.

30114 **NAME**

30115 fpclassify — classify real floating type

30116 **SYNOPSIS**

30117 #include <math.h>

30118 int fpclassify(real-floating x);

30119 **DESCRIPTION**

30120 CX The functionality described on this reference page is aligned with the ISO C standard. Any
 30121 conflict between the requirements described here and the ISO C standard is unintentional. This
 30122 volume of POSIX.1-2008 defers to the ISO C standard.

30123 The *fpclassify()* macro shall classify its argument value as NaN, infinite, normal, subnormal,
 30124 zero, or into another implementation-defined category. First, an argument represented in a
 30125 format wider than its semantic type is converted to its semantic type. Then classification is based
 30126 on the type of the argument.

30127 **RETURN VALUE**

30128 The *fpclassify()* macro shall return the value of the number classification macro appropriate to
 30129 the value of its argument.

30130 **ERRORS**

30131 No errors are defined.

30132 **EXAMPLES**

30133 None.

30134 **APPLICATION USAGE**

30135 None.

30136 **RATIONALE**

30137 None.

30138 **FUTURE DIRECTIONS**

30139 None.

30140 **SEE ALSO**30141 *isfinite()*, *isinf()*, *isnan()*, *isnormal()*, *signbit()*

30142 XBD <math.h>

30143 **CHANGE HISTORY**

30144 First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

30145 NAME

30146 `dprintf, fprintf, printf, snprintf, sprintf` — print formatted output

30147 SYNOPSIS

30148 `#include <stdio.h>`

```
30149 CX int dprintf(int filides, const char *restrict format, ...);
30150 int fprintf(FILE *restrict stream, const char *restrict format, ...);
30151 int printf(const char *restrict format, ...);
30152 int snprintf(char *restrict s, size_t n,
30153             const char *restrict format, ...);
30154 int sprintf(char *restrict s, const char *restrict format, ...);
```

30155 DESCRIPTION

30156 CX Excluding *dprintf()*: The functionality described on this reference page is aligned with the ISO C
 30157 standard. Any conflict between the requirements described here and the ISO C standard is
 30158 unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

30159 The *fprintf()* function shall place output on the named output *stream*. The *printf()* function shall
 30160 place output on the standard output stream *stdout*. The *sprintf()* function shall place output
 30161 followed by the null byte, '`\0`', in consecutive bytes starting at *s*; it is the user's responsibility
 30162 to ensure that enough space is available.

30163 CX The *dprintf()* function shall be equivalent to the *fprintf()* function, except that *dprintf()* shall
 30164 write output to the file associated with the file descriptor specified by the *filides* argument rather
 30165 than place output on a stream.

30166 The *snprintf()* function shall be equivalent to *sprintf()*, with the addition of the *n* argument
 30167 which states the size of the buffer referred to by *s*. If *n* is zero, nothing shall be written and *s*
 30168 may be a null pointer. Otherwise, output bytes beyond the *n*-1st shall be discarded instead of
 30169 being written to the array, and a null byte is written at the end of the bytes actually written into
 30170 the array.

30171 If copying takes place between objects that overlap as a result of a call to *sprintf()* or *snprintf()*,
 30172 the results are undefined.

30173 Each of these functions converts, formats, and prints its arguments under control of the *format*.
 30174 The *format* is a character string, beginning and ending in its initial shift state, if any. The *format* is
 30175 composed of zero or more directives: *ordinary characters*, which are simply copied to the output
 30176 stream, and *conversion specifications*, each of which shall result in the fetching of zero or more
 30177 arguments. The results are undefined if there are insufficient arguments for the *format*. If the
 30178 *format* is exhausted while arguments remain, the excess arguments shall be evaluated but are
 30179 otherwise ignored.

30180 CX Conversions can be applied to the *n*th argument after the *format* in the argument list, rather than
 30181 to the next unused argument. In this case, the conversion specifier character `%` (see below) is
 30182 replaced by the sequence "`%n$`", where *n* is a decimal integer in the range `[1, {NL_ARGMAX}]`,
 30183 giving the position of the argument in the argument list. This feature provides for the definition
 30184 of format strings that select arguments in an order appropriate to specific languages (see the
 30185 EXAMPLES section).

30186 The *format* can contain either numbered argument conversion specifications (that is, "`%n$`" and
 30187 "`*m$`"), or unnumbered argument conversion specifications (that is, `%` and `*`), but not both. The
 30188 only exception to this is that `%%` can be mixed with the "`%n$`" form. The results of mixing
 30189 numbered and unnumbered argument specifications in a *format* string are undefined. When
 30190 numbered argument specifications are used, specifying the *N*th argument requires that all the
 30191 leading arguments, from the first to the (*N*-1)th, are specified in the format string.

30192 In format strings containing the "%n\$" form of conversion specification, numbered arguments
 30193 in the argument list can be referenced from the format string as many times as required.

30194 In format strings containing the % form of conversion specification, each conversion specification
 30195 uses the first unused argument in the argument list.

30196 CX All forms of the *fprintf()* functions allow for the insertion of a language-dependent radix
 30197 character in the output string. The radix character is defined in the current locale (category
 30198 LC_NUMERIC). In the POSIX locale, or in a locale where the radix character is not defined, the
 30199 radix character shall default to a <period> ('.').

30200 CX Each conversion specification is introduced by the '%' character or by the character sequence
 30201 "%n\$", after which the following appear in sequence:

- 30202 • Zero or more *flags* (in any order), which modify the meaning of the conversion
 30203 specification.
- 30204 • An optional minimum *field width*. If the converted value has fewer bytes than the field
 30205 width, it shall be padded with <space> characters by default on the left; it shall be padded
 30206 on the right if the left-adjustment flag ('-'), described below, is given to the field width.
 30207 The field width takes the form of an <asterisk> ('*'), described below, or a decimal
 30208 integer.
- 30209 • An optional *precision* that gives the minimum number of digits to appear for the d, i, o, u,
 30210 x, and X conversion specifiers; the number of digits to appear after the radix character for
 30211 the a, A, e, E, f, and F conversion specifiers; the maximum number of significant digits for
 30212 the g and G conversion specifiers; or the maximum number of bytes to be printed from a
 30213 XSI string in the s and S conversion specifiers. The precision takes the form of a <period>
 30214 ('.') followed either by an <asterisk> ('*'), described below, or an optional decimal digit
 30215 string, where a null digit string is treated as zero. If a precision appears with any other
 30216 conversion specifier, the behavior is undefined.
- 30217 • An optional length modifier that specifies the size of the argument.
- 30218 • A *conversion specifier* character that indicates the type of conversion to be applied.

30219 A field width, or precision, or both, may be indicated by an <asterisk> ('*'). In this case an
 30220 argument of type *int* supplies the field width or precision. Applications shall ensure that
 30221 arguments specifying field width, or precision, or both appear in that order before the argument,
 30222 if any, to be converted. A negative field width is taken as a '-' flag followed by a positive field
 30223 CX width. A negative precision is taken as if the precision were omitted. In *format* strings
 30224 containing the "%n\$" form of a conversion specification, a field width or precision may be
 30225 indicated by the sequence "%m\$", where *m* is a decimal integer in the range [1,{NL_ARGMAX}]
 30226 giving the position in the argument list (after the *format* argument) of an integer argument
 30227 containing the field width or precision, for example:

30228 `printf("%1$d:%2$.*3$d:%4$.*3$d\n", hour, min, precision, sec);`

30229 The flag characters and their meanings are:

- 30230 CX ' (The <apostrophe>.) The integer portion of the result of a decimal conversion (%i, %d,
 30231 %u, %f, %F, %g, or %G) shall be formatted with thousands' grouping characters. For
 30232 other conversions the behavior is undefined. The non-monetary grouping character is
 30233 used.
- 30234 - The result of the conversion shall be left-justified within the field. The conversion is
 30235 right-justified if this flag is not specified.

30236 + The result of a signed conversion shall always begin with a sign ('+' or '-'). The
 30237 conversion shall begin with a sign only when a negative value is converted if this flag is
 30238 not specified.

30239 <space> If the first character of a signed conversion is not a sign or if a signed conversion results
 30240 in no characters, a <space> shall be prefixed to the result. This means that if the
 30241 <space> and '+' flags both appear, the <space> flag shall be ignored.

30242 # Specifies that the value is to be converted to an alternative form. For o conversion, it
 30243 increases the precision (if necessary) to force the first digit of the result to be zero. For x
 30244 or X conversion specifiers, a non-zero result shall have 0x (or 0X) prefixed to it. For a, A,
 30245 e, E, f, F, g, and G conversion specifiers, the result shall always contain a radix
 30246 character, even if no digits follow the radix character. Without this flag, a radix
 30247 character appears in the result of these conversions only if a digit follows it. For g and G
 30248 conversion specifiers, trailing zeros shall *not* be removed from the result as they
 30249 normally are. For other conversion specifiers, the behavior is undefined.

30250 0 For d, i, o, u, x, X, a, A, e, E, f, F, g, and G conversion specifiers, leading zeros
 30251 (following any indication of sign or base) are used to pad to the field width rather than
 30252 performing space padding, except when converting an infinity or NaN. If the '0' and
 30253 '-' flags both appear, the '0' flag is ignored. For d, i, o, u, x, and X conversion
 30254 CX specifiers, if a precision is specified, the '0' flag shall be ignored. If the '0' and
 30255 <apostrophe> flags both appear, the grouping characters are inserted before zero
 30256 padding. For other conversions, the behavior is undefined.

30257 The length modifiers and their meanings are:

30258 hh Specifies that a following d, i, o, u, x, or X conversion specifier applies to a **signed char**
 30259 or **unsigned char** argument (the argument will have been promoted according to the
 30260 integer promotions, but its value shall be converted to **signed char** or **unsigned char**
 30261 before printing); or that a following n conversion specifier applies to a pointer to a
 30262 **signed char** argument.

30263 h Specifies that a following d, i, o, u, x, or X conversion specifier applies to a **short** or
 30264 **unsigned short** argument (the argument will have been promoted according to the
 30265 integer promotions, but its value shall be converted to **short** or **unsigned short** before
 30266 printing); or that a following n conversion specifier applies to a pointer to a **short**
 30267 argument.

30268 l (ell) Specifies that a following d, i, o, u, x, or X conversion specifier applies to a **long** or
 30269 **unsigned long** argument; that a following n conversion specifier applies to a pointer to
 30270 a **long** argument; that a following c conversion specifier applies to a **wint_t** argument;
 30271 that a following s conversion specifier applies to a pointer to a **wchar_t** argument; or
 30272 has no effect on a following a, A, e, E, f, F, g, or G conversion specifier.

30273 ll (ell-ell) Specifies that a following d, i, o, u, x, or X conversion specifier applies to a **long long**
 30274 or **unsigned long long** argument; or that a following n conversion specifier applies to a
 30275 pointer to a **long long** argument.

30277 j Specifies that a following d, i, o, u, x, or X conversion specifier applies to an **intmax_t**
 30278 or **uintmax_t** argument; or that a following n conversion specifier applies to a pointer
 30279 to an **intmax_t** argument.

30280 z Specifies that a following d, i, o, u, x, or X conversion specifier applies to a **size_t** or the
 30281 corresponding signed integer type argument; or that a following n conversion specifier
 30282 applies to a pointer to a signed integer type corresponding to a **size_t** argument.

30283	t	Specifies that a following d, i, o, u, x, or X conversion specifier applies to a ptrdiff_t or the corresponding unsigned type argument; or that a following n conversion specifier applies to a pointer to a ptrdiff_t argument.
30284		
30285		
30286	L	Specifies that a following a, A, e, E, f, F, g, or G conversion specifier applies to a long double argument.
30287		
30288		If a length modifier appears with any conversion specifier other than as specified above, the behavior is undefined.
30289		
30290		The conversion specifiers and their meanings are:
30291	d, i	The int argument shall be converted to a signed decimal in the style "[−] dddd". The precision specifies the minimum number of digits to appear; if the value being converted can be represented in fewer digits, it shall be expanded with leading zeros. The default precision is 1. The result of converting zero with an explicit precision of zero shall be no characters.
30292		
30293		
30294		
30295		
30296	o	The unsigned argument shall be converted to unsigned octal format in the style "ddd". The precision specifies the minimum number of digits to appear; if the value being converted can be represented in fewer digits, it shall be expanded with leading zeros. The default precision is 1. The result of converting zero with an explicit precision of zero shall be no characters.
30297		
30298		
30299		
30300		
30301	u	The unsigned argument shall be converted to unsigned decimal format in the style "ddd". The precision specifies the minimum number of digits to appear; if the value being converted can be represented in fewer digits, it shall be expanded with leading zeros. The default precision is 1. The result of converting zero with an explicit precision of zero shall be no characters.
30302		
30303		
30304		
30305		
30306	x	The unsigned argument shall be converted to unsigned hexadecimal format in the style "ddd"; the letters "abcdef" are used. The precision specifies the minimum number of digits to appear; if the value being converted can be represented in fewer digits, it shall be expanded with leading zeros. The default precision is 1. The result of converting zero with an explicit precision of zero shall be no characters.
30307		
30308		
30309		
30310		
30311	X	Equivalent to the x conversion specifier, except that letters "ABCDEF" are used instead of "abcdef".
30312		
30313	f, F	The double argument shall be converted to decimal notation in the style "[−] ddd.ddd", where the number of digits after the radix character is equal to the precision specification. If the precision is missing, it shall be taken as 6; if the precision is explicitly zero and no '#' flag is present, no radix character shall appear. If a radix character appears, at least one digit appears before it. The low-order digit shall be rounded in an implementation-defined manner.
30314		
30315		
30316		
30317		
30318		
30319		A double argument representing an infinity shall be converted in one of the styles "[−]inf" or "[−]infinity"; which style is implementation-defined. A double argument representing a NaN shall be converted in one of the styles "[−]nan(<i>n-char-sequence</i>)" or "[−]nan"; which style, and the meaning of any <i>n-char-sequence</i> , is implementation-defined. The F conversion specifier produces "INF", "INFINITY", or "NAN" instead of "inf", "infinity", or "nan", respectively.
30320		
30321		
30322		
30323		
30324		
30325	e, E	The double argument shall be converted in the style "[−] d.ddde±dd", where there is one digit before the radix character (which is non-zero if the argument is non-zero) and the number of digits after it is equal to the precision; if the precision is missing, it shall be taken as 6; if the precision is zero and no '#' flag is present, no radix character shall
30326		
30327		
30328		

30329		appear. The low-order digit shall be rounded in an implementation-defined manner.
30330		The E conversion specifier shall produce a number with ' E ' instead of ' e '
30331		introducing the exponent. The exponent shall always contain at least two digits. If the
30332		value is zero, the exponent shall be zero.
30333		A double argument representing an infinity or NaN shall be converted in the style of
30334		an f or F conversion specifier.
30335	g, G	The double argument representing a floating-point number shall be converted in the
30336		style f or e (or in the style F or E in the case of a G conversion specifier), depending on
30337		the value converted and the precision. Let P equal the precision if non-zero, 6 if the
30338		precision is omitted, or 1 if the precision is zero. Then, if a conversion with style E
30339		would have an exponent of X :
30340		— If $P > X \geq -4$, the conversion shall be with style f (or F) and precision $P - (X + 1)$.
30341		— Otherwise, the conversion shall be with style e (or E) and precision $P - 1$.
30342		Finally, unless the ' # ' flag is used, any trailing zeros shall be removed from the
30343		fractional portion of the result and the decimal-point character shall be removed if there
30344		is no fractional portion remaining.
30345		A double argument representing an infinity or NaN shall be converted in the style of
30346		an f or F conversion specifier.
30347	a, A	A double argument representing a floating-point number shall be converted in the
30348		style "[-]0x <h>.hhhhp±d", where there is one hexadecimal digit (which shall be non-</h>
30349		zero if the argument is a normalized floating-point number and is otherwise
30350		unspecified) before the decimal-point character and the number of hexadecimal digits
30351		after it is equal to the precision; if the precision is missing and FLT_RADIX is a power
30352		of 2, then the precision shall be sufficient for an exact representation of the value; if the
30353		precision is missing and FLT_RADIX is not a power of 2, then the precision shall be
30354		sufficient to distinguish values of type double , except that trailing zeros may be
30355		omitted; if the precision is zero and the ' # ' flag is not specified, no decimal-point
30356		character shall appear. The letters "abcdef" shall be used for a conversion and the
30357		letters "ABCDEF" for A conversion. The A conversion specifier produces a number with
30358		' X ' and ' P ' instead of ' x ' and ' p '. The exponent shall always contain at least one
30359		digit, and only as many more digits as necessary to represent the decimal exponent of
30360		2. If the value is zero, the exponent shall be zero.
30361		A double argument representing an infinity or NaN shall be converted in the style of
30362		an f or F conversion specifier.
30363	c	The int argument shall be converted to an unsigned char , and the resulting byte shall
30364		be written.
30365		If an l (ell) qualifier is present, the wint_t argument shall be converted as if by an ls
30366		conversion specification with no precision and an argument that points to a two-
30367		element array of type wchar_t , the first element of which contains the wint_t argument
30368		to the ls conversion specification and the second element contains a null wide
30369		character.
30370	s	The argument shall be a pointer to an array of char . Bytes from the array shall be
30371		written up to (but not including) any terminating null byte. If the precision is specified,
30372		no more than that many bytes shall be written. If the precision is not specified or is
30373		greater than the size of the array, the application shall ensure that the array contains a
30374		null byte.

30375			If an <code>l</code> (ell) qualifier is present, the argument shall be a pointer to an array of type wchar_t . Wide characters from the array shall be converted to characters (each as if by a call to the <i>wcrtomb()</i> function, with the conversion state described by an mbstate_t object initialized to zero before the first wide character is converted) up to and including a terminating null wide character. The resulting characters shall be written up to (but not including) the terminating null character (byte). If no precision is specified, the application shall ensure that the array contains a null wide character. If a precision is specified, no more than that many characters (bytes) shall be written (including shift sequences, if any), and the array shall contain a null wide character if, to equal the character sequence length given by the precision, the function would need to access a wide character one past the end of the array. In no case shall a partial character be written.
30376			
30377			
30378			
30379			
30380			
30381			
30382			
30383			
30384			
30385			
30386			
30387		<code>p</code>	The argument shall be a pointer to void . The value of the pointer is converted to a sequence of printable characters, in an implementation-defined manner.
30388			
30389		<code>n</code>	The argument shall be a pointer to an integer into which is written the number of bytes written to the output so far by this call to one of the <i>fprintf()</i> functions. No argument is converted.
30390			
30391			
30392	XSI	C	Equivalent to <code>lc</code> .
30393	XSI	S	Equivalent to <code>ls</code> .
30394		<code>%</code>	Print a <code>'%'</code> character; no argument is converted. The complete conversion specification shall be <code>%%</code> .
30395			
30396			If a conversion specification does not match one of the above forms, the behavior is undefined. If any argument is not the correct type for the corresponding conversion specification, the behavior is undefined.
30397			
30398			
30399			In no case shall a nonexistent or small field width cause truncation of a field; if the result of a conversion is wider than the field width, the field shall be expanded to contain the conversion result. Characters generated by <i>fprintf()</i> and <i>printf()</i> are printed as if <i>fputc()</i> had been called.
30400			
30401			
30402			For the <code>a</code> and <code>A</code> conversion specifiers, if <code>FLT_RADIX</code> is a power of 2, the value shall be correctly rounded to a hexadecimal floating number with the given precision.
30403			
30404			For <code>a</code> and <code>A</code> conversions, if <code>FLT_RADIX</code> is not a power of 2 and the result is not exactly representable in the given precision, the result should be one of the two adjacent numbers in hexadecimal floating style with the given precision, with the extra stipulation that the error should have a correct sign for the current rounding direction.
30405			
30406			
30407			
30408			For the <code>e</code> , <code>E</code> , <code>f</code> , <code>F</code> , <code>g</code> , and <code>G</code> conversion specifiers, if the number of significant decimal digits is at most <code>DECIMAL_DIG</code> , then the result should be correctly rounded. If the number of significant decimal digits is more than <code>DECIMAL_DIG</code> but the source value is exactly representable with <code>DECIMAL_DIG</code> digits, then the result should be an exact representation with trailing zeros. Otherwise, the source value is bounded by two adjacent decimal strings $L < U$, both having <code>DECIMAL_DIG</code> significant digits; the value of the resultant decimal string D should satisfy $L \leq D \leq U$, with the extra stipulation that the error should have a correct sign for the current rounding direction.
30409			
30410			
30411			
30412			
30413			
30414			
30415			
30416	CX		The last data modification and last file status change timestamps of the file shall be marked for update:
30417			
30418			1. Between the call to a successful execution of <i>fprintf()</i> or <i>printf()</i> and the next successful completion of a call to <i>fflush()</i> or <i>fclose()</i> on the same stream or a call to <i>exit()</i> or <i>abort()</i>
30419			

2. Upon successful completion of a call to *dprintf()*

RETURN VALUE

Upon successful completion, the *dprintf()*, *fprintf()*, and *printf()* functions shall return the number of bytes transmitted.

Upon successful completion, the *sprintf()* function shall return the number of bytes written to *s*, excluding the terminating null byte.

Upon successful completion, the *snprintf()* function shall return the number of bytes that would be written to *s* had *n* been sufficiently large excluding the terminating null byte.

If an output error was encountered, these functions shall return a negative value and set *errno* to indicate the error.

If the value of *n* is zero on a call to *snprintf()*, nothing shall be written, the number of bytes that would have been written had *n* been sufficiently large excluding the terminating null shall be returned, and *s* may be a null pointer.

ERRORS

For the conditions under which *dprintf()*, *fprintf()*, and *printf()* fail and may fail, refer to *fputc()* or *fputwc()*.

In addition, all forms of *fprintf()* shall fail if:

[EILSEQ] A wide-character code that does not correspond to a valid character has been detected.

[EOVERFLOW] The value to be returned is greater than {INT_MAX}.

In addition, all forms of *fprintf()* may fail if:

[EINVAL] There are insufficient arguments.

The *dprintf()* function may fail if:

[EBADF] The *fildest* argument is not a valid file descriptor.

The *dprintf()*, *fprintf()*, and *printf()* functions may fail if:

[ENOMEM] Insufficient storage space is available.

The *snprintf()* function shall fail if:

[EOVERFLOW] The value of *n* is greater than {INT_MAX}.

EXAMPLES

Printing Language-Independent Date and Time

The following statement can be used to print date and time using a language-independent format:

```
printf(format, weekday, month, day, hour, min);
```

For American usage, *format* could be a pointer to the following string:

```
"%s, %s %d, %d:%.2d\n"
```

This example would produce the following message:

```
Sunday, July 3, 10:02
```


For German usage, *format* could be a pointer to the following string:

```
"%1$s, %3$d. %2$s, %4$d:%5$.2d\n"
```

This definition of *format* would produce the following message:

```
Sonntag, 3. Juli, 10:02
```

Printing File Information

The following example prints information about the type, permissions, and number of links of a specific file in a directory.

The first two calls to *printf()* use data decoded from a previous *stat()* call. The user-defined *strperm()* function shall return a string similar to the one at the beginning of the output for the following command:

```
ls -l
```

The next call to *printf()* outputs the owner's name if it is found using *getpwuid()*; the *getpwuid()* function shall return a **passwd** structure from which the name of the user is extracted. If the user name is not found, the program instead prints out the numeric value of the user ID.

The next call prints out the group name if it is found using *getgrgid()*; *getgrgid()* is very similar to *getpwuid()* except that it shall return group information based on the group number. Once again, if the group is not found, the program prints the numeric value of the group for the entry.

The final call to *printf()* prints the size of the file.

```
#include <stdio.h>
#include <sys/types.h>
#include <pwd.h>
#include <grp.h>

char *strperm (mode_t);
...
struct stat statbuf;
struct passwd *pwd;
struct group *grp;
...
printf("%10.10s", strperm (statbuf.st_mode));
printf("%4d", statbuf.st_nlink);

if ((pwd = getpwuid(statbuf.st_uid)) != NULL)
    printf(" %-8.8s", pwd->pw_name);
else
    printf(" %-8ld", (long) statbuf.st_uid);

if ((grp = getgrgid(statbuf.st_gid)) != NULL)
    printf(" %-8.8s", grp->gr_name);
else
    printf(" %-8ld", (long) statbuf.st_gid);

printf("%9jd", (intmax_t) statbuf.st_size);
...
```

Printing a Localized Date String

The following example gets a localized date string. The `nl_langinfo()` function shall return the localized date string, which specifies the order and layout of the date. The `strftime()` function takes this information and, using the `tm` structure for values, places the date and time information into *datestring*. The `printf()` function then outputs *datestring* and the name of the entry.

```
#include <stdio.h>
#include <time.h>
#include <langinfo.h>
...
struct dirent *dp;
struct tm *tm;
char datestring[256];
...
strftime(datestring, sizeof(datestring), nl_langinfo (D_T_FMT), tm);
printf(" %s %s\n", datestring, dp->d_name);
...
```

Printing Error Information

The following example uses `fprintf()` to write error information to standard error.

In the first group of calls, the program tries to open the password lock file named **LOCKFILE**. If the file already exists, this is an error, as indicated by the `O_EXCL` flag on the `open()` function. If the call fails, the program assumes that someone else is updating the password file, and the program exits.

The next group of calls saves a new password file as the current password file by creating a link between **LOCKFILE** and the new password file **PASSWDFILE**.

```
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <errno.h>

#define LOCKFILE "/etc/ptmp"
#define PASSWDFILE "/etc/passwd"
...
int pfd;
...
if ((pfd = open(LOCKFILE, O_WRONLY | O_CREAT | O_EXCL,
    S_IRUSR | S_IWUSR | S_IRGRP | S_IROTH)) == -1)
{
    fprintf(stderr, "Cannot open /etc/ptmp. Try again later.\n");
    exit(1);
}
...
if (link(LOCKFILE, PASSWDFILE) == -1) {
```

```

30543         fprintf(stderr, "Link error: %s\n", strerror(errno));
30544         exit(1);
30545     }
30546     ...

```

Printing Usage Information

The following example checks to make sure the program has the necessary arguments, and uses *fprintf()* to print usage information if the expected number of arguments is not present.

```

30550 #include <stdio.h>
30551 #include <stdlib.h>
30552 ...
30553 char *Options = "hdbtl";
30554 ...
30555 if (argc < 2) {
30556     fprintf(stderr, "Usage: %s -%s <file\n", argv[0], Options); exit(1);
30557 }
30558 ...

```

Formatting a Decimal String

The following example prints a key and data pair on *stdout*. Note use of the <asterisk> ('*') in the format string; this ensures the correct number of decimal places for the element based on the number of elements requested.

```

30563 #include <stdio.h>
30564 ...
30565 long i;
30566 char *keystr;
30567 int elementlen, len;
30568 ...
30569 while (len < elementlen) {
30570     ...
30571     printf("%s Element%0*ld\n", keystr, elementlen, i);
30572     ...
30573 }

```

Creating a Pathname

The following example creates a pathname using information from a previous *getpwnam()* | function that returned the password database entry of the user.

```

30577 #include <stdint.h> +
30578 #include <stdio.h> +
30579 #include <stdlib.h> +
30580 #include <string.h> +
30581 #include <sys/types.h>
30582 #include <unistd.h>
30583 ...
30584 char *pathname; |
30585 struct passwd *pw;
30586 size_t len; +

```

```

30587     ...
30588     // digits required for pid_t is number of bits times
30589     // log2(10) = approx 10/33
30590     len = strlen(pw->pw_dir) + 1 + 1+(sizeof(pid_t)*80+32)/33 +
30591         sizeof ".out";
30592     pathname = malloc(len);
30593     if (pathname != NULL)
30594     {
30595         snprintf(pathname, len, "%s/%jd.out", pw->pw_dir,
30596             (intmax_t) getpid());
30597         ...
30598     }

```

Reporting an Event

The following example loops until an event has timed out. The *pause()* function waits forever unless it receives a signal. The *fprintf()* statement should never occur due to the possible return values of *pause()*.

```

30603     #include <stdio.h>
30604     #include <unistd.h>
30605     #include <string.h>
30606     #include <errno.h>
30607     ...
30608     while (!event_complete) {
30609     ...
30610         if (pause() != -1 || errno != EINTR)
30611             fprintf(stderr, "pause: unknown error: %s\n", strerror(errno));
30612     }
30613     ...

```

Printing Monetary Information

The following example uses *strfmon()* to convert a number and store it as a formatted monetary string named *convbuf*. If the first number is printed, the program prints the format and the description; otherwise, it just prints the number.

```

30618     #include <monetary.h>
30619     #include <stdio.h>
30620     ...
30621     struct tblfmt {
30622         char *format;
30623         char *description;
30624     };
30625     struct tblfmt table[] = {
30626         { "%n", "default formatting" },
30627         { "%11n", "right align within an 11 character field" },
30628         { "%#5n", "aligned columns for values up to 99999" },
30629         { "%=*#5n", "specify a fill character" },
30630         { "%=0#5n", "fill characters do not use grouping" },
30631         { "%^#5n", "disable the grouping separator" },
30632         { "%^#5.0n", "round off to whole units" },

```

```

30633         { "%^#5.4n", "increase the precision" },
30634         { "%(#5n", "use an alternative pos/neg style" },
30635         { "%!(#5n", "disable the currency symbol" },
30636     };
30637     ...
30638     float input[3];
30639     int i, j;
30640     char convbuf[100];
30641     ...
30642     strfmon(convbuf, sizeof(convbuf), table[i].format, input[j]);
30643
30644     if (j == 0) {
30645         printf("%s%s%s\n", table[i].format,
30646             convbuf, table[i].description);
30647     }
30648     else {
30649         printf("%s\n", convbuf);
30650     }
30651     ...

```

Printing Wide Characters

The following example prints a series of wide characters. Suppose that "L`@`" expands to three bytes:

```

30654     wchar_t wz [3] = L"@@";           // Zero-terminated
30655     wchar_t wn [3] = L"@@";           // Unterminated
30656
30657     fprintf (stdout, "%ls", wz);        // Outputs 6 bytes
30658     fprintf (stdout, "%ls", wn);        // Undefined because wn has no terminator
30659     fprintf (stdout, "%4ls", wz);       // Outputs 3 bytes
30660     fprintf (stdout, "%4ls", wn);       // Outputs 3 bytes; no terminator needed
30661     fprintf (stdout, "%9ls", wz);       // Outputs 6 bytes
30662     fprintf (stdout, "%9ls", wn);       // Outputs 9 bytes; no terminator needed
30663     fprintf (stdout, "%10ls", wz);      // Outputs 6 bytes
30664     fprintf (stdout, "%10ls", wn);      // Undefined because wn has no terminator

```

In the last line of the example, after processing three characters, nine bytes have been output. The fourth character must then be examined to determine whether it converts to one byte or more. If it converts to more than one byte, the output is only nine bytes. Since there is no fourth character in the array, the behavior is undefined.

APPLICATION USAGE

If the application calling *fprintf()* has any objects of type **wint_t** or **wchar_t**, it must also include the **<wchar.h>** header to have these objects defined.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

Section 2.5 (on page 495), *fputc()*, *fscanf()*, *setlocale()*, *strfmon()*, *wcrtomb()*

XBD Chapter 7 (on page 135), *<stdio.h>*, *<wchar.h>*

CHANGE HISTORY

First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 5

Aligned with ISO/IEC 9899:1990/Amendment 1:1995 (E). Specifically, the `l` (ell) qualifier can now be used with `c` and `s` conversion specifiers.

The *snprintf()* function is new in Issue 5.

Issue 6

Extensions beyond the ISO C standard are marked.

The normative text is updated to avoid use of the term “must” for application requirements.

The following changes are made for alignment with the ISO/IEC 9899:1999 standard:

- The prototypes for *fprintf()*, *printf()*, *snprintf()*, and *sprintf()* are updated, and the XSI shading is removed from *snprintf()*.
- The description of *snprintf()* is aligned with the ISO C standard. Note that this supersedes the *snprintf()* description in The Open Group Base Resolution bwg98-006, which changed the behavior from Issue 5.
- The DESCRIPTION is updated.

The DESCRIPTION is updated to use the terms “conversion specifier” and “conversion specification” consistently.

ISO/IEC 9899:1999 standard, Technical Corrigendum 1 is incorporated.

An example of printing wide characters is added.

Issue 7

Austin Group Interpretation 1003.1-2001 #161 is applied, updating the DESCRIPTION of the `0` flag.

Austin Group Interpretation 1003.1-2001 #170 is applied.

ISO/IEC 9899:1999 standard, Technical Corrigendum 2 #68 (SD5-XSH-ERN-70) is applied, revising the description of `g` and `G`.

SD5-XSH-ERN-174 is applied.

The *dprintf()* function is added from The Open Group Technical Standard, 2006, Extended API Set Part 1.

Functionality relating to the `%n$` form of conversion specification and the `<apostrophe>` flag is moved from the XSI option to the Base.

Changes are made related to support for finegrained timestamps.

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0163 [302], XSH/TC1-2008/0164 [316], XSH/TC1-2008/0165 [316], XSH/TC1-2008/0166 [451,291], and XSH/TC1-2008/0167 [14] are applied.

30713 **NAME**

30714 fputc — put a byte on a stream

30715 **SYNOPSIS**

30716 #include <stdio.h>

30717 int fputc(int *c*, FILE **stream*);30718 **DESCRIPTION**

30719 CX The functionality described on this reference page is aligned with the ISO C standard. Any
 30720 conflict between the requirements described here and the ISO C standard is unintentional. This
 30721 volume of POSIX.1-2008 defers to the ISO C standard.

30722 The *fputc()* function shall write the byte specified by *c* (converted to an **unsigned char**) to the
 30723 output stream pointed to by *stream*, at the position indicated by the associated file-position
 30724 indicator for the stream (if defined), and shall advance the indicator appropriately. If the file
 30725 cannot support positioning requests, or if the stream was opened with append mode, the byte
 30726 shall be appended to the output stream.

30727 CX The last data modification and last file status change timestamps of the file shall be marked for
 30728 update between the successful execution of *fputc()* and the next successful completion of a call
 30729 to *fflush()* or *fclose()* on the same stream or a call to *exit()* or *abort()*.

30730 **RETURN VALUE**

30731 Upon successful completion, *fputc()* shall return the value it has written. Otherwise, it shall
 30732 CX return EOF, the error indicator for the stream shall be set, and *errno* shall be set to indicate the
 30733 error.

30734 **ERRORS**

30735 The *fputc()* function shall fail if either the *stream* is unbuffered or the *stream*'s buffer needs to be
 30736 flushed, and:

30737 CX [EAGAIN] The O_NONBLOCK flag is set for the file descriptor underlying *stream* and
 30738 the thread would be delayed in the write operation.

30739 CX [EBADF] The file descriptor underlying *stream* is not a valid file descriptor open for
 30740 writing.

30741 CX [EFBIG] An attempt was made to write to a file that exceeds the maximum file size.

30742 XSI [EFBIG] An attempt was made to write to a file that exceeds the file size limit of the
 30743 process.

30744 CX [EFBIG] The file is a regular file and an attempt was made to write at or beyond the
 30745 offset maximum.

30746 CX [EINTR] The write operation was terminated due to the receipt of a signal, and no data
 30747 was transferred.

30748 CX [EIO] A physical I/O error has occurred, or the process is a member of a background
 30749 process group attempting to write to its controlling terminal, TOSTOP is set,
 30750 the calling thread is not blocking SIGTTOU, the process is not ignoring
 30751 SIGTTOU, and the process group of the process is orphaned. This error may
 30752 also be returned under implementation-defined conditions.

30753 CX [ENOSPC] There was no free space remaining on the device containing the file.

30754 CX [EPIPE] An attempt is made to write to a pipe or FIFO that is not open for reading by
 30755 any process. A SIGPIPE signal shall also be sent to the thread.

30756 The *fputc()* function may fail if:

30757 CX [ENOMEM] Insufficient storage space is available.

30758 CX [ENXIO] A request was made of a nonexistent device, or the request was outside the
30759 capabilities of the device.

30760 EXAMPLES

30761 None.

30762 APPLICATION USAGE

30763 None.

30764 RATIONALE

30765 None.

30766 FUTURE DIRECTIONS

30767 None.

30768 SEE ALSO

30769 Section 2.5 (on page 495), *error()*, *fopen()*, *getrlimit()*, *putc()*, *puts()*, *setbuf()*, *ulimit()*

30770 XBD <stdio.h>

30771 CHANGE HISTORY

30772 First released in Issue 1. Derived from Issue 1 of the SVID.

30773 Issue 5

30774 Large File Summit extensions are added.

30775 Issue 6

30776 Extensions beyond the ISO C standard are marked.

30777 The following new requirements on POSIX implementations derive from alignment with the
30778 Single UNIX Specification:

- 30779 • The [EIO] and [EFBIG] mandatory error conditions are added.
- 30780 • The [ENOMEM] and [ENXIO] optional error conditions are added.

30781 IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/37 is applied, updating the [EAGAIN]
30782 error in the ERRORS section from “the process would be delayed” to “the thread would be
30783 delayed”.

30784 Issue 7

30785 Changes are made related to support for finegrained timestamps.

30786 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0168 [79] and XSH/TC1-2008/0169 +
30787 [14] are applied.

30788 **NAME**

30789 fputs — put a string on a stream

30790 **SYNOPSIS**

30791 #include <stdio.h>

30792 int fputs(const char *restrict *s*, FILE *restrict *stream*);30793 **DESCRIPTION**

30794 CX The functionality described on this reference page is aligned with the ISO C standard. Any
 30795 conflict between the requirements described here and the ISO C standard is unintentional. This
 30796 volume of POSIX.1-2008 defers to the ISO C standard.

30797 The *fputs()* function shall write the null-terminated string pointed to by *s* to the stream pointed
 30798 to by *stream*. The terminating null byte shall not be written.

30799 CX The last data modification and last file status change timestamps of the file shall be marked for
 30800 update between the successful execution of *fputs()* and the next successful completion of a call
 30801 to *fflush()* or *fclose()* on the same stream or a call to *exit()* or *abort()*.

30802 **RETURN VALUE**

30803 Upon successful completion, *fputs()* shall return a non-negative number. Otherwise, it shall
 30804 CX return EOF, set an error indicator for the stream, and set *errno* to indicate the error.

30805 **ERRORS**30806 Refer to *fputc()*.30807 **EXAMPLES**30808 **Printing to Standard Output**

30809 The following example gets the current time, converts it to a string using *localtime()* and
 30810 *asctime()*, and prints it to standard output using *fputs()*. It then prints the number of minutes to
 30811 an event for which it is waiting.

```
30812 #include <time.h>
30813 #include <stdio.h>
30814 ...
30815 time_t now;
30816 int minutes_to_event;
30817 ...
30818 time(&now);
30819 printf("The time is ");
30820 fputs(asctime(localtime(&now)), stdout);
30821 printf("There are still %d minutes to the event.\n",
30822      minutes_to_event);
30823 ...
```

30824 **APPLICATION USAGE**30825 The *puts()* function appends a <newline> while *fputs()* does not.

30826 This volume of POSIX.1-2008 requires that successful completion simply return a non-negative +
 30827 integer. There are at least three known different implementation conventions for this +
 30828 requirement: +

- 30829 • Return a constant value. +

30830	• Return the last character written.	+
30831	• Return the number of bytes written. Note that this implementation convention cannot be	+
30832	adhered to for strings longer than {INT_MAX} bytes as the value would not be	+
30833	representable in the return type of the function. For backwards-compatibility,	+
30834	implementations can return the number of bytes for strings of up to {INT_MAX} bytes, and	+
30835	return {INT_MAX} for all longer strings.	+
30836	RATIONALE	
30837	The <i>fputs()</i> function is one whose source code was specified in the referenced <i>The C Programming</i>	
30838	<i>Language</i> . In the original edition, the function had no defined return value, yet many practical	
30839	implementations would, as a side-effect, return the value of the last character written as that was	
30840	the value remaining in the accumulator used as a return value. In the second edition of the book,	
30841	either the fixed value 0 or EOF would be returned depending upon the return value of <i>ferror()</i> ;	
30842	however, for compatibility with extant implementations, several implementations would, upon	
30843	success, return a positive value representing the last byte written.	
30844	FUTURE DIRECTIONS	
30845	None.	
30846	SEE ALSO	
30847	Section 2.5 (on page 495), <i>fopen()</i> , <i>putc()</i> , <i>puts()</i>	
30848	XBD <stdio.h>	
30849	CHANGE HISTORY	
30850	First released in Issue 1. Derived from Issue 1 of the SVID.	
30851	Issue 6	
30852	Extensions beyond the ISO C standard are marked.	
30853	The <i>fputs()</i> prototype is updated for alignment with the ISO/IEC 9899:1999 standard.	
30854	Issue 7	
30855	Changes are made related to support for finegrained timestamps.	
30856	POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0170 [174,412], XSH/TC1-2008/0171	+
30857	[412], and XSH/TC1-2008/0172 [14] are applied.	

30858 **NAME**30859 `fputwc` — put a wide-character code on a stream30860 **SYNOPSIS**30861 `#include <stdio.h>`30862 `#include <wchar.h>`30863 `wint_t fputwc(wchar_t wc, FILE *stream);`30864 **DESCRIPTION**

30865 CX The functionality described on this reference page is aligned with the ISO C standard. Any
 30866 conflict between the requirements described here and the ISO C standard is unintentional. This
 30867 volume of POSIX.1-2008 defers to the ISO C standard.

30868 The `fputwc()` function shall write the character corresponding to the wide-character code *wc* to
 30869 the output stream pointed to by *stream*, at the position indicated by the associated file-position
 30870 indicator for the stream (if defined), and advances the indicator appropriately. If the file cannot
 30871 support positioning requests, or if the stream was opened with append mode, the character is
 30872 appended to the output stream. If an error occurs while writing the character, the shift state of
 30873 the output file is left in an undefined state.

30874 CX The last data modification and last file status change timestamps of the file shall be marked for
 30875 update between the successful execution of `fputwc()` and the next successful completion of a call
 30876 to `fflush()` or `fclose()` on the same stream or a call to `exit()` or `abort()`.

30877 The `fputwc()` function shall not change the setting of `errno` if successful.

+

30878 **RETURN VALUE**

30879 Upon successful completion, `fputwc()` shall return *wc*. Otherwise, it shall return WEOF, the error
 30880 CX indicator for the stream shall be set, and `errno` shall be set to indicate the error.

30881 **ERRORS**

30882 The `fputwc()` function shall fail if either the stream is unbuffered or data in the *stream*'s buffer
 30883 needs to be written, and:

30884 CX [EAGAIN] The O_NONBLOCK flag is set for the file descriptor underlying *stream* and
 30885 the thread would be delayed in the write operation.

30886 CX [EBADF] The file descriptor underlying *stream* is not a valid file descriptor open for
 30887 writing.

30888 CX [EFBIG] An attempt was made to write to a file that exceeds the maximum file size or
 30889 the file size limit of the process.

30890 CX [EFBIG] The file is a regular file and an attempt was made to write at or beyond the
 30891 offset maximum associated with the corresponding stream.

30892 [EILSEQ] The wide-character code *wc* does not correspond to a valid character.

30893 CX [EINTR] The write operation was terminated due to the receipt of a signal, and no data
 30894 was transferred.

30895 CX [EIO] A physical I/O error has occurred, or the process is a member of a background
 30896 process group attempting to write to its controlling terminal, TOSTOP is set,
 30897 the calling thread is not blocking SIGTTOU, the process is not ignoring
 30898 SIGTTOU, and the process group of the process is orphaned. This error may
 30899 also be returned under implementation-defined conditions.

30900 CX [ENOSPC] There was no free space remaining on the device containing the file.

30901 CX [EPIPE] An attempt is made to write to a pipe or FIFO that is not open for reading by
30902 any process. A SIGPIPE signal shall also be sent to the thread.

30903 The *fputc()* function may fail if:

30904 CX [ENOMEM] Insufficient storage space is available.

30905 CX [ENXIO] A request was made of a nonexistent device, or the request was outside the
30906 capabilities of the device.

30907 EXAMPLES

30908 None.

30909 APPLICATION USAGE

30910 None.

30911 RATIONALE

30912 None.

30913 FUTURE DIRECTIONS

30914 None.

30915 SEE ALSO

30916 Section 2.5 (on page 495), *ferror()*, *fopen()*, *setbuf()*, *ulimit()*

30917 XBD <stdio.h>, <wchar.h>

30918 CHANGE HISTORY

30919 First released in Issue 4. Derived from the MSE working draft.

30920 Issue 5

30921 Aligned with ISO/IEC 9899:1990/Amendment 1:1995 (E). Specifically, the type of argument *wc*
30922 is changed from **wint_t** to **wchar_t**.

30923 The Optional Header (OH) marking is removed from <stdio.h>.

30924 Large File Summit extensions are added.

30925 Issue 6

30926 Extensions beyond the ISO C standard are marked.

30927 The following new requirements on POSIX implementations derive from alignment with the
30928 Single UNIX Specification:

- 30929 • The [EFBIG] and [EIO] mandatory error conditions are added.
- 30930 • The [ENOMEM] and [ENXIO] optional error conditions are added.

30931 IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/38 is applied, updating the [EAGAIN]
30932 error in the ERRORS section from “the process would be delayed” to “the thread would be
30933 delayed”.

30934 Issue 7

30935 Changes are made related to support for finegrained timestamps.

30936 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0173 [105], XSH/TC1-2008/0174 [79], +
30937 and XSH/TC1-2008/0175 [14] are applied.

30938 **NAME**30939 `fputws` — put a wide-character string on a stream30940 **SYNOPSIS**30941 `#include <stdio.h>`30942 `#include <wchar.h>`30943 `int fputws(const wchar_t *restrict ws, FILE *restrict stream);`30944 **DESCRIPTION**

30945 CX The functionality described on this reference page is aligned with the ISO C standard. Any
 30946 conflict between the requirements described here and the ISO C standard is unintentional. This
 30947 volume of POSIX.1-2008 defers to the ISO C standard.

30948 The `fputws()` function shall write a character string corresponding to the (null-terminated) wide-
 30949 character string pointed to by `ws` to the stream pointed to by `stream`. No character corresponding
 30950 to the terminating null wide-character code shall be written.

30951 CX The last data modification and last file status change timestamps of the file shall be marked for
 30952 update between the successful execution of `fputws()` and the next successful completion of a call
 30953 to `fflush()` or `fclose()` on the same stream or a call to `exit()` or `abort()`.

30954 **RETURN VALUE**

30955 Upon successful completion, `fputws()` shall return a non-negative number. Otherwise, it shall
 30956 CX return `-1`, set an error indicator for the stream, and set `errno` to indicate the error.

30957 **ERRORS**30958 Refer to `fputwc()`.30959 **EXAMPLES**

30960 None.

30961 **APPLICATION USAGE**30962 The `fputws()` function does not append a `<newline>`.

30963 This volume of POSIX.1-2008 requires that successful completion simply return a non-negative +
 30964 integer. There are at least three known different implementation conventions for this +
 30965 requirement: +

- 30966 • Return a constant value. +
- 30967 • Return the last character written. +
- 30968 • Return the number of bytes written. Note that this implementation convention cannot be +
 30969 adhered to for strings longer than `{INT_MAX}` bytes as the value would not be +
 30970 representable in the return type of the function. For backwards-compatibility, +
 30971 implementations can return the number of bytes for strings of up to `{INT_MAX}` bytes, and +
 30972 return `{INT_MAX}` for all longer strings. +

30973 **RATIONALE**

30974 None.

30975 **FUTURE DIRECTIONS**

30976 None.

30977 **SEE ALSO**30978 [Section 2.5](#) (on page 495), `fopen()`30979 XBD `<stdio.h>`, `<wchar.h>`

30980 **CHANGE HISTORY**

30981 First released in Issue 4. Derived from the MSE working draft.

30982 **Issue 5**30983 The Optional Header (OH) marking is removed from `<stdio.h>`.30984 **Issue 6**

30985 Extensions beyond the ISO C standard are marked.

30986 The `fputws()` prototype is updated for alignment with the ISO/IEC 9899:1999 standard.30987 **Issue 7**

30988 Changes are made related to support for finegrained timestamps.

30989 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0176 [412] and XSH/TC1-2008/0177 +
30990 [14] are applied.

30991 NAME

30992 fread — binary input

30993 SYNOPSIS

```
30994 #include <stdio.h>
30995 size_t fread(void *restrict ptr, size_t size, size_t nitems,
30996 FILE *restrict stream);
```

30997 DESCRIPTION

30998 CX The functionality described on this reference page is aligned with the ISO C standard. Any
 30999 conflict between the requirements described here and the ISO C standard is unintentional. This
 31000 volume of POSIX.1-2008 defers to the ISO C standard.

31001 The *fread()* function shall read into the array pointed to by *ptr* up to *nitems* elements whose size
 31002 is specified by *size* in bytes, from the stream pointed to by *stream*. For each object, *size* calls shall
 31003 be made to the *fgetc()* function and the results stored, in the order read, in an array of **unsigned**
 31004 **char** exactly overlaying the object. The file position indicator for the stream (if defined) shall be
 31005 advanced by the number of bytes successfully read. If an error occurs, the resulting value of the
 31006 file position indicator for the stream is unspecified. If a partial element is read, its value is
 31007 unspecified.

31008 CX The *fread()* function may mark the last data access timestamp of the file associated with *stream*
 31009 for update. The last data access timestamp shall be marked for update by the first successful
 31010 execution of *fgetc()*, *fgets()*, *fread()*, *fscanf()*, *getc()*, *getchar()*, *getdelim()*, *getline()*, *gets()*, or
 31011 *scanf()* using *stream* that returns data not supplied by a prior call to *ungetc()*.

31012 RETURN VALUE

31013 Upon successful completion, *fread()* shall return the number of elements successfully read which
 31014 is less than *nitems* only if a read error or end-of-file is encountered. If *size* or *nitems* is 0, *fread()*
 31015 shall return 0 and the contents of the array and the state of the stream remain unchanged.
 31016 CX Otherwise, if a read error occurs, the error indicator for the stream shall be set, and *errno* shall
 31017 be set to indicate the error.

31018 ERRORS

31019 Refer to *fgetc()*.

31020 EXAMPLES

31021 Reading from a Stream

31022 The following example reads a single element from the *fp* stream into the array pointed to by
 31023 *buf*.

```
31024 #include <stdio.h>
31025 ...
31026 size_t elements_read;
31027 char buf[100];
31028 FILE *fp;
31029 ...
31030 elements_read = fread(buf, sizeof(buf), 1, fp);
31031 ...
```

31032 If a read error occurs, *elements_read* will be zero but the number of bytes read from the stream
 31033 could be anything from zero to *sizeof(buf)*-1.

31034 The following example reads multiple single-byte elements from the *fp* stream into the array
 31035 pointed to by *buf*.

31036	#include <stdio.h>	+
31037	...	+
31038	size_t bytes_read;	
31039	char buf[100];	
31040	FILE *fp;	
31041	...	
31042	bytes_read = fread(buf, 1, sizeof(buf), fp);	
31043	...	
31044	If a read error occurs, <i>bytes_read</i> will contain the number of bytes read from the stream.	+
31045	APPLICATION USAGE	
31046	The <i>ferror()</i> or <i>feof()</i> functions must be used to distinguish between an error condition and an	
31047	end-of-file condition.	
31048	Because of possible differences in element length and byte ordering, files written using <i>fwrite()</i>	
31049	are application-dependent, and possibly cannot be read using <i>fread()</i> by a different application	
31050	or by the same application on a different processor.	
31051	RATIONALE	
31052	None.	
31053	FUTURE DIRECTIONS	
31054	None.	
31055	SEE ALSO	
31056	Section 2.5 (on page 495), feof() , ferror() , fgetc() , fopen() , fscanf() , getc() , gets()	
31057	XBD <stdio.h>	
31058	CHANGE HISTORY	
31059	First released in Issue 1. Derived from Issue 1 of the SVID.	
31060	Issue 6	
31061	Extensions beyond the ISO C standard are marked.	
31062	The following changes are made for alignment with the ISO/IEC 9899:1999 standard:	
31063	• The <i>fread()</i> prototype is updated.	
31064	• The DESCRIPTION is updated to describe how the bytes from a call to <i>fgetc()</i> are stored.	
31065	Issue 7	
31066	Changes are made related to support for finegrained timestamps.	
31067	POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0178 [232] and XSH/TC1-2008/0179	+
31068	[14] are applied.	

31069 **NAME**

31070 free — free allocated memory

31071 **SYNOPSIS**

31072 #include <stdlib.h>

31073 void free(void *ptr);

31074 **DESCRIPTION**

31075 CX The functionality described on this reference page is aligned with the ISO C standard. Any
 31076 conflict between the requirements described here and the ISO C standard is unintentional. This
 31077 volume of POSIX.1-2008 defers to the ISO C standard.

31078 The *free()* function shall cause the space pointed to by *ptr* to be deallocated; that is, made
 31079 available for further allocation. If *ptr* is a null pointer, no action shall occur. Otherwise, if the
 31080 argument does not match a pointer earlier returned by a function in POSIX.1-2008 that allocates
 31081 memory as if by *malloc()*, or if the space has been deallocated by a call to *free()* or *realloc()*, the
 31082 behavior is undefined.

31083 Any use of a pointer that refers to freed space results in undefined behavior.

31084 **RETURN VALUE**31085 The *free()* function shall not return a value.31086 **ERRORS**

31087 No errors are defined.

31088 **EXAMPLES**

31089 None.

31090 **APPLICATION USAGE**

31091 There is now no requirement for the implementation to support the inclusion of <malloc.h>.

31092 **RATIONALE**

31093 None.

31094 **FUTURE DIRECTIONS**

31095 None.

31096 **SEE ALSO**31097 *calloc()*, *malloc()*, *posix_memalign()*, *realloc()*

31098 XBD <stdlib.h>

31099 **CHANGE HISTORY**

31100 First released in Issue 1. Derived from Issue 1 of the SVID.

31101 **Issue 6**31102 Reference to the *valloc()* function is removed.31103 **Issue 7**

31104 The DESCRIPTION is updated to clarify that if the pointer returned is not by a function that
 31105 allocates memory as if by *malloc()*, then the behavior is undefined.

31106 **NAME**

31107 freeaddrinfo, getaddrinfo — get address information

31108 **SYNOPSIS**

```

31109 #include <sys/socket.h>
31110 #include <netdb.h>

31111 void freeaddrinfo(struct addrinfo *ai);
31112 int getaddrinfo(const char *restrict nodename,
31113                const char *restrict servname,
31114                const struct addrinfo *restrict hints,
31115                struct addrinfo **restrict res);

```

31116 **DESCRIPTION**

31117 The *freeaddrinfo()* function shall free one or more **addrinfo** structures returned by *getaddrinfo()*,
 31118 along with any additional storage associated with those structures. If the *ai_next* field of the
 31119 structure is not null, the entire list of structures shall be freed. The *freeaddrinfo()* function shall
 31120 support the freeing of arbitrary sublists of an **addrinfo** list originally returned by *getaddrinfo()*.

31121 The *getaddrinfo()* function shall translate the name of a service location (for example, a host
 31122 name) and/or a service name and shall return a set of socket addresses and associated
 31123 information to be used in creating a socket with which to address the specified service.

31124 **Note:** In many cases it is implemented by the Domain Name System, as documented in RFC 1034,
 31125 RFC 1035, and RFC 1886.

31126 The *freeaddrinfo()* and *getaddrinfo()* functions shall be thread-safe.

31127 The *nodename* and *servname* arguments are either null pointers or pointers to null-terminated
 31128 strings. One or both of these two arguments shall be supplied by the application as a non-null
 31129 pointer.

31130 The format of a valid name depends on the address family or families. If a specific family is not
 31131 given and the name could be interpreted as valid within multiple supported families, the
 31132 implementation shall attempt to resolve the name in all supported families and, in absence of
 31133 errors, one or more results shall be returned.

31134 If the *nodename* argument is not null, it can be a descriptive name or can be an address string. If
 31135 IP6 the specified address family is AF_INET, AF_INET6, or AF_UNSPEC, valid descriptive names
 31136 include host names. If the specified address family is AF_INET or AF_UNSPEC, address strings
 31137 using Internet standard dot notation as specified in *inet_addr()* are valid.

31138 IP6 If the specified address family is AF_INET6 or AF_UNSPEC, standard IPv6 text forms described
 31139 in *inet_ntop()* are valid.

31140 If *nodename* is not null, the requested service location is named by *nodename*; otherwise, the
 31141 requested service location is local to the caller.

31142 If *servname* is null, the call shall return network-level addresses for the specified *nodename*. If
 31143 *servname* is not null, it is a null-terminated character string identifying the requested service.
 31144 This can be either a descriptive name or a numeric representation suitable for use with the
 31145 IP6 address family or families. If the specified address family is AF_INET, AF_INET6, or
 31146 AF_UNSPEC, the service can be specified as a string specifying a decimal port number.

31147 If the *hints* argument is not null, it refers to a structure containing input values that directs the
 31148 operation by providing options and by limiting the returned information to a specific socket
 31149 type, address family, and/or protocol, as described below. In this *hints* structure every member
 31150 other than *ai_flags*, *ai_family*, *ai_socktype*, and *ai_protocol* shall be set to zero or a null pointer. A
 31151 value of AF_UNSPEC for *ai_family* means that the caller shall accept any address family. A value

of zero for *ai_socktype* means that the caller shall accept any socket type. A value of zero for *ai_protocol* means that the caller shall accept any protocol. If *hints* is a null pointer, the behavior shall be as if it referred to a structure containing the value zero for the *ai_flags*, *ai_socktype*, and *ai_protocol* fields, and AF_UNSPEC for the *ai_family* field.

The *ai_flags* field to which the *hints* parameter points shall be set to zero or be the bitwise-inclusive OR of one or more of the values AI_PASSIVE, AI_CANONNAME, AI_NUMERICHOST, AI_NUMERICSERV, AI_V4MAPPED, AI_ALL, and AI_ADDRCONFIG.

If the AI_PASSIVE flag is specified, the returned address information shall be suitable for use in binding a socket for accepting incoming connections for the specified service. In this case, if the *nodename* argument is null, then the IP address portion of the socket address structure shall be set to INADDR_ANY for an IPv4 address or IN6ADDR_ANY_INIT for an IPv6 address. If the AI_PASSIVE flag is not specified, the returned address information shall be suitable for a call to *connect()* (for a connection-mode protocol) or for a call to *connect()*, *sendto()*, or *sendmsg()* (for a connectionless protocol). In this case, if the *nodename* argument is null, then the IP address portion of the socket address structure shall be set to the loopback address. The AI_PASSIVE flag shall be ignored if the *nodename* argument is not null.

If the AI_CANONNAME flag is specified and the *nodename* argument is not null, the function shall attempt to determine the canonical name corresponding to *nodename* (for example, if *nodename* is an alias or shorthand notation for a complete name).

Note: Since different implementations use different conceptual models, the terms “canonical name” and “alias” cannot be precisely defined for the general case. However, Domain Name System implementations are expected to interpret them as they are used in RFC 1034.

A numeric host address string is not a “name”, and thus does not have a “canonical name” form; no address to host name translation is performed. See below for handling of the case where a canonical name cannot be obtained.

If the AI_NUMERICHOST flag is specified, then a non-null *nodename* string supplied shall be a numeric host address string. Otherwise, an [EAI_NONAME] error is returned. This flag shall prevent any type of name resolution service (for example, the DNS) from being invoked.

If the AI_NUMERICSERV flag is specified, then a non-null *servname* string supplied shall be a numeric port string. Otherwise, an [EAI_NONAME] error shall be returned. This flag shall prevent any type of name resolution service (for example, NIS+) from being invoked.

If the AI_V4MAPPED flag is specified along with an *ai_family* of AF_INET6, then *getaddrinfo()* shall return IPv4-mapped IPv6 addresses on finding no matching IPv6 addresses (*ai_addrlen* shall be 16). The AI_V4MAPPED flag shall be ignored unless *ai_family* equals AF_INET6. If the AI_ALL flag is used with the AI_V4MAPPED flag, then *getaddrinfo()* shall return all matching IPv6 and IPv4 addresses. The AI_ALL flag without the AI_V4MAPPED flag is ignored.

If the AI_ADDRCONFIG flag is specified, IPv4 addresses shall be returned only if an IPv4 address is configured on the local system, and IPv6 addresses shall be returned only if an IPv6 address is configured on the local system.

The *ai_socktype* field to which argument *hints* points specifies the socket type for the service, as defined in *socket()*. If a specific socket type is not given (for example, a value of zero) and the service name could be interpreted as valid with multiple supported socket types, the implementation shall attempt to resolve the service name for all supported socket types and, in the absence of errors, all possible results shall be returned. A non-zero socket type value shall limit the returned information to values with the specified socket type.

If the *ai_family* field to which *hints* points has the value AF_UNSPEC, addresses shall be returned for use with any address family that can be used with the specified *nodename* and/or

servname. Otherwise, addresses shall be returned for use only with the specified address family. If *ai_family* is not AF_UNSPEC and *ai_protocol* is not zero, then addresses shall be returned for use only with the specified address family and protocol; the value of *ai_protocol* shall be interpreted as in a call to the *socket()* function with the corresponding values of *ai_family* and *ai_protocol*.

RETURN VALUE

A zero return value for *getaddrinfo()* indicates successful completion; a non-zero return value indicates failure. The possible values for the failures are listed in the ERRORS section.

Upon successful return of *getaddrinfo()*, the location to which *res* points shall refer to a linked list of **addrinfo** structures, each of which shall specify a socket address and information for use in creating a socket with which to use that socket address. The list shall include at least one **addrinfo** structure. The *ai_next* field of each structure contains a pointer to the next structure on the list, or a null pointer if it is the last structure on the list. Each structure on the list shall include values for use with a call to the *socket()* function, and a socket address for use with the *connect()* function or, if the AI_PASSIVE flag was specified, for use with the *bind()* function. The fields *ai_family*, *ai_socktype*, and *ai_protocol* shall be usable as the arguments to the *socket()* function to create a socket suitable for use with the returned address. The fields *ai_addr* and *ai_addrlen* are usable as the arguments to the *connect()* or *bind()* functions with such a socket, according to the AI_PASSIVE flag.

If *nodename* is not null, and if requested by the AI_CANONNAME flag, the *ai_canonname* field of the first returned **addrinfo** structure shall point to a null-terminated string containing the canonical name corresponding to the input *nodename*; if the canonical name is not available, then *ai_canonname* shall refer to the *nodename* argument or a string with the same contents. The contents of the *ai_flags* field of the returned structures are undefined.

All fields in socket address structures returned by *getaddrinfo()* that are not filled in through an explicit argument (for example, *sin6_flowinfo*) shall be set to zero.

Note: This makes it easier to compare socket address structures.

ERRORS

The *getaddrinfo()* function shall fail and return the corresponding error value if:

[EAI_AGAIN] The name could not be resolved at this time. Future attempts may succeed.

[EAI_BADFLAGS]

The *flags* parameter had an invalid value.

[EAI_FAIL] A non-recoverable error occurred when attempting to resolve the name.

[EAI_FAMILY] The address family was not recognized.

[EAI_MEMORY] There was a memory allocation failure when trying to allocate storage for the return value.

[EAI_NONAME] The name does not resolve for the supplied parameters.

Neither *nodename* nor *servname* were supplied. At least one of these shall be supplied.

[EAI_SERVICE] The service passed was not recognized for the specified socket type.

[EAI_SOCKTYPE]

The intended socket type was not recognized.

31241 [EAI_SYSTEM] A system error occurred; the error code can be found in *errno*.

31242 EXAMPLES

31243 The following (incomplete) program demonstrates the use of *getaddrinfo()* to obtain the socket
31244 address structure(s) for the service named in the program's command-line argument. The
31245 program then loops through each of the address structures attempting to create and bind a
31246 socket to the address, until it performs a successful *bind()*.

```

31247 #include <stdio.h>
31248 #include <stdlib.h>
31249 #include <unistd.h>
31250 #include <string.h>
31251 #include <sys/socket.h>
31252 #include <netdb.h>

31253 int
31254 main(int argc, char *argv[])
31255 {
31256     struct addrinfo *result, *rp;
31257     int sfd, s;

31258     if (argc != 2) {
31259         fprintf(stderr, "Usage: %s port\n", argv[0]);
31260         exit(EXIT_FAILURE);
31261     }

31262     struct addrinfo hints = {};
31263     hints.ai_family = AF_UNSPEC;
31264     hints.ai_socktype = SOCK_DGRAM;
31265     hints.ai_flags = AI_PASSIVE;
31266     hints.ai_protocol = 0;

31267     s = getaddrinfo(NULL, argv[1], &hints, &result);
31268     if (s != 0) {
31269         fprintf(stderr, "getaddrinfo: %s\n", gai_strerror(s));
31270         exit(EXIT_FAILURE);
31271     }

31272     /* getaddrinfo() returns a list of address structures.
31273        Try each address until a successful bind().
31274        If socket(2) (or bind(2)) fails, close the socket
31275        and try the next address. */

31276     for (rp = result; rp != NULL; rp = rp->ai_next) {
31277         sfd = socket(rp->ai_family, rp->ai_socktype,
31278                     rp->ai_protocol);
31279         if (sfd == -1)
31280             continue;

31281         if (bind(sfd, rp->ai_addr, rp->ai_addrlen) == 0)
31282             break; /* Success */

31283         close(sfd);
31284     }

31285     if (rp == NULL) { /* No address succeeded */
31286         fprintf(stderr, "Could not bind\n");

```

```

31287         exit(EXIT_FAILURE);
31288     }
31289     freeaddrinfo(result);    /* No longer needed */
31290     /* ... use socket bound to sfd ... */
31291 }

```

APPLICATION USAGE

If the caller handles only TCP and not UDP, for example, then the *ai_protocol* member of the *hints* structure should be set to IPPROTO_TCP when *getaddrinfo()* is called.

If the caller handles only IPv4 and not IPv6, then the *ai_family* member of the *hints* structure should be set to AF_INET when *getaddrinfo()* is called.

The term “canonical name” is misleading; it is taken from the Domain Name System (RFC 2181). It should be noted that the canonical name is a result of alias processing, and not necessarily a unique attribute of a host, address, or set of addresses. See RFC 2181 for more discussion of this in the Domain Name System context.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

connect(), *endservent()*, *gai_strerror()*, *getnameinfo()*, *socket()*

XBD <netdb.h>, <sys/socket.h>

CHANGE HISTORY

First released in Issue 6. Derived from the XNS, Issue 5.2 specification.

The **restrict** keyword is added to the *getaddrinfo()* prototype for alignment with the ISO/IEC 9899:1999 standard.

IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/19 is applied, adding three notes to the DESCRIPTION and adding text to the APPLICATION USAGE related to the term “canonical name”. A reference to RFC 2181 is also added to the Informative References.

IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/20 is applied, making changes for alignment with IPv6. These include the following:

- Adding AI_V4MAPPED, AI_ALL, and AI_ADDRCONFIG to the allowed values for the *ai_flags* field
- Adding a description of AI_ADDRCONFIG
- Adding a description of the consequences of ignoring the AI_PASSIVE flag.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/39 is applied, changing “corresponding value” to “corresponding error value” in the ERRORS section.

Issue 7

Austin Group Interpretation 1003.1-2001 #013 is applied.

Austin Group Interpretation 1003.1-2001 #146 is applied, updating the DESCRIPTION.

An example is added.

31327 **NAME**

31328 freelocale — free resources allocated for a locale object

31329 **SYNOPSIS**

```
31330 CX      #include <locale.h>
31331          void freelocale(locale_t locobj);
```

31332 **DESCRIPTION**

31333 The *freelocale()* function shall cause the resources allocated for a locale object returned by a call
 31334 to the *newlocale()* or *duplocale()* functions to be released.

31335 The behavior is undefined if the *locobj* argument is the special locale object `LC_GLOBAL_LOCALE` or is not a valid locale object handle. +
 31336 +

31337 Any use of a locale object that has been freed results in undefined behavior.

31338 **RETURN VALUE**

31339 None.

31340 **ERRORS**

31341 None.

31342 **EXAMPLES**31343 **Freeing Up a Locale Object**31344 The following example shows a code fragment to free a locale object created by *newlocale()*:

```
31345      #include <locale.h>
31346      ...
31347      /* Every locale object allocated with newlocale() should be
31348       * freed using freelocale():
31349       */
31350      locale_t loc;
31351      /* Get the locale. */
31352      loc = newlocale (LC_CTYPE_MASK | LC_TIME_MASK, "locname", NULL);
31353      /* ... Use the locale object ... */
31354      ...
31355      /* Free the locale object resources. */
31356      freelocale (loc);
```

31357 **APPLICATION USAGE**

31358 None.

31359 **RATIONALE**

31360 None.

31361 **FUTURE DIRECTIONS**

31362 None.

31363 **SEE ALSO**31364 *duplocale()*, *newlocale()*, *uselocale()*31365 XBD [<locale.h>](#)

31366	CHANGE HISTORY	
31367	First released in Issue 7.	
31368	POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0180 [283] is applied.	+

31369 **NAME**

31370 freopen — open a stream

31371 **SYNOPSIS**

31372 #include <stdio.h>

```
31373 FILE *freopen(const char *restrict pathname, const char *restrict mode, |
31374 FILE *restrict stream);
```

31375 **DESCRIPTION**

31376 CX The functionality described on this reference page is aligned with the ISO C standard. Any
 31377 conflict between the requirements described here and the ISO C standard is unintentional. This
 31378 volume of POSIX.1-2008 defers to the ISO C standard.

31379 The *freopen()* function shall first attempt to flush the stream associated with *stream* as if by a call
 31380 to *fflush(stream)*. Failure to flush the stream successfully shall be ignored. If *pathname* is not a
 31381 null pointer, *freopen()* shall close any file descriptor associated with *stream*. Failure to close the
 31382 file descriptor successfully shall be ignored. The error and end-of-file indicators for the stream
 31383 shall be cleared.

31384 The *freopen()* function shall open the file whose pathname is the string pointed to by *pathname*
 31385 and associate the stream pointed to by *stream* with it. The *mode* argument shall be used just as in
 31386 *fopen()*.

31387 The original stream shall be closed regardless of whether the subsequent open succeeds.

31388 If *pathname* is a null pointer, the *freopen()* function shall attempt to change the mode of the
 31389 stream to that specified by *mode*, as if the name of the file currently associated with the stream
 31390 had been used. In this case, the file descriptor associated with the stream need not be closed if
 31391 the call to *freopen()* succeeds. It is implementation-defined which changes of mode are permitted
 31392 (if any), and under what circumstances.

31393 XSI After a successful call to the *freopen()* function, the orientation of the stream shall be cleared, the
 31394 encoding rule shall be cleared, and the associated **mbstate_t** object shall be set to describe an
 31395 initial conversion state.

31396 CX If *pathname* is not a null pointer, or if *pathname* is a null pointer and the specified mode change
 31397 necessitates the file descriptor associated with the stream to be closed and reopened, the file
 31398 descriptor associated with the reopened stream shall be allocated and opened as if by a call to
 31399 *open()* with the following flags:

<i>freopen()</i> Mode	<i>open()</i> Flags
<i>r</i> or <i>rb</i>	O_RDONLY
<i>w</i> or <i>wb</i>	O_WRONLY O_CREAT O_TRUNC
<i>a</i> or <i>ab</i>	O_WRONLY O_CREAT O_APPEND
<i>r+</i> or <i>rb+</i> or <i>r+b</i>	O_RDWR
<i>w+</i> or <i>wb+</i> or <i>w+b</i>	O_RDWR O_CREAT O_TRUNC
<i>a+</i> or <i>ab+</i> or <i>a+b</i>	O_RDWR O_CREAT O_APPEND

31407 **RETURN VALUE**

31408 Upon successful completion, *freopen()* shall return the value of *stream*. Otherwise, a null pointer
 31409 CX shall be returned, and *errno* shall be set to indicate the error.

31410 **ERRORS**31411 The *freopen()* function shall fail if:

31412 CX [EACCES] Search permission is denied on a component of the path prefix, or the file
 31413 exists and the permissions specified by *mode* are denied, or the file does not
 31414 exist and write permission is denied for the parent directory of the file to be
 31415 created.

31416 CX [EBADF] The file descriptor underlying the stream is not a valid file descriptor when
 31417 *pathname* is a null pointer.

31418 CX [EINTR] A signal was caught during *freopen()*.

31419 CX [EISDIR] The named file is a directory and *mode* requires write access.

31420 CX [ELOOP] A loop exists in symbolic links encountered during resolution of the *path*
 31421 argument.

31422 CX [EMFILE] All file descriptors available to the process are currently open.

31423 CX [ENAMETOOLONG]

31424 The length of a component of a pathname is longer than {NAME_MAX}.

31425 CX [ENFILE] The maximum allowable number of files is currently open in the system.

31426 CX [ENOENT] The *mode* string begins with 'r' and a component of *pathname* does not name
 31427 an existing file, or *mode* begins with 'w' or 'a' and a component of the path
 31428 prefix of *pathname* does not name an existing file, or *pathname* is an empty
 31429 string.

31430 CX [ENOENT] or [ENOTDIR]

31431 The *pathname* argument contains at least one non-*<slash>* character and ends
 31432 with one or more trailing *<slash>* characters. If *pathname* names an existing
 31433 file, an [ENOENT] error shall not occur. +

31434 CX [ENOSPC] The directory or file system that would contain the new file cannot be
 31435 expanded, the file does not exist, and it was to be created.

31436 CX [ENOTDIR] A component of the path prefix names an existing file that is neither a
 31437 directory nor a symbolic link to a directory, or the *pathname* argument contains
 31438 at least one non-*<slash>* character and ends with one or more trailing *<slash>*
 31439 characters and the last pathname component names an existing file that is
 31440 neither a directory nor a symbolic link to a directory. |

31441 CX [ENXIO] The named file is a character special or block special file, and the device
 31442 associated with this special file does not exist.

31443 CX [EOVERFLOW] The named file is a regular file and the size of the file cannot be represented
 31444 correctly in an object of type *off_t*.

31445 CX [EROFS] The named file resides on a read-only file system and *mode* requires write
 31446 access.

31447 The *freopen()* function may fail if:

31448 CX [EBADF] The mode with which the file descriptor underlying the stream was opened
 31449 does not support the requested mode when *pathname* is a null pointer.

31450 CX [EINVAL] The value of the *mode* argument is not valid.

31451	CX	[ELOOP]	More than {SYMLOOP_MAX} symbolic links were encountered during resolution of the <i>path</i> argument.
31452			
31453	CX	[ENAMETOOLONG]	
31454			The length of a pathname exceeds {PATH_MAX}, or pathname resolution of a symbolic link produced an intermediate result with a length that exceeds {PATH_MAX}.
31455			
31456			
31457	CX	[ENOMEM]	Insufficient storage space is available.
31458	CX	[ENXIO]	A request was made of a nonexistent device, or the request was outside the capabilities of the device.
31459			
31460	CX	[ETXTBSY]	The file is a pure procedure (shared text) file that is being executed and <i>mode</i> requires write access.
31461			

31462 EXAMPLES

31463 Directing Standard Output to a File

31464 The following example logs all standard output to the */tmp/logfile* file.

```
31465 #include <stdio.h>
31466 ...
31467 FILE *fp;
31468 ...
31469 fp = freopen ("/tmp/logfile", "a+", stdout);
31470 ...
```

31471 APPLICATION USAGE

31472 The *freopen()* function is typically used to attach the pre-opened *streams* associated with *stdin*,
31473 *stdout*, and *stderr* to other files.

31474 Since implementations are not required to support any stream mode changes when the *pathname*
31475 argument is NULL, portable applications cannot rely on the use of *freopen()* to change the stream
31476 mode, and use of this feature is discouraged. The feature was originally added to the ISO C
31477 standard in order to facilitate changing *stdin* and *stdout* to binary mode. Since a 'b' character in
31478 the mode has no effect on POSIX systems, this use of the feature is unnecessary in POSIX
31479 applications. However, even though the 'b' is ignored, a successful call to *freopen*(NULL, "wb",
31480 *stdout*) does have an effect. In particular, for regular files it truncates the file and sets the file-
31481 position indicator for the stream to the start of the file. It is possible that these side-effects are an
31482 unintended consequence of the way the feature is specified in the ISO/IEC 9899:1999 standard,
31483 but unless or until the ISO C standard is changed, applications which successfully call
31484 *freopen*(NULL, "wb", *stdout*) will behave in unexpected ways on conforming systems in situations
31485 such as:

```
31486 { appl file1; appl file2; } > file3
```

31487 which will result in *file3* containing only the output from the second invocation of *appl*.

31488 RATIONALE

31489 None.

31490 FUTURE DIRECTIONS

31491 None.

SEE ALSO

Section 2.5 (on page 495), *fclose()*, *fdopen()*, *fflush()*, *fmemopen()*, *fopen()*, *mbsinit()*, *open()*, *open_memstream()*

XBD <stdio.h>

CHANGE HISTORY

First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 5

The DESCRIPTION is updated to indicate that the orientation of the stream is cleared and the conversion state of the stream is set to an initial conversion state by a successful call to the *freopen()* function.

Large File Summit extensions are added.

Issue 6

Extensions beyond the ISO C standard are marked.

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- In the DESCRIPTION, text is added to indicate setting of the offset maximum in the open file description. This change is to support large files.
- In the ERRORS section, the [Eoverflow] condition is added. This change is to support large files.
- The [ELOOP] mandatory error condition is added.
- A second [ENAMETOOLONG] is added as an optional error condition.
- The [EINVAL], [ENOMEM], [ENXIO], and [ETXTBSY] optional error conditions are added.

The following changes are made for alignment with the ISO/IEC 9899:1999 standard:

- The *freopen()* prototype is updated.
- The DESCRIPTION is updated.

The wording of the mandatory [ELOOP] error condition is updated, and a second optional [ELOOP] error condition is added.

The DESCRIPTION is updated regarding failure to close, changing the “file” to “file descriptor”.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/40 is applied, adding the following sentence to the DESCRIPTION: “In this case, the file descriptor associated with the stream need not be closed if the call to *freopen()* succeeds.”.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/41 is applied, adding an mandatory [EBADF] error, and an optional [EBADF] error to the ERRORS section.

Issue 7

Austin Group Interpretation 1003.1-2001 #043 is applied, clarifying that the *freopen()* function allocates a file descriptor as per *open()*.

Austin Group Interpretation 1003.1-2001 #143 is applied.

Austin Group Interpretation 1003.1-2001 #159 is applied, clarifying requirements for the flags set on the open file description.

SD5-XBD-ERN-4 is applied, changing the definition of the [EMFILE] error.

31532	SD5-XSH-ERN-150 and SD5-XSH-ERN-219 are applied.
31533	POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0181 [291,433], XSH/TC1-2008/0182 +
31534	[146,433], XSH/TC1-2008/0183 [324], and XSH/TC1-2008/0184 [14] are applied.

31535 **NAME**

31536 frexp, frexpf, frexpl — extract mantissa and exponent from a double precision number

31537 **SYNOPSIS**

```
31538 #include <math.h>
31539 double frexp(double num, int *exp);
31540 float frexpf(float num, int *exp);
31541 long double frexpl(long double num, int *exp);
```

31542 **DESCRIPTION**

31543 CX The functionality described on this reference page is aligned with the ISO C standard. Any
 31544 conflict between the requirements described here and the ISO C standard is unintentional. This
 31545 volume of POSIX.1-2008 defers to the ISO C standard.

31546 These functions shall break a floating-point number *num* into a normalized fraction and an
 31547 integral power of 2. The integer exponent shall be stored in the **int** object pointed to by *exp*.

31548 **RETURN VALUE**

31549 For finite arguments, these functions shall return the value *x*, such that *x* has a magnitude in the
 31550 interval $[\frac{1}{2}, 1)$ or 0, and *num* equals *x* times 2 raised to the power **exp*.

31551 MX If *num* is NaN, a NaN shall be returned, and the value of **exp* is unspecified.

31552 If *num* is ± 0 , ± 0 shall be returned, and the value of **exp* shall be 0.

31553 If *num* is $\pm \text{Inf}$, *num* shall be returned, and the value of **exp* is unspecified.

31554 **ERRORS**

31555 No errors are defined.

31556 **EXAMPLES**

31557 None.

31558 **APPLICATION USAGE**

31559 None.

31560 **RATIONALE**

31561 None.

31562 **FUTURE DIRECTIONS**

31563 None.

31564 **SEE ALSO**

31565 *isnan()*, *ldexp()*, *modf()*

31566 XBD **<math.h>**

31567 **CHANGE HISTORY**

31568 First released in Issue 1. Derived from Issue 1 of the SVID.

31569 **Issue 5**

31570 The DESCRIPTION is updated to indicate how an application should check for an error. This
 31571 text was previously published in the APPLICATION USAGE section.

31572 **Issue 6**

31573 The *frexpf()* and *frexpl()* functions are added for alignment with the ISO/IEC 9899:1999
31574 standard.

31575 The DESCRIPTION, RETURN VALUE, ERRORS, and APPLICATION USAGE sections are
31576 revised to align with the ISO/IEC 9899:1999 standard.

31577 IEC 60559:1989 standard floating-point extensions over the ISO/IEC 9899:1999 standard are
31578 marked.

31579 **NAME**

31580 fscanf, scanf, sscanf — convert formatted input

31581 **SYNOPSIS**

31582 #include <stdio.h>

31583 int fscanf(FILE *restrict *stream*, const char *restrict *format*, ...);31584 int scanf(const char *restrict *format*, ...);31585 int sscanf(const char *restrict *s*, const char *restrict *format*, ...);31586 **DESCRIPTION**

31587 CX The functionality described on this reference page is aligned with the ISO C standard. Any
 31588 conflict between the requirements described here and the ISO C standard is unintentional. This
 31589 volume of POSIX.1-2008 defers to the ISO C standard.

31590 The *fscanf()* function shall read from the named input *stream*. The *scanf()* function shall read
 31591 from the standard input stream *stdin*. The *sscanf()* function shall read from the string *s*. Each
 31592 function reads bytes, interprets them according to a format, and stores the results in its
 31593 arguments. Each expects, as arguments, a control string *format* described below, and a set of
 31594 *pointer* arguments indicating where the converted input should be stored. The result is
 31595 undefined if there are insufficient arguments for the format. If the format is exhausted while
 31596 arguments remain, the excess arguments shall be evaluated but otherwise ignored.

31597 CX Conversions can be applied to the *n*th argument after the *format* in the argument list, rather than
 31598 to the next unused argument. In this case, the conversion specifier character % (see below) is
 31599 replaced by the sequence "%*n*\$", where *n* is a decimal integer in the range [1,{NL_ARGMAX}].
 31600 This feature provides for the definition of format strings that select arguments in an order
 31601 appropriate to specific languages. In format strings containing the "%*n*\$" form of conversion
 31602 specifications, it is unspecified whether numbered arguments in the argument list can be
 31603 referenced from the format string more than once.

31604 The *format* can contain either form of a conversion specification—that is, % or "%*n*\$"—but the
 31605 two forms cannot be mixed within a single *format* string. The only exception to this is that %% or
 31606 %* can be mixed with the "%*n*\$" form. When numbered argument specifications are used,
 31607 specifying the *N*th argument requires that all the leading arguments, from the first to the
 31608 (*N*–1)th, are pointers.

31609 The *fscanf()* function in all its forms shall allow detection of a language-dependent radix
 31610 character in the input string. The radix character is defined in the current locale (category
 31611 LC_NUMERIC). In the POSIX locale, or in a locale where the radix character is not defined, the
 31612 radix character shall default to a <period> ('.').

31613 The format is a character string, beginning and ending in its initial shift state, if any, composed
 31614 of zero or more directives. Each directive is composed of one of the following: one or more
 31615 white-space characters (<space>, <tab>, <newline>, <vertical-tab>, or <form-feed>); an ordinary
 31616 character (neither '%' nor a white-space character); or a conversion specification. Each
 31617 CX conversion specification is introduced by the character '%' or the character sequence "%*n*\$",
 31618 after which the following appear in sequence:

- 31619 • An optional assignment-suppressing character '*'.
- 31620 • An optional non-zero decimal integer that specifies the maximum field width.
- 31621 CX • An optional assignment-allocation character 'm'.
- 31622 • An option length modifier that specifies the size of the receiving object.

- A *conversion specifier* character that specifies the type of conversion to be applied. The valid conversion specifiers are described below.

The *fscanf()* functions shall execute each directive of the format in turn. If a directive fails, as detailed below, the function shall return. Failures are described as input failures (due to the unavailability of input bytes) or matching failures (due to inappropriate input).

A directive composed of one or more white-space characters shall be executed by reading input until no more valid input can be read, or up to the first byte which is not a white-space character, which remains unread.

A directive that is an ordinary character shall be executed as follows: the next byte shall be read from the input and compared with the byte that comprises the directive; if the comparison shows that they are not equivalent, the directive shall fail, and the differing and subsequent bytes shall remain unread. Similarly, if end-of-file, an encoding error, or a read error prevents a character from being read, the directive shall fail.

A directive that is a conversion specification defines a set of matching input sequences, as described below for each conversion character. A conversion specification shall be executed in the following steps.

Input white-space characters (as specified by *isspace()*) shall be skipped, unless the conversion specification includes a *[], c, C, or n* conversion specifier.

An item shall be read from the input, unless the conversion specification includes an *n* conversion specifier. An input item shall be defined as the longest sequence of input bytes (up to any specified maximum field width, which may be measured in characters or bytes dependent on the conversion specifier) which is an initial subsequence of a matching sequence. The first byte, if any, after the input item shall remain unread. If the length of the input item is 0, the execution of the conversion specification shall fail; this condition is a matching failure, unless end-of-file, an encoding error, or a read error prevented input from the stream, in which case it is an input failure.

Except in the case of a *%* conversion specifier, the input item (or, in the case of a *%n* conversion specification, the count of input bytes) shall be converted to a type appropriate to the conversion character. If the input item is not a matching sequence, the execution of the conversion specification fails; this condition is a matching failure. Unless assignment suppression was indicated by a *'*'*, the result of the conversion shall be placed in the object pointed to by the first argument following the *format* argument that has not already received a conversion result if the conversion specification is introduced by *%*, or in the *n*th argument if introduced by the character sequence *"%n\$"*. If this object does not have an appropriate type, or if the result of the conversion cannot be represented in the space provided, the behavior is undefined.

The *%c*, *%s*, and *[%* conversion specifiers shall accept an optional assignment-allocation character *'m'*, which shall cause a memory buffer to be allocated to hold the string converted including a terminating null character. In such a case, the argument corresponding to the conversion specifier should be a reference to a pointer variable that will receive a pointer to the allocated buffer. The system shall allocate a buffer as if *malloc()* had been called. The application shall be responsible for freeing the memory after usage. If there is insufficient memory to allocate a buffer, the function shall set *errno* to [ENOMEM] and a conversion error shall result. If the function returns EOF, any memory successfully allocated for parameters using assignment-allocation character *'m'* by this call shall be freed before the function returns.

31667	The length modifiers and their meanings are:	
31668	hh	Specifies that a following d, i, o, u, x, X, or n conversion specifier applies to an argument with type pointer to signed char or unsigned char .
31669		
31670	h	Specifies that a following d, i, o, u, x, X, or n conversion specifier applies to an argument with type pointer to short or unsigned short .
31671		
31672	l (ell)	Specifies that a following d, i, o, u, x, X, or n conversion specifier applies to an argument with type pointer to long or unsigned long ; that a following a, A, e, E, f, F, g, or G conversion specifier applies to an argument with type pointer to double ; or that a following c, s, or [conversion specifier applies to an argument with type pointer to wchar_t . If the 'm' assignment-allocation character is specified, the conversion applies to an argument with the type pointer to a pointer to wchar_t .
31673		
31674		
31675		
31676	CX	
31677		
31678	ll (ell-ell)	Specifies that a following d, i, o, u, x, X, or n conversion specifier applies to an argument with type pointer to long long or unsigned long long .
31679		
31680		
31681	j	Specifies that a following d, i, o, u, x, X, or n conversion specifier applies to an argument with type pointer to intmax_t or uintmax_t .
31682		
31683	z	Specifies that a following d, i, o, u, x, X, or n conversion specifier applies to an argument with type pointer to size_t or the corresponding signed integer type.
31684		
31685	t	Specifies that a following d, i, o, u, x, X, or n conversion specifier applies to an argument with type pointer to ptrdiff_t or the corresponding unsigned type.
31686		
31687	L	Specifies that a following a, A, e, E, f, F, g, or G conversion specifier applies to an argument with type pointer to long double .
31688		
31689	If a length modifier appears with any conversion specifier other than as specified above, the behavior is undefined.	
31690		
31691	The following conversion specifiers are valid:	
31692	d	Matches an optionally signed decimal integer, whose format is the same as expected for the subject sequence of <i>strtol()</i> with the value 10 for the <i>base</i> argument. In the absence of a size modifier, the application shall ensure that the corresponding argument is a pointer to int .
31693		
31694		
31695		
31696	i	Matches an optionally signed integer, whose format is the same as expected for the subject sequence of <i>strtol()</i> with 0 for the <i>base</i> argument. In the absence of a size modifier, the application shall ensure that the corresponding argument is a pointer to int .
31697		
31698		
31699		
31700	o	Matches an optionally signed octal integer, whose format is the same as expected for the subject sequence of <i>strtoul()</i> with the value 8 for the <i>base</i> argument. In the absence of a size modifier, the application shall ensure that the corresponding argument is a pointer to unsigned .
31701		
31702		
31703		
31704	u	Matches an optionally signed decimal integer, whose format is the same as expected for the subject sequence of <i>strtoul()</i> with the value 10 for the <i>base</i> argument. In the absence of a size modifier, the application shall ensure that the corresponding argument is a pointer to unsigned .
31705		
31706		
31707		
31708	x	Matches an optionally signed hexadecimal integer, whose format is the same as expected for the subject sequence of <i>strtoul()</i> with the value 16 for the <i>base</i> argument. In the absence of a size modifier, the application shall ensure that the corresponding
31709		
31710		

31711 argument is a pointer to **unsigned**.

31712 a, e, f, g

31713 Matches an optionally signed floating-point number, infinity, or NaN, whose format is

31714 the same as expected for the subject sequence of *strtod*(). In the absence of a size

31715 modifier, the application shall ensure that the corresponding argument is a pointer to

31716 **float**.

31717 If the *fprintf*() family of functions generates character string representations for infinity

31718 and NaN (a symbolic entity encoded in floating-point format) to support

31719 IEEE Std 754-1985, the *fscanf*() family of functions shall recognize them as input.

31720 s

31721 Matches a sequence of bytes that are not white-space characters. If the 'm' assignment-

31722 allocation character is not specified, the application shall ensure that the corresponding

31723 argument is a pointer to the initial byte of an array of **char**, **signed char**, or **unsigned**

31724 **char** large enough to accept the sequence and a terminating null character code, which

31725 CX shall be added automatically. Otherwise, the application shall ensure that the

31726 corresponding argument is a pointer to a pointer to a **char**.

31727 If an l (ell) qualifier is present, the input is a sequence of characters that begins in the

31728 initial shift state. Each character shall be converted to a wide character as if by a call to

31729 the *mbrtowc*() function, with the conversion state described by an **mbstate_t** object

31730 initialized to zero before the first character is converted. If the 'm' assignment-

31731 allocation character is not specified, the application shall ensure that the corresponding

31732 argument is a pointer to an array of **wchar_t** large enough to accept the sequence and

31733 CX the terminating null wide character, which shall be added automatically. Otherwise,

31734 the application shall ensure that the corresponding argument is a pointer to a pointer to

31735 a **wchar_t**.

31736 [

31737 Matches a non-empty sequence of bytes from a set of expected bytes (the *scanset*). The

31738 normal skip over white-space characters shall be suppressed in this case. If the 'm'

31739 assignment-allocation character is not specified, the application shall ensure that the

31740 corresponding argument is a pointer to the initial byte of an array of **char**, **signed char**,

31741 CX or **unsigned char** large enough to accept the sequence and a terminating null byte,

31742 which shall be added automatically. Otherwise, the application shall ensure that the

31743 corresponding argument is a pointer to a pointer to a **char**.

31744 If an l (ell) qualifier is present, the input is a sequence of characters that begins in the

31745 initial shift state. Each character in the sequence shall be converted to a wide character

31746 as if by a call to the *mbrtowc*() function, with the conversion state described by an

31747 **mbstate_t** object initialized to zero before the first character is converted. If the 'm'

31748 assignment-allocation character is not specified, the application shall ensure that the

31749 corresponding argument is a pointer to an array of **wchar_t** large enough to accept the

31750 CX sequence and the terminating null wide character, which shall be added automatically.

31751 Otherwise, the application shall ensure that the corresponding argument is a pointer to

31752 a pointer to a **wchar_t**.

31753 The conversion specification includes all subsequent bytes in the *format* string up to

31754 and including the matching <right-square-bracket> ('*]*'). The bytes between the

31755 square brackets (the *scanlist*) comprise the *scanset*, unless the byte after the <left-

31756 square-bracket> is a <circumflex> ('*^*'), in which case the *scanset* contains all bytes

31757 that do not appear in the *scanlist* between the <circumflex> and the <right-square-

31758 bracket>. If the conversion specification begins with "[*]*" or "[*^*]", the <right-

31759			first <right-square-bracket> is the one that ends the conversion specification. If a '-' is
31760			in the scanlist and is not the first character, nor the second where the first character is a
31761			'^', nor the last character, the behavior is implementation-defined.
31762		c	Matches a sequence of bytes of the number specified by the field width (1 if no field
31763			width is present in the conversion specification). No null byte is added. The normal
31764			skip over white-space characters shall be suppressed in this case. If the 'm'
31765			assignment-allocation character is not specified, the application shall ensure that the
31766			corresponding argument is a pointer to the initial byte of an array of char , signed char ,
31767	CX		or unsigned char large enough to accept the sequence. Otherwise, the application
31768			shall ensure that the corresponding argument is a pointer to a pointer to a char .
31769			If an l (ell) qualifier is present, the input shall be a sequence of characters that begins in
31770			the initial shift state. Each character in the sequence is converted to a wide character as
31771			if by a call to the <i>mbrtowc()</i> function, with the conversion state described by an
31772			mbstate_t object initialized to zero before the first character is converted. No null wide
31773			character is added. If the 'm' assignment-allocation character is not specified, the
31774			application shall ensure that the corresponding argument is a pointer to an array of
31775	CX		wchar_t large enough to accept the resulting sequence of wide characters. Otherwise,
31776			the application shall ensure that the corresponding argument is a pointer to a pointer to
31777			a wchar_t .
31778		p	Matches an implementation-defined set of sequences, which shall be the same as the set
31779			of sequences that is produced by the %p conversion specification of the corresponding
31780			<i>fprintf()</i> functions. The application shall ensure that the corresponding argument is a
31781			pointer to a pointer to void . The interpretation of the input item is implementation-
31782			defined. If the input item is a value converted earlier during the same program
31783			execution, the pointer that results shall compare equal to that value; otherwise, the
31784			behavior of the %p conversion specification is undefined.
31785		n	No input is consumed. The application shall ensure that the corresponding argument is
31786			a pointer to the integer into which shall be written the number of bytes read from the
31787			input so far by this call to the <i>fscanf()</i> functions. Execution of a %n conversion
31788			specification shall not increment the assignment count returned at the completion of
31789			execution of the function. No argument shall be converted, but one shall be consumed.
31790			If the conversion specification includes an assignment-suppressing character or a field
31791			width, the behavior is undefined.
31792	XSI	C	Equivalent to lc.
31793	XSI	S	Equivalent to ls.
31794		%	Matches a single '%' character; no conversion or assignment occurs. The complete
31795			conversion specification shall be %%.
31796			If a conversion specification is invalid, the behavior is undefined.
31797			The conversion specifiers A, E, F, G, and X are also valid and shall be equivalent to a, e, f, g, and
31798			x, respectively.
31799			If end-of-file is encountered during input, conversion shall be terminated. If end-of-file occurs
31800			before any bytes matching the current conversion specification (except for %n) have been read
31801			(other than leading white-space characters, where permitted), execution of the current
31802			conversion specification shall terminate with an input failure. Otherwise, unless execution of the
31803			current conversion specification is terminated with a matching failure, execution of the
31804			following conversion specification (if any) shall be terminated with an input failure.

31805 Reaching the end of the string in *sscanf()* shall be equivalent to encountering end-of-file for
 31806 *fscanf()*.

31807 If conversion terminates on a conflicting input, the offending input is left unread in the input.
 31808 Any trailing white space (including <newline> characters) shall be left unread unless matched
 31809 by a conversion specification. The success of literal matches and suppressed assignments is only
 31810 directly determinable via the %n conversion specification.

31811 CX The *fscanf()* and *scanf()* functions may mark the last data access timestamp of the file associated
 31812 with *stream* for update. The last data access timestamp shall be marked for update by the first
 31813 successful execution of *fgetc()*, *fgets()*, *fread()*, *getc()*, *getchar()*, *getdelim()*, *getline()*, *gets()*,
 31814 *fscanf()*, or *scanf()* using *stream* that returns data not supplied by a prior call to *ungetc()*.

31815 RETURN VALUE

31816 Upon successful completion, these functions shall return the number of successfully matched
 31817 and assigned input items; this number can be zero in the event of an early matching failure. If
 31818 the input ends before the first conversion (if any) has completed, and without a matching failure
 31819 having occurred, EOF shall be returned. If an error occurs before the first conversion (if any) has
 31820 CX completed, and without a matching failure having occurred, EOF shall be returned and *errno*
 31821 shall be set to indicate the error. If a read error occurs, the error indicator for the stream shall be
 31822 set.

31823 ERRORS

31824 For the conditions under which the *fscanf()* functions fail and may fail, refer to *fgetc()* or
 31825 *fgetwc()*.

31826 In addition, the *fscanf()* function shall fail if:

31827 CX [EILSEQ] Input byte sequence does not form a valid character.

31828 [ENOMEM] Insufficient storage space is available.

31829 In addition, the *fscanf()* function may fail if:

31830 CX [EINVAL] There are insufficient arguments.

31831 EXAMPLES

31832 The call:

```
31833 int i, n; float x; char name[50];
31834 n = scanf("%d%f%s", &i, &x, name);
```

31835 with the input line:

```
31836 25 54.32E-1 Hamster
```

31837 assigns to *n* the value 3, to *i* the value 25, to *x* the value 5.432, and *name* contains the string
 31838 "Hamster".

31839 The call:

```
31840 int i; float x; char name[50];
31841 (void) scanf("%2d%f%d %[0123456789]", &i, &x, name);
```

31842 with input:

```
31843 56789 0123 56a72
```

31844 assigns 56 to *i*, 789.0 to *x*, skips 0123, and places the string "56\0" in *name*. The next call to
 31845 *getchar()* shall return the character 'a'.

Reading Data into an Array

The following call uses *fscanf()* to read three floating-point numbers from standard input into the *input* array.

```
float input[3]; fscanf (stdin, "%f %f %f", input, input+1, input+2);
```

APPLICATION USAGE

If the application calling *fscanf()* has any objects of type **wint_t** or **wchar_t**, it must also include the **<wchar.h>** header to have these objects defined.

For functions that allocate memory as if by *malloc()*, the application should release such memory when it is no longer required by a call to *free()*. For *fscanf()*, this is memory allocated via use of the 'm' assignment-allocation character.

RATIONALE

This function is aligned with the ISO/IEC 9899:1999 standard, and in doing so a few “obvious” things were not included. Specifically, the set of characters allowed in a scanset is limited to single-byte characters. In other similar places, multi-byte characters have been permitted, but for alignment with the ISO/IEC 9899:1999 standard, it has not been done here. Applications needing this could use the corresponding wide-character functions to achieve the desired results.

FUTURE DIRECTIONS

None.

SEE ALSO

[Section 2.5](#) (on page 495), *fprintf()*, *getc()*, *setlocale()*, *strtod()*, *strtol()*, *strtoul()*, *wcrtomb()*

[XBD Chapter 7](#) (on page 135), **<langinfo.h>**, **<stdio.h>**, **<wchar.h>**

CHANGE HISTORY

First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 5

Aligned with ISO/IEC 9899:1990/Amendment 1:1995 (E). Specifically, the **l** (ell) qualifier is now defined for the **c**, **s**, and **[** conversion specifiers.

The DESCRIPTION is updated to indicate that if infinity and NaN can be generated by the *fprintf()* family of functions, then they are recognized by the *fscanf()* family.

Issue 6

The Open Group Corrigenda U021/7 and U028/10 are applied. These correct several occurrences of “characters” in the text which have been replaced with the term “bytes”.

The normative text is updated to avoid use of the term “must” for application requirements.

The following changes are made for alignment with the ISO/IEC 9899:1999 standard:

- The prototypes for *fscanf()*, *scanf()*, and *sscanf()* are updated.
- The DESCRIPTION is updated.
- The **hh**, **ll**, **j**, **t**, and **z** length modifiers are added.
- The **a**, **A**, and **F** conversion characters are added.

The DESCRIPTION is updated to use the terms “conversion specifier” and “conversion specification” consistently.

31886 **Issue 7**

31887 Austin Group Interpretation 1003.1-2001 #170 is applied.

31888 SD5-XSH-ERN-9 is applied, correcting *fscanf()* to *scanf()* in the DESCRIPTION.

31889 SD5-XSH-ERN-132 is applied, adding the assignment-allocation character 'm'.

31890 Functionality relating to the %n\$ form of conversion specification is moved from the XSI option
31891 to the Base.

31892 Changes are made related to support for finegrained timestamps.

31893 The APPLICATION USAGE section is updated to clarify that memory is allocated as if by
31894 *malloc()*.

31895 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0185 [302], XSH/TC1-2008/0186 [90], +
31896 and XSH/TC1-2008/0187 [14] are applied. XSH/TC1-2008/0186 [90] changes the second +
31897 sentence in the RETURN VALUE section to align with expected wording changes in the next +
31898 revision of the ISO C standard.

31899 **NAME**

31900 fseek, fseeko — reposition a file-position indicator in a stream

31901 **SYNOPSIS**

31902 #include <stdio.h>

31903 int fseek(FILE *stream, long offset, int whence);

31904 CX int fseeko(FILE *stream, off_t offset, int whence);

31905 **DESCRIPTION**

31906 CX The functionality described on this reference page is aligned with the ISO C standard. Any
 31907 conflict between the requirements described here and the ISO C standard is unintentional. This
 31908 volume of POSIX.1-2008 defers to the ISO C standard.

31909 The *fseek()* function shall set the file-position indicator for the stream pointed to by *stream*. If a
 31910 read or write error occurs, the error indicator for the stream shall be set and *fseek()* fails.

31911 The new position, measured in bytes from the beginning of the file, shall be obtained by adding
 31912 *offset* to the position specified by *whence*. The specified point is the beginning of the file for
 31913 SEEK_SET, the current value of the file-position indicator for SEEK_CUR, or end-of-file for
 31914 SEEK_END.

31915 If the stream is to be used with wide-character input/output functions, the application shall
 31916 ensure that *offset* is either 0 or a value returned by an earlier call to *ftell()* on the same stream and
 31917 *whence* is SEEK_SET.

31918 A successful call to *fseek()* shall clear the end-of-file indicator for the stream and undo any effects
 31919 of *ungetc()* and *ungetwc()* on the same stream. After an *fseek()* call, the next operation on an
 31920 update stream may be either input or output.

31921 CX If the most recent operation, other than *ftell()*, on a given stream is *fflush()*, the file offset in the
 31922 underlying open file description shall be adjusted to reflect the location specified by *fseek()*.

31923 The *fseek()* function shall allow the file-position indicator to be set beyond the end of existing
 31924 data in the file. If data is later written at this point, subsequent reads of data in the gap shall
 31925 return bytes with the value 0 until data is actually written into the gap.

31926 The behavior of *fseek()* on devices which are incapable of seeking is implementation-defined.
 31927 The value of the file offset associated with such a device is undefined.

31928 If the stream is writable and buffered data had not been written to the underlying file, *fseek()*
 31929 shall cause the unwritten data to be written to the file and shall mark the last data modification
 31930 and last file status change timestamps of the file for update.

31931 In a locale with state-dependent encoding, whether *fseek()* restores the stream's shift state is
 31932 implementation-defined.

31933 The *fseeko()* function shall be equivalent to the *fseek()* function except that the *offset* argument is
 31934 of type **off_t**.

31935 **RETURN VALUE**31936 CX The *fseek()* and *fseeko()* functions shall return 0 if they succeed.31937 CX Otherwise, they shall return -1 and set *errno* to indicate the error.31938 **ERRORS**

31939 CX The *fseek()* and *fseeko()* functions shall fail if, either the *stream* is unbuffered or the *stream's*
 31940 buffer needed to be flushed, and the call to *fseek()* or *fseeko()* causes an underlying *lseek()* or
 31941 *write()* to be invoked, and:

31942	CX	[EAGAIN]	The O_NONBLOCK flag is set for the file descriptor and the thread would be delayed in the write operation.	
31943				
31944	CX	[EBADF]	The file descriptor underlying the stream file is not open for writing or the stream's buffer needed to be flushed and the file is not open.	
31945				
31946	CX	[EFBIG]	An attempt was made to write a file that exceeds the maximum file size.	
31947	XSI	[EFBIG]	An attempt was made to write a file that exceeds the file size limit of the process.	
31948				
31949	CX	[EFBIG]	The file is a regular file and an attempt was made to write at or beyond the offset maximum associated with the corresponding stream.	
31950				
31951	CX	[EINTR]	The write operation was terminated due to the receipt of a signal, and no data was transferred.	
31952				
31953	CX	[EINVAL]	The <i>whence</i> argument is invalid. The resulting file-position indicator would be set to a negative value.	
31954				
31955	CX	[EIO]	A physical I/O error has occurred, or the process is a member of a background process group attempting to perform a <i>write()</i> to its controlling terminal, TOSTOP is set, the calling thread is not blocking SIGTTOU, the process is not ignoring SIGTTOU, and the process group of the process is orphaned. This error may also be returned under implementation-defined conditions.	
31956				
31957				
31958				
31959				
31960	CX	[ENOSPC]	There was no free space remaining on the device containing the file.	-
31961	CX	[EOVERFLOW]	For <i>fseek()</i> , the resulting file offset would be a value which cannot be represented correctly in an object of type long .	
31962				
31963	CX	[EOVERFLOW]	For <i>fseeko()</i> , the resulting file offset would be a value which cannot be represented correctly in an object of type off_t .	
31964				
31965	CX	[EPIPE]	An attempt was made to write to a pipe or FIFO that is not open for reading by any process; a SIGPIPE signal shall also be sent to the thread.	
31966				
31967	CX	[ESPIPE]	The file descriptor underlying <i>stream</i> is associated with a pipe, FIFO, or socket.	
31968				
31969	CX	The <i>fseek()</i> and <i>fseeko()</i> functions may fail if:		
31970	CX	[ENXIO]	A request was made of a nonexistent device, or the request was outside the capabilities of the device.	
31971				

EXAMPLES

None.

APPLICATION USAGE

None.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

Section 2.5 (on page 495), *fopen()*, *fsetpos()*, *ftell()*, *getrlimit()*, *lseek()*, *rewind()*, *ulimit()*, *ungetc()*, *write()*

XBD <stdio.h>

31984 **CHANGE HISTORY**

31985 First released in Issue 1. Derived from Issue 1 of the SVID.

31986 **Issue 5**31987 Normative text previously in the APPLICATION USAGE section is moved to the
31988 DESCRIPTION.

31989 Large File Summit extensions are added.

31990 **Issue 6**

31991 Extensions beyond the ISO C standard are marked.

31992 The following new requirements on POSIX implementations derive from alignment with the
31993 Single UNIX Specification:31994 • The *fseeko()* function is added.

31995 • The [EFBIG], [EOVERFLOW], and [ENXIO] mandatory error conditions are added.

31996 The following change is incorporated for alignment with the FIPS requirements:

31997 • The [EINTR] error is no longer an indication that the implementation does not report
31998 partial transfers.

31999 The normative text is updated to avoid use of the term “must” for application requirements.

32000 The DESCRIPTION is updated to explicitly state that *fseek()* sets the file-position indicator, and
32001 then on error the error indicator is set and *fseek()* fails. This is for alignment with the
32002 ISO/IEC 9899:1999 standard.32003 IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/42 is applied, updating the [EAGAIN]
32004 error in the ERRORS section from “the process would be delayed” to “the thread would be
32005 delayed”.32006 **Issue 7**

32007 Changes are made related to support for finegrained timestamps.

32008 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0188 [79], XSH/TC1-2008/0189 [122], +
32009 XSH/TC1-2008/0190 [225], and XSH/TC1-2008/0191 [14] are applied.

32010 **NAME**

32011 fsetpos — set current file position

32012 **SYNOPSIS**

32013 #include <stdio.h>

32014 int fsetpos(FILE **stream*, const fpos_t **pos*);32015 **DESCRIPTION**

32016 CX The functionality described on this reference page is aligned with the ISO C standard. Any
 32017 conflict between the requirements described here and the ISO C standard is unintentional. This
 32018 volume of POSIX.1-2008 defers to the ISO C standard.

32019 The *fsetpos()* function shall set the file position and state indicators for the stream pointed to by
 32020 *stream* according to the value of the object pointed to by *pos*, which the application shall ensure is
 32021 a value obtained from an earlier call to *fgetpos()* on the same stream. If a read or write error
 32022 occurs, the error indicator for the stream shall be set and *fsetpos()* fails.

32023 A successful call to the *fsetpos()* function shall clear the end-of-file indicator for the stream and
 32024 undo any effects of *ungetc()* on the same stream. After an *fsetpos()* call, the next operation on an
 32025 update stream may be either input or output.

32026 CX The behavior of *fsetpos()* on devices which are incapable of seeking is implementation-defined.
 32027 The value of the file offset associated with such a device is undefined.

32028 The *fsetpos()* function shall not change the setting of *errno* if successful.

+

32029 **RETURN VALUE**

32030 The *fsetpos()* function shall return 0 if it succeeds; otherwise, it shall return a non-zero value and
 32031 set *errno* to indicate the error.

32032 **ERRORS**

32033 CX The *fsetpos()* function shall fail if, either the *stream* is unbuffered or the *stream*'s buffer needed to
 32034 be flushed, and the call to *fsetpos()* causes an underlying *lseek()* or *write()* to be invoked, and:

32035 CX [EAGAIN] The O_NONBLOCK flag is set for the file descriptor and the thread would be
 32036 delayed in the write operation.

32037 CX [EBADF] The file descriptor underlying the stream file is not open for writing or the
 32038 stream's buffer needed to be flushed and the file is not open.

32039 CX [EFBIG] An attempt was made to write a file that exceeds the maximum file size.

32040 XSI [EFBIG] An attempt was made to write a file that exceeds the file size limit of the
 32041 process.

32042 CX [EFBIG] The file is a regular file and an attempt was made to write at or beyond the
 32043 offset maximum associated with the corresponding stream.

32044 CX [EINTR] The write operation was terminated due to the receipt of a signal, and no data
 32045 was transferred.

32046 CX [EIO] A physical I/O error has occurred, or the process is a member of a background
 32047 process group attempting to perform a *write()* to its controlling terminal,
 32048 TOSTOP is set, the calling thread is not blocking SIGTTOU, the process is not
 32049 ignoring SIGTTOU, and the process group of the process is orphaned. This
 32050 error may also be returned under implementation-defined conditions.

32051 CX [ENOSPC] There was no free space remaining on the device containing the file.

-

32052 CX [EPIPE] An attempt was made to write to a pipe or FIFO that is not open for reading
 32053 by any process; a SIGPIPE signal shall also be sent to the thread.

32054 CX [ESPIPE] The file descriptor underlying *stream* is associated with a pipe, FIFO, or socket.
 32055

32056 The *fsetpos()* function may fail if:

32057 CX [ENXIO] A request was made of a nonexistent device, or the request was outside the
 32058 capabilities of the device.

32059 EXAMPLES

32060 None.

32061 APPLICATION USAGE

32062 None.

32063 RATIONALE

32064 None.

32065 FUTURE DIRECTIONS

32066 None.

32067 SEE ALSO

32068 [Section 2.5](#) (on page 495), *fopen()*, *ftell()*, *lseek()*, *rewind()*, *ungetc()*, *write()*

32069 XBD `<stdio.h>`

32070 CHANGE HISTORY

32071 First released in Issue 4. Derived from the ISO C standard.

32072 Issue 6

32073 Extensions beyond the ISO C standard are marked.

32074 The normative text is updated to avoid use of the term “must” for application requirements. -

32075 The DESCRIPTION is updated to clarify that the error indicator is set for the stream on a read or
 32076 write error. This is for alignment with the ISO/IEC 9899:1999 standard.

32077 IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/21 is applied, deleting an erroneous
 32078 [EINVAL] error case from the ERRORS section.

32079 IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/43 is applied, updating the [EAGAIN]
 32080 error in the ERRORS section from “the process would be delayed” to “the thread would be
 32081 delayed”.

32082 Issue 7

32083 SD5-XSH-ERN-220 is applied.

32084 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0192 [105], XSH/TC1-2008/0193 [79], +
 32085 XSH/TC1-2008/0194 [225], XSH/TC1-2008/0195 [450], XSH/TC1-2008/0196 [450], and +
 32086 XSH/TC1-2008/0197 [14] are applied.

32087 **NAME**32088 `fstat` — get file status32089 **SYNOPSIS**32090 `#include <sys/stat.h>`32091 `int fstat(int fildes, struct stat *buf);`32092 **DESCRIPTION**32093 The `fstat()` function shall obtain information about an open file associated with the file
32094 descriptor *fildes*, and shall write it to the area pointed to by *buf*.32095 SHM If *fildes* references a shared memory object, the implementation shall update in the **stat** structure
32096 pointed to by the *buf* argument the *st_uid*, *st_gid*, *st_size*, and *st_mode* fields, and only the
32097 S_IRUSR, S_IWUSR, S_IRGRP, S_IWGRP, S_IROTH, and S_IWOTH file permission bits need be
32098 valid. The implementation may update other fields and flags.32099 TYM If *fildes* references a typed memory object, the implementation shall update in the **stat** structure
32100 pointed to by the *buf* argument the *st_uid*, *st_gid*, *st_size*, and *st_mode* fields, and only the
32101 S_IRUSR, S_IWUSR, S_IRGRP, S_IWGRP, S_IROTH, and S_IWOTH file permission bits need be
32102 valid. The implementation may update other fields and flags.32103 The *buf* argument is a pointer to a **stat** structure, as defined in `<sys/stat.h>`, into which
32104 information is placed concerning the file.32105 For all other file types defined in this volume of POSIX.1-2008, the structure members *st_mode*,
32106 *st_ino*, *st_dev*, *st_uid*, *st_gid*, *st_atim*, *st_ctim*, and *st_mtim* shall have meaningful values and the
32107 value of the *st_nlink* member shall be set to the number of links to the file.32108 An implementation that provides additional or alternative file access control mechanisms may,
32109 under implementation-defined conditions, cause `fstat()` to fail.32110 The `fstat()` function shall update any time-related fields (as described in XBD [Section 4.8](#), on
32111 page 109), before writing into the **stat** structure.32112 **RETURN VALUE**32113 Upon successful completion, 0 shall be returned. Otherwise, -1 shall be returned and *errno* set to
32114 indicate the error.32115 **ERRORS**32116 The `fstat()` function shall fail if:32117 [EBADF] The *fildes* argument is not a valid file descriptor.

32118 [EIO] An I/O error occurred while reading from the file system.

32119 [EOVERFLOW] The file size in bytes or the number of blocks allocated to the file or the file
32120 serial number cannot be represented correctly in the structure pointed to by
32121 *buf*.32122 The `fstat()` function may fail if:32123 [EOVERFLOW] One of the values is too large to store into the structure pointed to by the *buf*
32124 argument.

32125 **EXAMPLES**32126 **Obtaining File Status Information**

32127 The following example shows how to obtain file status information for a file named
 32128 **/home/cnd/mod1**. The structure variable *buffer* is defined for the **stat** structure. The
 32129 **/home/cnd/mod1** file is opened with read/write privileges and is passed to the open file
 32130 descriptor *fildes*.

```
32131 #include <sys/types.h>
32132 #include <sys/stat.h>
32133 #include <fcntl.h>

32134 struct stat buffer;
32135 int      status;
32136 ...
32137 fildes = open("/home/cnd/mod1", O_RDWR);
32138 status = fstat(fildes, &buffer);
```

32139 **APPLICATION USAGE**

32140 None.

32141 **RATIONALE**

32142 None.

32143 **FUTURE DIRECTIONS**

32144 None.

32145 **SEE ALSO**

32146 [*fstatat\(\)*](#)

32147 XBD [Section 4.8](#) (on page 109), [**<sys/stat.h>**](#), [**<sys/types.h>**](#)

32148 **CHANGE HISTORY**

32149 First released in Issue 1. Derived from Issue 1 of the SVID.

32150 **Issue 5**

32151 The DESCRIPTION is updated for alignment with the POSIX Realtime Extension.

32152 Large File Summit extensions are added.

32153 **Issue 6**

32154 In the SYNOPSIS, the optional include of the **<sys/types.h>** header is removed.

32155 The following new requirements on POSIX implementations derive from alignment with the
 32156 Single UNIX Specification:

- 32157 • The requirement to include **<sys/types.h>** has been removed. Although **<sys/types.h>** was
 32158 required for conforming implementations of previous POSIX specifications, it was not
 32159 required for UNIX applications.
- 32160 • The [EIO] mandatory error condition is added.
- 32161 • The [EOVERFLOW] mandatory error condition is added. This change is to support large
 32162 files.
- 32163 • The [EOVERFLOW] optional error condition is added.

32164 The DESCRIPTION is updated for alignment with IEEE Std 1003.1j-2000 by specifying that
 32165 shared memory object semantics apply to typed memory objects.

Issue 7

32166 XSH-SD5-ERN-161 is applied, updating the DESCRIPTION to clarify to which file types *st_nlink*
32167 applies.
32168

32169 Changes are made related to support for finegrained timestamps.

NAME

fstatat, lstat, stat — get file status

SYNOPSIS

```
#include <sys/stat.h>

int fstatat(int fd, const char *restrict path,
            struct stat *restrict buf, int flag);
int lstat(const char *restrict path, struct stat *restrict buf);
int stat(const char *restrict path, struct stat *restrict buf);
```

DESCRIPTION

The *stat()* function shall obtain information about the named file and write it to the area pointed to by the *buf* argument. The *path* argument points to a pathname naming a file. Read, write, or execute permission of the named file is not required. An implementation that provides additional or alternate file access control mechanisms may, under implementation-defined conditions, cause *stat()* to fail. In particular, the system may deny the existence of the file specified by *path*.

If the named file is a symbolic link, the *stat()* function shall continue pathname resolution using the contents of the symbolic link, and shall return information pertaining to the resulting file if the file exists.

The *buf* argument is a pointer to a **stat** structure, as defined in the **<sys/stat.h>** header, into which information is placed concerning the file.

The *stat()* function shall update any time-related fields (as described in XBD [Section 4.8](#), on page 109), before writing into the **stat** structure.

SHM

If the named file is a shared memory object, the implementation shall update in the **stat** structure pointed to by the *buf* argument the *st_uid*, *st_gid*, *st_size*, and *st_mode* fields, and only the S_IRUSR, S_IWUSR, S_IRGRP, S_IWGRP, S_IROTH, and S_IWOTH file permission bits need be valid. The implementation may update other fields and flags.

TYM

If the named file is a typed memory object, the implementation shall update in the **stat** structure pointed to by the *buf* argument the *st_uid*, *st_gid*, *st_size*, and *st_mode* fields, and only the S_IRUSR, S_IWUSR, S_IRGRP, S_IWGRP, S_IROTH, and S_IWOTH file permission bits need be valid. The implementation may update other fields and flags.

For all other file types defined in this volume of POSIX.1-2008, the structure members *st_mode*, *st_ino*, *st_dev*, *st_uid*, *st_gid*, *st_atim*, *st_ctim*, and *st_mtim* shall have meaningful values and the value of the member *st_nlink* shall be set to the number of links to the file.

The *lstat()* function shall be equivalent to *stat()*, except when *path* refers to a symbolic link. In that case *lstat()* shall return information about the link, while *stat()* shall return information about the file the link references.

For symbolic links, the *st_mode* member shall contain meaningful information when used with the file type macros. The file mode bits in *st_mode* are unspecified. The structure members *st_ino*, *st_dev*, *st_uid*, *st_gid*, *st_atim*, *st_ctim*, and *st_mtim* shall have meaningful values and the value of the *st_nlink* member shall be set to the number of (hard) links to the symbolic link. The value of the *st_size* member shall be set to the length of the pathname contained in the symbolic link not including any terminating null byte.

The *fstatat()* function shall be equivalent to the *stat()* or *lstat()* function, depending on the value of *flag* (see below), except in the case where *path* specifies a relative path. In this case the status shall be retrieved from a file relative to the directory associated with the file descriptor *fd* instead of the current working directory. If the file descriptor was opened without O_SEARCH, the

32216 function shall check whether directory searches are permitted using the current permissions of
 32217 the directory underlying the file descriptor. If the file descriptor was opened with O_SEARCH,
 32218 the function shall not perform the check.

32219 Values for *flag* are constructed by a bitwise-inclusive OR of flags from the following list, defined
 32220 in **<fcntl.h>**:

32221 AT_SYMLINK_NOFOLLOW

32222 If *path* names a symbolic link, the status of the symbolic link is returned.

32223 If *fstatat()* is passed the special value AT_FDCWD in the *fd* parameter, the current working
 32224 directory shall be used and the behavior shall be identical to a call to *stat()* or *lstat()* respectively,
 32225 depending on whether or not the AT_SYMLINK_NOFOLLOW bit is set in *flag*.

32226 RETURN VALUE

32227 Upon successful completion, these functions shall return 0. Otherwise, these functions shall
 32228 return -1 and set *errno* to indicate the error.

32229 ERRORS

32230 These functions shall fail if:

32231 [EACCES] Search permission is denied for a component of the path prefix.

32232 [EIO] An error occurred while reading from the file system.

32233 [ELOOP] A loop exists in symbolic links encountered during resolution of the *path*
 32234 argument.

32235 [ENAMETOOLONG]

32236 The length of a component of a pathname is longer than {NAME_MAX}.

32237 [ENOENT] A component of *path* does not name an existing file or *path* is an empty string.

32238 [ENOTDIR] A component of the path prefix names an existing file that is neither a
 32239 directory nor a symbolic link to a directory, or the *path* argument contains at
 32240 least one non-*<slash>* character and ends with one or more trailing *<slash>*
 32241 characters and the last pathname component names an existing file that is
 32242 neither a directory nor a symbolic link to a directory.

32243 [EOVERFLOW] The file size in bytes or the number of blocks allocated to the file or the file
 32244 serial number cannot be represented correctly in the structure pointed to by
 32245 *buf*.

32246 The *fstatat()* function shall fail if:

32247 [EACCES] *fd* was not opened with O_SEARCH and the permissions of the directory
 32248 underlying *fd* do not permit directory searches.

32249 [EBADF] The *path* argument does not specify an absolute path and the *fd* argument is
 32250 neither AT_FDCWD nor a valid file descriptor open for reading or searching.

32251 [ENOTDIR] The *path* argument is not an absolute path and *fd* is a file descriptor associated +
 32252 with a non-directory file.

32253 These functions may fail if:

32254 [ELOOP] More than {SYMLOOP_MAX} symbolic links were encountered during
 32255 resolution of the *path* argument.

[ENAMETOOLONG]

The length of a pathname exceeds {PATH_MAX}, or pathname resolution of a symbolic link produced an intermediate result with a length that exceeds {PATH_MAX}.

[EOVERFLOW] A value to be stored would overflow one of the members of the **stat** structure.

The *fstatat()* function may fail if:

[EINVAL] The value of the *flag* argument is not valid.

EXAMPLES

Obtaining File Status Information

The following example shows how to obtain file status information for a file named **/home/cnd/mod1**. The structure variable *buffer* is defined for the **stat** structure.

```
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>

struct stat buffer;
int      status;
...
status = stat("/home/cnd/mod1", &buffer);
```

Getting Directory Information

The following example fragment gets status information for each entry in a directory. The call to the *stat()* function stores file information in the **stat** structure pointed to by *statbuf*. The lines that follow the *stat()* call format the fields in the **stat** structure for presentation to the user of the program.

```
#include <sys/types.h>
#include <sys/stat.h>
#include <dirent.h>
#include <pwd.h>
#include <grp.h>
#include <time.h>
#include <locale.h>
#include <langinfo.h>
#include <stdio.h>
#include <stdint.h>

struct dirent *dp;
struct stat   statbuf;
struct passwd *pwd;
struct group  *grp;
struct tm     *tm;
char          datestring[256];
...
/* Loop through directory entries. */
while ((dp = readdir(dir)) != NULL) {

    /* Get entry's information. */
    if (stat(dp->d_name, &statbuf) == -1)
```

```

32300         continue;

32301         /* Print out type, permissions, and number of links. */
32302         printf("%10.10s", spem (statbuf.st_mode));
32303         printf("%4d", statbuf.st_nlink);

32304         /* Print out owner's name if it is found using getpwuid(). */
32305         if ((pwd = getpwuid(statbuf.st_uid)) != NULL)
32306             printf(" %-8.8s", pwd->pw_name);
32307         else
32308             printf(" %-8d", statbuf.st_uid);

32309         /* Print out group name if it is found using getgrgid(). */
32310         if ((grp = getgrgid(statbuf.st_gid)) != NULL)
32311             printf(" %-8.8s", grp->gr_name);
32312         else
32313             printf(" %-8d", statbuf.st_gid);

32314         /* Print size of file. */
32315         printf(" %9jd", (intmax_t)statbuf.st_size);

32316         tm = localtime(&statbuf.st_mtime);

32317         /* Get localized date string. */
32318         strftime(datestring, sizeof(datestring), nl_langinfo(D_T_FMT), tm);

32319         printf(" %s %s\n", datestring, dp->d_name);
32320     }

```

Obtaining Symbolic Link Status Information

The following example shows how to obtain status information for a symbolic link named **/modules/pass1**. The structure variable *buffer* is defined for the **stat** structure. If the *path* argument specified the pathname for the file pointed to by the symbolic link (**/home/cnd/mod1**), the results of calling the function would be the same as those returned by a call to the *stat()* function.

```

32327 #include <sys/stat.h>

32328 struct stat buffer;
32329 int status;
32330 ...
32331 status = lstat("/modules/pass1", &buffer);

```

APPLICATION USAGE

None.

RATIONALE

The intent of the paragraph describing “additional or alternate file access control mechanisms” is to allow a secure implementation where a process with a label that does not dominate the file’s label cannot perform a *stat()* function. This is not related to read permission; a process with a label that dominates the file’s label does not need read permission. An implementation that supports write-up operations could fail *fstat()* function calls even though it has a valid file descriptor open for writing.

The *lstat()* function is not required to update the time-related fields if the named file is not a symbolic link. While the *st_uid*, *st_gid*, *st_atim*, *st_mtim*, and *st_ctim* members of the **stat** structure may apply to a symbolic link, they are not required to do so. No functions in POSIX.1-2008 are

required to maintain any of these time fields.

The purpose of the *fstatat()* function is to obtain the status of files in directories other than the current working directory without exposure to race conditions. Any part of the path of a file could be changed in parallel to a call to *stat()*, resulting in unspecified behavior. By opening a file descriptor for the target directory and using the *fstatat()* function it can be guaranteed that the file for which status is returned is located relative to the desired directory.

FUTURE DIRECTIONS

None.

SEE ALSO

access(), *chmod()*, *fdopendir()*, *fstat()*, *mknod()*, *readlink()*, *symlink()*

XBD Section 4.8 (on page 109), *<fcntl.h>*, *<sys/stat.h>*, *<sys/types.h>*

CHANGE HISTORY

First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 5

Large File Summit extensions are added.

Issue 6

In the SYNOPSIS, the optional include of the *<sys/types.h>* header is removed.

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The requirement to include *<sys/types.h>* has been removed. Although *<sys/types.h>* was required for conforming implementations of previous POSIX specifications, it was not required for UNIX applications.
- The [EIO] mandatory error condition is added.
- The [ELOOP] mandatory error condition is added.
- The [EOVERFLOW] mandatory error condition is added. This change is to support large files.
- The [ENAMETOOLONG] and the second [EOVERFLOW] optional error conditions are added.

The following changes were made to align with the IEEE P1003.1a draft standard:

- Details are added regarding the treatment of symbolic links.
- The [ELOOP] optional error condition is added.

The normative text is updated to avoid use of the term “must” for application requirements.

The **restrict** keyword is added to the *stat()* prototype for alignment with the ISO/IEC 9899:1999 standard.

Issue 7

Austin Group Interpretation 1003.1-2001 #143 is applied.

XSH-SD5-ERN-161 is applied, updating the DESCRIPTION to clarify to which file types *st_nlink* applies.

The *fstatat()* function is added from The Open Group Technical Standard, 2006, Extended API Set Part 2.

Changes are made related to support for finegrained timestamps.

32385 The *lstat()* function is now required to return meaningful data for symbolic links in all **stat**
32386 structure fields, except for the permission bits of *st_mode*.

32387 Changes are made to allow a directory to be opened for searching.

32388 The [ENOTDIR] error condition is clarified to cover the condition where the last component of a
32389 pathname exists but is not a directory or a symbolic link to a directory.

32390 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0198 [461], XSH/TC1-2008/0199 [324], +
32391 XSH/TC1-2008/0200 [278], XSH/TC1-2008/0201 [278], and XSH/TC1-2008/0202 [291] are +
32392 applied.

NAME

fstatvfs, statvfs — get file system information

SYNOPSIS

```
#include <sys/statvfs.h>
```

```
int fstatvfs(int fildes, struct statvfs *buf);
```

```
int statvfs(const char *restrict path, struct statvfs *restrict buf);
```

DESCRIPTION

The *fstatvfs()* function shall obtain information about the file system containing the file referenced by *fildes*.

The *statvfs()* function shall obtain information about the file system containing the file named by *path*.

For both functions, the *buf* argument is a pointer to a **statvfs** structure that shall be filled. Read, write, or execute permission of the named file is not required.

The following flags can be returned in the *f_flag* member:

ST_RDONLY Read-only file system.

ST_NOSUID Setuid/setgid bits ignored by *exec*.

It is unspecified whether all members of the **statvfs** structure have meaningful values on all file systems.

RETURN VALUE

Upon successful completion, *statvfs()* shall return 0. Otherwise, it shall return -1 and set *errno* to indicate the error.

ERRORS

The *fstatvfs()* and *statvfs()* functions shall fail if:

[EIO] An I/O error occurred while reading the file system.

[EINTR] A signal was caught during execution of the function.

[EOVERFLOW] One of the values to be returned cannot be represented correctly in the structure pointed to by *buf*.

The *fstatvfs()* function shall fail if:

[EBADF] The *fildes* argument is not an open file descriptor.

The *statvfs()* function shall fail if:

[EACCES] Search permission is denied on a component of the path prefix.

[ELOOP] A loop exists in symbolic links encountered during resolution of the *path* argument.

[ENAMETOOLONG]

The length of a component of a pathname is longer than {NAME_MAX}.

[ENOENT] A component of *path* does not name an existing file or *path* is an empty string.

[ENOTDIR] A component of the path prefix names an existing file that is neither a directory nor a symbolic link to a directory, or the *path* argument contains at least one non-*<slash>* character and ends with one or more trailing *<slash>* characters and the last pathname component names an existing file that is neither a directory nor a symbolic link to a directory.

32434 The *statvfs()* function may fail if:

32435 [ELOOP] More than {SYMLOOP_MAX} symbolic links were encountered during

32436 resolution of the *path* argument.

32437 [ENAMETOOLONG]

32438 The length of a pathname exceeds {PATH_MAX}, or pathname resolution of a

32439 symbolic link produced an intermediate result with a length that exceeds

32440 {PATH_MAX}.

32441 EXAMPLES

32442 Obtaining File System Information Using *fstatvfs()*

32443 The following example shows how to obtain file system information for the file system upon

32444 which the file named **/home/cnd/mod1** resides, using the *fstatvfs()* function. The

32445 **/home/cnd/mod1** file is opened with read/write privileges and the open file descriptor is passed

32446 to the *fstatvfs()* function.

```
32447 #include <sys/statvfs.h>
32448 #include <fcntl.h>

32449 struct statvfs buffer;
32450 int          status;
32451 ...
32452 fildes = open("/home/cnd/mod1", O_RDWR);
32453 status = fstatvfs(fildes, &buffer);
```

32454 Obtaining File System Information Using *statvfs()*

32455 The following example shows how to obtain file system information for the file system upon

32456 which the file named **/home/cnd/mod1** resides, using the *statvfs()* function.

```
32457 #include <sys/statvfs.h>

32458 struct statvfs buffer;
32459 int          status;
32460 ...
32461 status = statvfs("/home/cnd/mod1", &buffer);
```

32462 APPLICATION USAGE

32463 None.

32464 RATIONALE

32465 None.

32466 FUTURE DIRECTIONS

32467 None.

32468 SEE ALSO

32469 *chmod()*, *chown()*, *creat()*, *dup()*, *exec*, *fcntl()*, *link()*, *mknod()*, *open()*, *pipe()*, *read()*, *time()*,

32470 *unlink()*, *utime()*, *write()*

32471 XBD *<sys/statvfs.h>*

32472 **CHANGE HISTORY**

32473 First released in Issue 4, Version 2.

32474 **Issue 5**

32475 Moved from X/OPEN UNIX extension to BASE.

32476 Large File Summit extensions are added.

32477 **Issue 6**

32478 The normative text is updated to avoid use of the term “must” for application requirements.

32479 The **restrict** keyword is added to the *statvfs()* prototype for alignment with the
32480 ISO/IEC 9899:1999 standard.32481 The wording of the mandatory [ELOOP] error condition is updated, and a second optional
32482 [ELOOP] error condition is added.32483 **Issue 7**

32484 Austin Group Interpretation 1003.1-2001 #143 is applied.

32485 SD5-XSH-ERN-68 is applied, correcting the EXAMPLES section.

32486 The *fstatvfs()* and *statvfs()* functions are moved from the XSI option to the Base.32487 The [ENOTDIR] error condition is clarified to cover the condition where the last component of a
32488 pathname exists but is not a directory or a symbolic link to a directory.

32489 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0203 [324] is applied.

+

32490 **NAME**

32491 fsync — synchronize changes to a file

32492 **SYNOPSIS**

```
32493 #include <unistd.h>
32494 int fsync(int fildes);
```

32495 **DESCRIPTION**

32496 The *fsync()* function shall request that all data for the open file descriptor named by *fildes* is to be
 32497 transferred to the storage device associated with the file described by *fildes*. The nature of the
 32498 transfer is implementation-defined. The *fsync()* function shall not return until the system has
 32499 completed that action or until an error is detected.

32500 SIO If `_POSIX_SYNCHRONIZED_IO` is defined, the *fsync()* function shall force all currently queued
 32501 I/O operations associated with the file indicated by file descriptor *fildes* to the synchronized I/O
 32502 completion state. All I/O operations shall be completed as defined for synchronized I/O file
 32503 integrity completion.

32504 **RETURN VALUE**

32505 Upon successful completion, *fsync()* shall return 0. Otherwise, `-1` shall be returned and *errno* set
 32506 to indicate the error. If the *fsync()* function fails, outstanding I/O operations are not guaranteed
 32507 to have been completed.

32508 **ERRORS**32509 The *fsync()* function shall fail if:

- 32510 [EBADF] The *fildes* argument is not a valid descriptor.
- 32511 [EINTR] The *fsync()* function was interrupted by a signal.
- 32512 [EINVAL] The *fildes* argument does not refer to a file on which this operation is possible.
- 32513 [EIO] An I/O error occurred while reading from or writing to the file system.

32514 In the event that any of the queued I/O operations fail, *fsync()* shall return the error conditions
 32515 defined for *read()* and *write()*.

32516 **EXAMPLES**

32517 None.

32518 **APPLICATION USAGE**

32519 The *fsync()* function should be used by programs which require modifications to a file to be
 32520 completed before continuing; for example, a program which contains a simple transaction
 32521 facility might use it to ensure that all modifications to a file or files caused by a transaction are
 32522 recorded.

32523 **RATIONALE**

32524 The *fsync()* function is intended to force a physical write of data from the buffer cache, and to
 32525 assure that after a system crash or other failure that all data up to the time of the *fsync()* call is
 32526 recorded on the disk. Since the concepts of “buffer cache”, “system crash”, “physical write”, and
 32527 “non-volatile storage” are not defined here, the wording has to be more abstract.

32528 If `_POSIX_SYNCHRONIZED_IO` is not defined, the wording relies heavily on the conformance
 32529 document to tell the user what can be expected from the system. It is explicitly intended that a
 32530 null implementation is permitted. This could be valid in the case where the system cannot assure
 32531 non-volatile storage under any circumstances or when the system is highly fault-tolerant and the
 32532 functionality is not required. In the middle ground between these extremes, *fsync()* might or
 32533 might not actually cause data to be written where it is safe from a power failure. The

conformance document should identify at least that one configuration exists (and how to obtain that configuration) where this can be assured for at least some files that the user can select to use for critical data. It is not intended that an exhaustive list is required, but rather sufficient information is provided so that if critical data needs to be saved, the user can determine how the system is to be configured to allow the data to be written to non-volatile storage.

It is reasonable to assert that the key aspects of *fsync()* are unreasonable to test in a test suite. That does not make the function any less valuable, just more difficult to test. A formal conformance test should probably force a system crash (power shutdown) during the test for this condition, but it needs to be done in such a way that automated testing does not require this to be done except when a formal record of the results is being made. It would also not be unreasonable to omit testing for *fsync()*, allowing it to be treated as a quality-of-implementation issue.

FUTURE DIRECTIONS

None.

SEE ALSO

sync()

XBD <unistd.h>

CHANGE HISTORY

First released in Issue 3.

Issue 5

Aligned with *fsync()* in the POSIX Realtime Extension. Specifically, the DESCRIPTION and RETURN VALUE sections are much expanded, and the ERRORS section is updated to indicate that *fsync()* can return the error conditions defined for *read()* and *write()*.

Issue 6

This function is marked as part of the File Synchronization option.

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The [EINVAL] and [EIO] mandatory error conditions are added.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/44 is applied, applying an editorial rewording of the DESCRIPTION. No change in meaning is intended.

32564 **NAME**

32565 ftell, ftello — return a file offset in a stream

32566 **SYNOPSIS**

32567 #include <stdio.h>

32568 long ftell(FILE *stream);

32569 CX off_t ftello(FILE *stream);

32570 **DESCRIPTION**

32571 CX The functionality described on this reference page is aligned with the ISO C standard. Any
 32572 conflict between the requirements described here and the ISO C standard is unintentional. This
 32573 volume of POSIX.1-2008 defers to the ISO C standard.

32574 The *ftell()* function shall obtain the current value of the file-position indicator for the stream
 32575 pointed to by *stream*.

32576 The *ftell()* function shall not change the setting of *errno* if successful. +

32577 CX The *ftello()* function shall be equivalent to *ftell()*, except that the return value is of type *off_t* and
 32578 the *ftello()* function may change the setting of *errno* if successful. |

32579 **RETURN VALUE**

32580 CX Upon successful completion, *ftell()* and *ftello()* shall return the current value of the file-position
 32581 indicator for the stream measured in bytes from the beginning of the file.

32582 Otherwise, *ftell()* and *ftello()* shall return *-1*, and set *errno* to indicate the error. |

32583 **ERRORS**

32584 CX The *ftell()* and *ftello()* functions shall fail if:

32585 CX [EBADF] The file descriptor underlying *stream* is not an open file descriptor.

32586 CX [E_OVERFLOW] For *ftell()*, the current file offset cannot be represented correctly in an object of
 32587 type *long*.

32588 CX [E_OVERFLOW] For *ftello()*, the current file offset cannot be represented correctly in an object
 32589 of type *off_t*.

32590 CX [ESPIPE] The file descriptor underlying *stream* is associated with a pipe, FIFO, or socket. |

32591 **EXAMPLES**

32592 None.

32593 **APPLICATION USAGE**

32594 None.

32595 **RATIONALE**

32596 None.

32597 **FUTURE DIRECTIONS**

32598 None.

32599 **SEE ALSO**32600 Section 2.5 (on page 495), *fgetpos()*, *fopen()*, *fseek()*, *lseek()*

32601 XBD <stdio.h>

CHANGE HISTORY

32602 First released in Issue 1. Derived from Issue 1 of the SVID.
32603

Issue 5

32604 Large File Summit extensions are added.
32605

Issue 6

32606 Extensions beyond the ISO C standard are marked.
32607

32608 The following new requirements on POSIX implementations derive from alignment with the
32609 Single UNIX Specification:

- 32610 • The *ftello()* function is added.
- 32611 • The [EOVERFLOW] error conditions are added.

32612 An additional [ESPIPE] error condition is added for sockets.

Issue 7

32613 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0204 [105], XSH/TC1-2008/0205 [421], +
32614 XSH/TC1-2008/0206 [122], XSH/TC1-2008/0207 [122], and XSH/TC1-2008/0208 [14] are +
32615 applied.
32616

32617 **NAME**

32618 ftok — generate an IPC key

32619 **SYNOPSIS**

```
32620 XSI      #include <sys/ipc.h>
32621         key_t ftok(const char *path, int id);
```

32622 **DESCRIPTION**

32623 The *ftok()* function shall return a key based on *path* and *id* that is usable in subsequent calls to
 32624 *msgget()*, *semget()*, and *shmget()*. The application shall ensure that the *path* argument is the
 32625 pathname of an existing file that the process is able to *stat()*, with the exception that if *stat()* |
 32626 would fail with [EOVERFLOW] due to file size, *ftok()* shall still succeed.

32627 The *ftok()* function shall return the same key value for all paths that name the same file, when
 32628 called with the same *id* value, and should return different key values when called with different |
 32629 *id* values or with paths that name different files existing on the same file system at the same
 32630 time. It is unspecified whether *ftok()* shall return the same key value when called again after the
 32631 file named by *path* is removed and recreated with the same name.

32632 Only the low-order 8-bits of *id* are significant. The behavior of *ftok()* is unspecified if these bits
 32633 are 0.

32634 **RETURN VALUE**

32635 Upon successful completion, *ftok()* shall return a key. Otherwise, *ftok()* shall return (**key_t**)-1
 32636 and set *errno* to indicate the error.

32637 **ERRORS**

32638 The *ftok()* function shall fail if:

32639 [EACCES] Search permission is denied for a component of the path prefix.

32640 [EIO] An error occurred while reading from the file system. +

32641 [ELOOP] A loop exists in symbolic links encountered during resolution of the *path*
 32642 argument.

32643 [ENAMETOOLONG] The length of a component of a pathname is longer than {NAME_MAX}.
 32644

32645 [ENOENT] A component of *path* does not name an existing file or *path* is an empty string.

32646 [ENOTDIR] A component of the path prefix names an existing file that is neither a |
 32647 directory nor a symbolic link to a directory, or the *path* argument contains at
 32648 least one non-*<slash>* character and ends with one or more trailing *<slash>*
 32649 characters and the last pathname component names an existing file that is
 32650 neither a directory nor a symbolic link to a directory.

32651 The *ftok()* function may fail if:

32652 [ELOOP] More than {SYMLOOP_MAX} symbolic links were encountered during
 32653 resolution of the *path* argument.

32654 [ENAMETOOLONG] The length of a pathname exceeds {PATH_MAX}, or pathname resolution of a
 32655 symbolic link produced an intermediate result with a length that exceeds
 32656 {PATH_MAX}.
 32657

EXAMPLES**Getting an IPC Key**

The following example gets a key based on the pathname `/tmp` and the ID value `a`. It also assigns the value of the resulting key to the `semkey` variable so that it will be available to a later call to `semget()`, `msgget()`, or `shmget()`.

```
#include <sys/ipc.h>
...
key_t semkey;

if ((semkey = ftok("/tmp", 'a')) == (key_t) -1) {
    perror("IPC error: ftok"); exit(1);
}
```

APPLICATION USAGE

For maximum portability, *id* should be a single-byte character.

Applications should not assume that the resulting key value is unique.

RATIONALE

None.

FUTURE DIRECTIONS

Future versions of this standard may add new interfaces to provide unique keys.

SEE ALSO

[*msgget\(\)*](#), [*semget\(\)*](#), [*shmget\(\)*](#)

XBD [*<sys/ipc.h>*](#)

CHANGE HISTORY

First released in Issue 4, Version 2.

Issue 5

Moved from X/OPEN UNIX extension to BASE.

Issue 6

The normative text is updated to avoid use of the term “must” for application requirements.

The wording of the mandatory [ELOOP] error condition is updated, and a second optional [ELOOP] error condition is added.

Issue 7

Austin Group Interpretation 1003.1-2001 #143 is applied.

The [ENOTDIR] error condition is clarified to cover the condition where the last component of a pathname exists but is not a directory or a symbolic link to a directory.

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0209 [343], XSH/TC1-2008/0210 [366], XSH/TC1-2008/0211 [343], XSH/TC1-2008/0212 [324], XSH/TC1-2008/0213 [366], XSH/TC1-2008/0214 [366], XSH/TC1-2008/0215 [366], and XSH/TC1-2008/0216 [366] are applied.

32695 **NAME**

32696 ftruncate — truncate a file to a specified length

32697 **SYNOPSIS**

32698 #include <unistd.h>

32699 int ftruncate(int *fildes*, off_t *length*);32700 **DESCRIPTION**32701 If *fildes* is not a valid file descriptor open for writing, the *ftruncate()* function shall fail.

32702 If *fildes* refers to a regular file, the *ftruncate()* function shall cause the size of the file to be truncated to *length*. If the size of the file previously exceeded *length*, the extra data shall no longer be available to reads on the file. If the file previously was smaller than this size, *ftruncate()* shall increase the size of the file. If the file size is increased, the extended area shall appear as if it were zero-filled. The value of the seek pointer shall not be modified by a call to *ftruncate()*.

32708 Upon successful completion, if *fildes* refers to a regular file, *ftruncate()* shall mark for update the last data modification and last file status change timestamps of the file and the S_ISUID and S_ISGID bits of the file mode may be cleared. If the *ftruncate()* function is unsuccessful, the file is unaffected.

32712 XSI If the request would cause the file size to exceed the soft file size limit for the process, the request shall fail and the implementation shall generate the SIGXFSZ signal for the thread.

32714 If *fildes* refers to a directory, *ftruncate()* shall fail.32715 If *fildes* refers to any other file type, except a shared memory object, the result is unspecified.

32716 SHM If *fildes* refers to a shared memory object, *ftruncate()* shall set the size of the shared memory object to *length*.

32718 SHM If the effect of *ftruncate()* is to decrease the size of a memory mapped file or a shared memory object and whole pages beyond the new end were previously mapped, then the whole pages beyond the new end shall be discarded.

32721 References to discarded pages shall result in the generation of a SIGBUS signal.

32722 If the effect of *ftruncate()* is to increase the size of a memory object, it is unspecified whether the contents of any mapped pages between the old end-of-file and the new are flushed to the underlying object.

32725 **RETURN VALUE**

32726 Upon successful completion, *ftruncate()* shall return 0; otherwise, -1 shall be returned and *errno* set to indicate the error.

32728 **ERRORS**32729 The *ftruncate()* function shall fail if:

32730 [EINTR] A signal was caught during execution.

32731 [EINVAL] The *length* argument was less than 0.

32732 [EFBIG] or [EINVAL]

32733 The *length* argument was greater than the maximum file size.

32734 [EFBIG] The file is a regular file and *length* is greater than the offset maximum established in the open file description associated with *fildes*.

32736 [EIO] An I/O error occurred while reading from or writing to a file system.
32737 [EBADF] or [EINVAL]
32738 The *fildest* argument is not a file descriptor open for writing.

32739 EXAMPLES

32740 None.

32741 APPLICATION USAGE

32742 None.

32743 RATIONALE

32744 None.

32745 FUTURE DIRECTIONS

32746 None.

32747 SEE ALSO

32748 *open()*, *truncate()*

32749 XBD <unistd.h>

32750 CHANGE HISTORY

32751 First released in Issue 4, Version 2.

32752 Issue 5

32753 Moved from X/OPEN UNIX extension to BASE and aligned with *ftruncate()* in the POSIX
32754 Realtime Extension. Specifically, the DESCRIPTION is extensively reworded and [EROFS] is
32755 added to the list of mandatory errors that can be returned by *ftruncate()*.

32756 Large File Summit extensions are added.

32757 Issue 6

32758 The *truncate()* function is split out into a separate reference page.

32759 The following new requirements on POSIX implementations derive from alignment with the
32760 Single UNIX Specification:

- 32761 • The DESCRIPTION is changed to indicate that if the file size is changed, and if the file is a
32762 regular file, the S_ISUID and S_ISGID bits in the file mode may be cleared.

32763 The following changes were made to align with the IEEE P1003.1a draft standard:

- 32764 • The DESCRIPTION text is updated.

32765 XSI-conformant systems are required to increase the size of the file if the file was previously
32766 smaller than the size requested.

32767 Issue 7

32768 Austin Group Interpretation 1003.1-2001 #056 is applied, revising the ERRORS section (although
32769 the [EINVAL] “may fail” error was subsequently removed during review of the XSI option).

32770 Functionality relating to the Memory Protection and Memory Mapped Files options is moved to
32771 the Base.

32772 The DESCRIPTION is updated so that a call to *ftruncate()* when the file is smaller than the size
32773 requested will increase the size of the file. Previously, non-XSI-conforming implementations
32774 were allowed to increase the size of the file or fail.

32775 Changes are made related to support for finegrained timestamps.

32776 **NAME**

32777 ftrylockfile — stdio locking functions

32778 **SYNOPSIS**

```
32779 CX      #include <stdio.h>
32780          int ftrylockfile(FILE *file);
```

32781 **DESCRIPTION**32782 Refer to *flockfile()*.

NAME

ftw — traverse (walk) a file tree

SYNOPSIS

```
#include <ftw.h>

int ftw(const char *path, int (*fn)(const char *,
    const struct stat *ptr, int flag), int ndirs);
```

DESCRIPTION

The *ftw()* function shall recursively descend the directory hierarchy rooted in *path*. For each object in the hierarchy, *ftw()* shall call the function pointed to by *fn*, passing it a pointer to a null-terminated character string containing the name of the object, a pointer to a **stat** structure containing information about the object, filled in as if *stat()* or *lstat()* had been called to retrieve the information. Possible values of the integer, defined in the **<ftw.h>** header, are:

FTW_D For a directory.

FTW_DNR For a directory that cannot be read.

FTW_F For a non-directory file.

FTW_SL For a symbolic link (but see also FTW_NS below).

FTW_NS For an object other than a symbolic link on which *stat()* could not successfully be executed. If the object is a symbolic link and *stat()* failed, it is unspecified whether *ftw()* passes FTW_SL or FTW_NS to the user-supplied function.

If the integer is FTW_DNR, descendants of that directory shall not be processed. If the integer is FTW_NS, the **stat** structure contains undefined values. An example of an object that would cause FTW_NS to be passed to the function pointed to by *fn* would be a file in a directory with read but without execute (search) permission.

The *ftw()* function shall visit a directory before visiting any of its descendants.

The *ftw()* function shall use at most one file descriptor for each level in the tree.

The argument *ndirs* should be in the range [1,{OPEN_MAX}].

The tree traversal shall continue until either the tree is exhausted, an invocation of *fn* returns a non-zero value, or some error, other than [EACCES], is detected within *ftw()*.

The *ndirs* argument shall specify the maximum number of directory streams or file descriptors or both available for use by *ftw()* while traversing the tree. When *ftw()* returns it shall close any directory streams and file descriptors it uses not counting any opened by the application-supplied *fn* function.

The results are unspecified if the application-supplied *fn* function does not preserve the current working directory.

The *ftw()* function need not be thread-safe.

RETURN VALUE

If the tree is exhausted, *ftw()* shall return 0. If the function pointed to by *fn* returns a non-zero value, *ftw()* shall stop its tree traversal and return whatever value was returned by the function pointed to by *fn*. If *ftw()* detects an error, it shall return -1 and set *errno* to indicate the error.

If *ftw()* encounters an error other than [EACCES] (see FTW_DNR and FTW_NS above), it shall return -1 and set *errno* to indicate the error. The external variable *errno* may contain any error value that is possible when a directory is opened or when one of the *stat* functions is executed on

32825 a directory or file.

32826 ERRORS

32827 The *ftw()* function shall fail if:

32828 [EACCES] Search permission is denied for any component of *path* or read permission is
32829 denied for *path*.

32830 [ELOOP] A loop exists in symbolic links encountered during resolution of the *path*
32831 argument.

32832 [ENAMETOOLONG]

32833 The length of a component of a pathname is longer than {NAME_MAX}.

32834 [ENOENT] A component of *path* does not name an existing file or *path* is an empty string.

32835 [ENOTDIR] A component of *path* names an existing file that is neither a directory nor a
32836 symbolic link to a directory.

32837 [EOVERFLOW] A field in the **stat** structure cannot be represented correctly in the current
32838 programming environment for one or more files found in the file hierarchy.

32839 The *ftw()* function may fail if:

32840 [EINVAL] The value of the *ndirs* argument is invalid.

32841 [ELOOP] More than {SYMLOOP_MAX} symbolic links were encountered during
32842 resolution of the *path* argument.

32843 [ENAMETOOLONG]

32844 The length of a pathname exceeds {PATH_MAX}, or pathname resolution of a
32845 symbolic link produced an intermediate result with a length that exceeds
32846 {PATH_MAX}.

32847 In addition, if the function pointed to by *fn* encounters system errors, *errno* may be set
32848 accordingly.

32849 EXAMPLES

32850 Walking a Directory Structure

32851 The following example walks the current directory structure, calling the *fn* function for every
32852 directory entry, using at most 10 file descriptors:

```
32853 #include <ftw.h>
32854 ...
32855 if (ftw(".", fn, 10) != 0) {
32856     perror("ftw"); exit(2);
32857 }
```

32858 APPLICATION USAGE

32859 The *ftw()* function may allocate dynamic storage during its operation. If *ftw()* is forcibly
32860 terminated, such as by *longjmp()* or *siglongjmp()* being executed by the function pointed to by *fn*
32861 or an interrupt routine, *ftw()* does not have a chance to free that storage, so it remains
32862 permanently allocated. A safe way to handle interrupts is to store the fact that an interrupt has
32863 occurred, and arrange to have the function pointed to by *fn* return a non-zero value at its next
32864 invocation.

32865 Applications should use the *nftw()* function instead of the obsolescent *ftw()* function.

32866 **RATIONALE**

32867 None.

32868 **FUTURE DIRECTIONS**32869 The *ftw()* function may be removed in a future version.32870 **SEE ALSO**32871 *fdopendir()*, *fstatat()*, *longjmp()*, *nftw()*, *siglongjmp()*

32872 XBD <ftw.h>, <sys/stat.h>

32873 **CHANGE HISTORY**

32874 First released in Issue 1. Derived from Issue 1 of the SVID.

32875 **Issue 5**

32876 UX codings in the DESCRIPTION, RETURN VALUE, and ERRORS sections are changed to EX.

32877 **Issue 6**

32878 The ERRORS section is updated as follows:

- 32879 • The wording of the mandatory [ELOOP] error condition is updated.
- 32880 • A second optional [ELOOP] error condition is added.
- 32881 • The [Eoverflow] mandatory error condition is added.

32882 A note is added to the DESCRIPTION indicating that this function need not be reentrant, and
 32883 that the results are unspecified if the application-supplied *fn* function does not preserve the
 32884 current working directory.

32885 **Issue 7**

32886 Austin Group Interpretations 1003.1-2001 #143 and #156 are applied.

32887 SD5-XBD-ERN-61 is applied.

32888 The *ftw()* function is marked obsolescent. |

32889 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0217 [403], XSH/TC1-2008/0218 [324], |
 32890 and XSH/TC1-2008/0219 [361] are applied.

32891 **NAME**

32892 funlockfile — stdio locking functions

32893 **SYNOPSIS**

```
32894 CX      #include <stdio.h>  
32895      void funlockfile(FILE *file);
```

32896 **DESCRIPTION**32897 Refer to *flockfile()*.

NAME

futimens, utimensat, utimes — set file access and modification times

SYNOPSIS

```
#include <sys/stat.h>

int futimens(int fd, const struct timespec times[2]);
int utimensat(int fd, const char *path, const struct timespec times[2],
              int flag);

#include <sys/time.h>

int utimes(const char *path, const struct timeval times[2]);
```

DESCRIPTION

The *futimens()* and *utimensat()* functions shall set the access and modification times of a file to the values of the *times* argument. The *futimens()* function changes the times of the file associated with the file descriptor *fd*. The *utimensat()* function changes the times of the file pointed to by the *path* argument, relative to the directory associated with the file descriptor *fd*. Both functions allow time specifications accurate to the nanosecond.

For *futimens()* and *utimensat()*, the *times* argument is an array of two **timespec** structures. The first array member represents the date and time of last access, and the second member represents the date and time of last modification. The times in the **timespec** structure are measured in seconds and nanoseconds since the Epoch. The file's relevant timestamp shall be set to the greatest value supported by the file system that is not greater than the specified time.

If the *tv_nsec* field of a **timespec** structure has the special value **UTIME_NOW**, the file's relevant timestamp shall be set to the greatest value supported by the file system that is not greater than the current time. If the *tv_nsec* field has the special value **UTIME_OMIT**, the file's relevant timestamp shall not be changed. In either case, the *tv_sec* field shall be ignored.

If the *times* argument is a null pointer, both the access and modification timestamps shall be set to the greatest value supported by the file system that is not greater than the current time. If *utimensat()* is passed a relative path in the *path* argument, the file to be used shall be relative to the directory associated with the file descriptor *fd* instead of the current working directory. If the file descriptor was opened without **O_SEARCH**, the function shall check whether directory searches are permitted using the current permissions of the directory underlying the file descriptor. If the file descriptor was opened with **O_SEARCH**, the function shall not perform the check.

If *utimensat()* is passed the special value **AT_FDCWD** in the *fd* parameter, the current working directory shall be used.

Only a process with the effective user ID equal to the user ID of the file, or with write access to the file, or with appropriate privileges may use *futimens()* or *utimensat()* with a null pointer as the *times* argument or with both *tv_nsec* fields set to the special value **UTIME_NOW**. Only a process with the effective user ID equal to the user ID of the file or with appropriate privileges may use *futimens()* or *utimensat()* with a non-null *times* argument that does not have both *tv_nsec* fields set to **UTIME_NOW** and does not have both *tv_nsec* fields set to **UTIME_OMIT**. If both *tv_nsec* fields are set to **UTIME_OMIT**, no ownership or permissions check shall be performed for the file, but other error conditions may still be detected (including [EACCES] errors related to the path prefix).

Values for the *flag* argument of *utimensat()* are constructed by a bitwise-inclusive OR of flags from the following list, defined in **<fcntl.h>**:

32943 AT_SYMLINK_NOFOLLOW

32944 If *path* names a symbolic link, then the access and modification times of the symbolic link
32945 are changed.

32946 Upon completion, *futimens()* and *utimensat()* shall mark the last file status change timestamp for
32947 update.

32948 The *utimes()* function shall be equivalent to the *utimensat()* function with the special value
32949 AT_FDCWD as the *fd* argument and the *flag* argument set to zero, except that the *times* argument
32950 is a **timeval** structure rather than a **timespec** structure, and accuracy is only to the microsecond,
32951 not nanosecond, and rounding towards the nearest second may occur.

32952 RETURN VALUE

32953 Upon successful completion, these functions shall return 0. Otherwise, these functions shall
32954 return -1 and set *errno* to indicate the error. If -1 is returned, the file times shall not be affected.

32955 ERRORS

32956 These functions shall fail if:

32957 [EACCES] The *times* argument is a null pointer, or both *tv_nsec* values are UTIME_NOW,
32958 and the effective user ID of the process does not match the owner of the file
32959 and write access is denied.

32960 [EINVAL] Either of the *times* argument structures specified a *tv_nsec* value that was
32961 neither UTIME_NOW nor UTIME_OMIT, and was a value less than zero or
32962 greater than or equal to 1 000 million.

32963 [EINVAL] A new file timestamp would be a value whose *tv_sec* component is not a value
32964 supported by the file system.

32965 [EPERM] The *times* argument is not a null pointer, does not have both *tv_nsec* fields set
32966 to UTIME_NOW, does not have both *tv_nsec* fields set to UTIME_OMIT, the
32967 calling process' effective user ID does not match the owner of the file, and the
32968 calling process does not have appropriate privileges.

32969 [EROFS] The file system containing the file is read-only.

32970 The *futimens()* function shall fail if:

32971 [EBADF] The *fd* argument is not a valid file descriptor.

32972 The *utimensat()* function shall fail if:

32973 [EACCES] *fd* was not opened with O_SEARCH and the permissions of the directory
32974 underlying *fd* do not permit directory searches.

32975 [EBADF] The *path* argument does not specify an absolute path and the *fd* argument is
32976 neither AT_FDCWD nor a valid file descriptor open for reading or searching.

32977 [ENOTDIR] The *path* argument is not an absolute path and *fd* is a file descriptor associated
32978 with a non-directory file.

32979 The *utimensat()* and *utimes()* functions shall fail if:

32980 [EACCES] Search permission is denied by a component of the path prefix.

32981 [ELOOP] A loop exists in symbolic links encountered during resolution of the *path*
32982 argument.

- 32983 [ENAMETOOLONG]
 32984 The length of a component of a pathname is longer than {NAME_MAX}.
- 32985 [ENOENT] A component of *path* does not name an existing file or *path* is an empty string.
- 32986 [ENOTDIR] A component of the path prefix names an existing file that is neither a
 32987 directory nor a symbolic link to a directory, or the *path* argument contains at
 32988 least one non-`<slash>` character and ends with one or more trailing `<slash>`
 32989 characters and the last pathname component names an existing file that is
 32990 neither a directory nor a symbolic link to a directory.
- 32991 The *utimensat()* and *utimes()* functions may fail if:
- 32992 [ELOOP] More than {SYMLOOP_MAX} symbolic links were encountered during
 32993 resolution of the *path* argument.
- 32994 [ENAMETOOLONG]
 32995 The length of a pathname exceeds {PATH_MAX}, or pathname resolution of a
 32996 symbolic link produced an intermediate result with a length that exceeds
 32997 {PATH_MAX}.
- 32998 The *utimensat()* function may fail if:
- 32999 [EINVAL] The value of the *flag* argument is not valid. |

EXAMPLES

33001 None.

APPLICATION USAGE

33003 None.

RATIONALE

33005 The purpose of the *utimensat()* function is to set the access and modification time of files in
 33006 directories other than the current working directory without exposure to race conditions. Any
 33007 part of the path of a file could be changed in parallel to a call to *utimes()*, resulting in unspecified
 33008 behavior. By opening a file descriptor for the target directory and using the *utimensat()* function
 33009 it can be guaranteed that the changed file is located relative to the desired directory.

33010 The standard developers considered including a special case for the permissions required by
 33011 *utimensat()* when one *tv_nsec* field is `UTIME_NOW` and the other is `UTIME_OMIT`. One
 33012 possibility would be to include this case in with the cases where *times* is a null pointer or both
 33013 fields are `UTIME_NOW`, where the call is allowed if the process has write permission for the file.
 33014 However, associating write permission with an update to just the last data access timestamp
 33015 (which is normally updated by *read()*) did not seem appropriate. The other possibility would be
 33016 to specify that this one case is allowed if the process has read permission, but this was felt to be
 33017 too great a departure from the *utime()* and *utimes()* functions on which *utimensat()* is based. If
 33018 an application needs to set the last data access timestamp to the current time for a file on which
 33019 it has read permission but is not the owner, it can do so by opening the file, reading one or more
 33020 bytes (or reading a directory entry, if the file is a directory), and then closing it.

FUTURE DIRECTIONS

33022 None.

SEE ALSO

33024 *read()*, *utime()*

33025 XBD `<fcntl.h>`, `<sys/stat.h>`, `<sys/time.h>`

CHANGE HISTORY

33026 First released in Issue 4, Version 2.

Issue 5

33028 Moved from X/OPEN UNIX extension to BASE.

Issue 6

33030 This function is marked LEGACY.

33031 The normative text is updated to avoid use of the term “must” for application requirements.

33032 The wording of the mandatory [ELOOP] error condition is updated, and a second optional [ELOOP] error condition is added.

Issue 7

33033 Austin Group Interpretation 1003.1-2001 #143 is applied.

33034 The LEGACY marking is removed.

33035 The *utimensat()* function (renamed from *futimesat()*) is added from The Open Group Technical Standard, 2006, Extended API Set Part 2, and changed to allow modifying a symbolic link by adding a *flag* argument.

33036 The *futimens()* function is added.

33037 Changes are made related to support for finegrained timestamps.

33038 Changes are made to allow a directory to be opened for searching.

33039 The [ENOTDIR] error condition is clarified to cover the condition where the last component of a pathname exists but is not a directory or a symbolic link to a directory.

33040 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0220 [63,428], XSH/TC1-2008/0221 + [278], XSH/TC1-2008/0222 [324], XSH/TC1-2008/0223 [306], and XSH/TC1-2008/0224 [278] are + applied.

33049 **NAME**

33050 fwide — set stream orientation

33051 **SYNOPSIS**

33052 #include <stdio.h>

33053 #include <wchar.h>

33054 int fwide(FILE **stream*, int *mode*);33055 **DESCRIPTION**

33056 CX The functionality described on this reference page is aligned with the ISO C standard. Any
 33057 conflict between the requirements described here and the ISO C standard is unintentional. This
 33058 volume of POSIX.1-2008 defers to the ISO C standard.

33059 The *fwide()* function shall determine the orientation of the stream pointed to by *stream*. If *mode* is
 33060 greater than zero, the function first attempts to make the stream wide-oriented. If *mode* is less
 33061 than zero, the function first attempts to make the stream byte-oriented. Otherwise, *mode* is zero
 33062 and the function does not alter the orientation of the stream.

33063 If the orientation of the stream has already been determined, *fwide()* shall not change it.

33064 CX The *fwide()* function shall not change the setting of *errno* if successful. +

33065 Since no return value is reserved to indicate an error, an application wishing to check for error
 33066 situations should set *errno* to 0, then call *fwide()*, then check *errno*, and if it is non-zero, assume
 33067 an error has occurred.

33068 **RETURN VALUE**

33069 The *fwide()* function shall return a value greater than zero if, after the call, the stream has wide-
 33070 orientation, a value less than zero if the stream has byte-orientation, or zero if the stream has no
 33071 orientation.

33072 **ERRORS**

33073 The *fwide()* function may fail if:

33074 CX [EBADF] The *stream* argument is not a valid stream.

33075 **EXAMPLES**

33076 None.

33077 **APPLICATION USAGE**

33078 A call to *fwide()* with *mode* set to zero can be used to determine the current orientation of a
 33079 stream.

33080 **RATIONALE**

33081 None.

33082 **FUTURE DIRECTIONS**

33083 None.

33084 **SEE ALSO**

33085 XBD <stdio.h>, <wchar.h>

33086 **CHANGE HISTORY**

33087 First released in Issue 5. Included for alignment with ISO/IEC 9899:1990/Amendment 1:1995
 33088 (E).

33089 **Issue 6**

33090 Extensions beyond the ISO C standard are marked.

33091 **Issue 7**

33092 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0225 [272] is applied.

+

33093 NAME

33094 fwprintf, swprintf, wprintf — print formatted wide-character output

33095 SYNOPSIS

```

33096 #include <stdio.h>
33097 #include <wchar.h>
33098 int fwprintf(FILE *restrict stream, const wchar_t *restrict format, ...);
33099 int swprintf(wchar_t *restrict ws, size_t n,
33100             const wchar_t *restrict format, ...);
33101 int wprintf(const wchar_t *restrict format, ...);

```

33102 DESCRIPTION

33103 CX The functionality described on this reference page is aligned with the ISO C standard. Any
 33104 conflict between the requirements described here and the ISO C standard is unintentional. This
 33105 volume of POSIX.1-2008 defers to the ISO C standard.

33106 The *fwprintf()* function shall place output on the named output *stream*. The *wprintf()* function
 33107 shall place output on the standard output stream *stdout*. The *swprintf()* function shall place
 33108 output followed by the null wide character in consecutive wide characters starting at **ws*; no
 33109 more than *n* wide characters shall be written, including a terminating null wide character, which
 33110 is always added (unless *n* is zero).

33111 Each of these functions shall convert, format, and print its arguments under control of the *format*
 33112 wide-character string. The *format* is composed of zero or more directives: *ordinary wide-characters*,
 33113 which are simply copied to the output stream, and *conversion specifications*, each of which results
 33114 in the fetching of zero or more arguments. The results are undefined if there are insufficient
 33115 arguments for the *format*. If the *format* is exhausted while arguments remain, the excess
 33116 arguments are evaluated but are otherwise ignored.

33117 CX Conversions can be applied to the *n*th argument after the *format* in the argument list, rather than
 33118 to the next unused argument. In this case, the conversion specifier wide character % (see below)
 33119 is replaced by the sequence "%*n*\$", where *n* is a decimal integer in the range
 33120 [1,{NL_ARGMAX}], giving the position of the argument in the argument list. This feature
 33121 provides for the definition of *format* wide-character strings that select arguments in an order
 33122 appropriate to specific languages (see the EXAMPLES section).

33123 The *format* can contain either numbered argument specifications (that is, "%*n*\$" and "%*m*\$"), or
 33124 unnumbered argument conversion specifications (that is, % and *), but not both. The only
 33125 exception to this is that %% can be mixed with the "%*n*\$" form. The results of mixing numbered
 33126 and unnumbered argument specifications in a *format* wide-character string are undefined. When
 33127 numbered argument specifications are used, specifying the *N*th argument requires that all the
 33128 leading arguments, from the first to the (*N*–1)th, are specified in the *format* wide-character string.

33129 In *format* wide-character strings containing the "%*n*\$" form of conversion specification,
 33130 numbered arguments in the argument list can be referenced from the *format* wide-character
 33131 string as many times as required.

33132 In *format* wide-character strings containing the % form of conversion specification, each
 33133 argument in the argument list shall be used exactly once.

33134 CX All forms of the *fwprintf()* function allow for the insertion of a locale-dependent radix character
 33135 in the output string, output as a wide-character value. The radix character is defined in the
 33136 current locale (category LC_NUMERIC). In the POSIX locale, or in a locale where the radix
 33137 character is not defined, the radix character shall default to a <period> (' . ').

Each conversion specification is introduced by the `' % '` wide character or by the wide-character sequence `"%n$"`, after which the following appear in sequence:

- Zero or more *flags* (in any order), which modify the meaning of the conversion specification.
- An optional minimum *field width*. If the converted value has fewer wide characters than the field width, it shall be padded with `<space>` characters by default on the left; it shall be padded on the right, if the left-adjustment flag (`' - '`), described below, is given to the field width. The field width takes the form of an `<asterisk>` (`' * '`), described below, or a decimal integer.
- An optional *precision* that gives the minimum number of digits to appear for the `d`, `i`, `o`, `u`, `x`, and `X` conversion specifiers; the number of digits to appear after the radix character for the `a`, `A`, `e`, `E`, `f`, and `F` conversion specifiers; the maximum number of significant digits for the `g` and `G` conversion specifiers; or the maximum number of wide characters to be printed from a string in the `s` conversion specifiers. The precision takes the form of a `<period>` (`' . '`) followed either by an `<asterisk>` (`' * '`), described below, or an optional decimal digit string, where a null digit string is treated as 0. If a precision appears with any other conversion wide character, the behavior is undefined.
- An optional length modifier that specifies the size of the argument.
- A *conversion specifier* wide character that indicates the type of conversion to be applied.

A field width, or precision, or both, may be indicated by an `<asterisk>` (`' * '`). In this case an argument of type `int` supplies the field width or precision. Applications shall ensure that arguments specifying field width, or precision, or both appear in that order before the argument, if any, to be converted. A negative field width is taken as a `' - '` flag followed by a positive field width. A negative precision is taken as if the precision were omitted. In *format* wide-character strings containing the `"%n$"` form of a conversion specification, a field width or precision may be indicated by the sequence `"*m$"`, where `m` is a decimal integer in the range `[1, {NL_ARGMAX}]` giving the position in the argument list (after the *format* argument) of an integer argument containing the field width or precision, for example:

```
wprintf(L"%1$d:%2$.*3$d:%4$.*3$d\n", hour, min, precision, sec);
```

The flag wide characters and their meanings are:

- `'` (The `<apostrophe>`.) The integer portion of the result of a decimal conversion (`%i`, `%d`, `%u`, `%f`, `%F`, `%g`, or `%G`) shall be formatted with thousands' grouping wide characters. For other conversions, the behavior is undefined. The numeric grouping wide character is used.
- `-` The result of the conversion shall be left-justified within the field. The conversion shall be right-justified if this flag is not specified.
- `+` The result of a signed conversion shall always begin with a sign (`' + '` or `' - '`). The conversion shall begin with a sign only when a negative value is converted if this flag is not specified.
- `<space>` If the first wide character of a signed conversion is not a sign, or if a signed conversion results in no wide characters, a `<space>` shall be prefixed to the result. This means that if the `<space>` and `' + '` flags both appear, the `<space>` flag shall be ignored.
- `#` Specifies that the value is to be converted to an alternative form. For `o` conversion, it increases the precision (if necessary) to force the first digit of the result to be 0. For `x` or `X` conversion specifiers, a non-zero result shall have `0x` (or `0X`) prefixed to it. For `a`, `A`, `e`,

33183 E, f, F, g, and G conversion specifiers, the result shall always contain a radix character,
 33184 even if no digits follow it. Without this flag, a radix character appears in the result of
 33185 these conversions only if a digit follows it. For g and G conversion specifiers, trailing
 33186 zeros shall *not* be removed from the result as they normally are. For other conversion
 33187 specifiers, the behavior is undefined.

33188 0 For d, i, o, u, x, X, a, A, e, E, f, F, g, and G conversion specifiers, leading zeros
 33189 (following any indication of sign or base) are used to pad to the field width rather than
 33190 performing space padding, except when converting an infinity or NaN. If the '0' and
 33191 '-' flags both appear, the '0' flag shall be ignored. For d, i, o, u, x, and X conversion
 33192 CX specifiers, if a precision is specified, the '0' flag shall be ignored. If the '0' and
 33193 <apostrophe> flags both appear, the grouping wide characters are inserted before zero
 33194 padding. For other conversions, the behavior is undefined.

33195 The length modifiers and their meanings are:

33196 hh Specifies that a following d, i, o, u, x, or X conversion specifier applies to a **signed char**
 33197 or **unsigned char** argument (the argument will have been promoted according to the
 33198 integer promotions, but its value shall be converted to **signed char** or **unsigned char**
 33199 before printing); or that a following n conversion specifier applies to a pointer to a
 33200 **signed char** argument.

33201 h Specifies that a following d, i, o, u, x, or X conversion specifier applies to a **short** or
 33202 **unsigned short** argument (the argument will have been promoted according to the
 33203 integer promotions, but its value shall be converted to **short** or **unsigned short** before
 33204 printing); or that a following n conversion specifier applies to a pointer to a **short**
 33205 argument.

33206 l (ell) Specifies that a following d, i, o, u, x, or X conversion specifier applies to a **long** or
 33207 **unsigned long** argument; that a following n conversion specifier applies to a pointer to
 33208 a **long** argument; that a following c conversion specifier applies to a **wint_t** argument;
 33209 that a following s conversion specifier applies to a pointer to a **wchar_t** argument; or
 33210 has no effect on a following a, A, e, E, f, F, g, or G conversion specifier.

33211 ll (ell-ell) Specifies that a following d, i, o, u, x, or X conversion specifier applies to a **long long**
 33212 or **unsigned long long** argument; or that a following n conversion specifier applies to a
 33213 pointer to a **long long** argument.

33215 j Specifies that a following d, i, o, u, x, or X conversion specifier applies to an **intmax_t**
 33216 or **uintmax_t** argument; or that a following n conversion specifier applies to a pointer
 33217 to an **intmax_t** argument.

33218 z Specifies that a following d, i, o, u, x, or X conversion specifier applies to a **size_t** or the
 33219 corresponding signed integer type argument; or that a following n conversion specifier
 33220 applies to a pointer to a signed integer type corresponding to a **size_t** argument.

33221 t Specifies that a following d, i, o, u, x, or X conversion specifier applies to a **ptrdiff_t** or
 33222 the corresponding **unsigned** type argument; or that a following n conversion specifier
 33223 applies to a pointer to a **ptrdiff_t** argument.

33224 L Specifies that a following a, A, e, E, f, F, g, or G conversion specifier applies to a **long**
 33225 **double** argument.

33226		If a length modifier appears with any conversion specifier other than as specified above, the
33227		behavior is undefined.
33228		The conversion specifiers and their meanings are:
33229	d, i	The int argument shall be converted to a signed decimal in the style "[−] dddd". The
33230		precision specifies the minimum number of digits to appear; if the value being
33231		converted can be represented in fewer digits, it shall be expanded with leading zeros.
33232		The default precision shall be 1. The result of converting zero with an explicit precision
33233		of zero shall be no wide characters.
33234	o	The unsigned argument shall be converted to unsigned octal format in the style
33235		"dddd". The precision specifies the minimum number of digits to appear; if the value
33236		being converted can be represented in fewer digits, it shall be expanded with leading
33237		zeros. The default precision shall be 1. The result of converting zero with an explicit
33238		precision of zero shall be no wide characters.
33239	u	The unsigned argument shall be converted to unsigned decimal format in the style
33240		"dddd". The precision specifies the minimum number of digits to appear; if the value
33241		being converted can be represented in fewer digits, it shall be expanded with leading
33242		zeros. The default precision shall be 1. The result of converting zero with an explicit
33243		precision of zero shall be no wide characters.
33244	x	The unsigned argument shall be converted to unsigned hexadecimal format in the style
33245		"dddd"; the letters "abcdef" are used. The precision specifies the minimum number
33246		of digits to appear; if the value being converted can be represented in fewer digits, it
33247		shall be expanded with leading zeros. The default precision shall be 1. The result of
33248		converting zero with an explicit precision of zero shall be no wide characters.
33249	X	Equivalent to the x conversion specifier, except that letters "ABCDEF" are used instead
33250		of "abcdef".
33251	f, F	The double argument shall be converted to decimal notation in the style
33252		"[−] ddd.ddd", where the number of digits after the radix character shall be equal to
33253		the precision specification. If the precision is missing, it shall be taken as 6; if the
33254		precision is explicitly zero and no ' # ' flag is present, no radix character shall appear. If
33255		a radix character appears, at least one digit shall appear before it. The value shall be
33256		rounded in an implementation-defined manner to the appropriate number of digits.
33257		A double argument representing an infinity shall be converted in one of the styles
33258		"[−]inf" or "[−]infinity"; which style is implementation-defined. A double
33259		argument representing a NaN shall be converted in one of the styles "[−]nan" or
33260		"[−]nan(n-char-sequence)"; which style, and the meaning of any <i>n-char-sequence</i> ,
33261		is implementation-defined. The F conversion specifier produces "INF", "INFINITY",
33262		or "NAN" instead of "inf", "infinity", or "nan", respectively.
33263	e, E	The double argument shall be converted in the style "[−] d.ddde±dd", where there
33264		shall be one digit before the radix character (which is non-zero if the argument is non-
33265		zero) and the number of digits after it shall be equal to the precision; if the precision is
33266		missing, it shall be taken as 6; if the precision is zero and no ' # ' flag is present, no
33267		radix character shall appear. The value shall be rounded in an implementation-defined
33268		manner to the appropriate number of digits. The E conversion wide character shall
33269		produce a number with 'E' instead of 'e' introducing the exponent. The exponent
33270		shall always contain at least two digits. If the value is zero, the exponent shall be zero.
33271		A double argument representing an infinity or NaN shall be converted in the style of
33272		an f or F conversion specifier.

33273	g, G	The double argument representing a floating-point number shall be converted in the style <code>f</code> or <code>e</code> (or in the style <code>F</code> or <code>E</code> in the case of a <code>G</code> conversion specifier), depending on the value converted and the precision. Let <code>P</code> equal the precision if non-zero, 6 if the precision is omitted, or 1 if the precision is zero. Then, if a conversion with style <code>E</code> would have an exponent of <code>X</code> :
33274		
33275		
33276		
33277		
33278		— If $P > X \geq -4$, the conversion shall be with style <code>f</code> (or <code>F</code>) and precision $P - (X + 1)$.
33279		— Otherwise, the conversion shall be with style <code>e</code> (or <code>E</code>) and precision $P - 1$.
33280		Finally, unless the <code>'#'</code> flag is used, any trailing zeros shall be removed from the fractional portion of the result and the decimal-point character shall be removed if there is no fractional portion remaining.
33281		
33282		
33283		A double argument representing an infinity or NaN shall be converted in the style of an <code>f</code> or <code>F</code> conversion specifier.
33284		
33285	a, A	A double argument representing a floating-point number shall be converted in the style <code>"[-]0xh.hhhhp±d"</code> , where there shall be one hexadecimal digit (which is non-zero if the argument is a normalized floating-point number and is otherwise unspecified) before the decimal-point wide character and the number of hexadecimal digits after it shall be equal to the precision; if the precision is missing and <code>FLT_RADIX</code> is a power of 2, then the precision shall be sufficient for an exact representation of the value; if the precision is missing and <code>FLT_RADIX</code> is not a power of 2, then the precision shall be sufficient to distinguish values of type double , except that trailing zeros may be omitted; if the precision is zero and the <code>'#'</code> flag is not specified, no decimal-point wide character shall appear. The letters <code>"abcdef"</code> are used for a conversion and the letters <code>"ABCDEF"</code> for A conversion. The A conversion specifier produces a number with <code>'X'</code> and <code>'P'</code> instead of <code>'x'</code> and <code>'p'</code> . The exponent shall always contain at least one digit, and only as many more digits as necessary to represent the decimal exponent of 2. If the value is zero, the exponent shall be zero.
33286		
33287		
33288		
33289		
33290		
33291		
33292		
33293		
33294		
33295		
33296		
33297		
33298		
33299		A double argument representing an infinity or NaN shall be converted in the style of an <code>f</code> or <code>F</code> conversion specifier.
33300		
33301	c	If no <code>l</code> (ell) qualifier is present, the int argument shall be converted to a wide character as if by calling the <code>btowc()</code> function and the resulting wide character shall be written. Otherwise, the wint_t argument shall be converted to wchar_t , and written.
33302		
33303		
33304	s	If no <code>l</code> (ell) qualifier is present, the application shall ensure that the argument is a pointer to a character array containing a character sequence beginning in the initial shift state. Characters from the array shall be converted as if by repeated calls to the <code>mbrtowc()</code> function, with the conversion state described by an mbstate_t object initialized to zero before the first character is converted, and written up to (but not including) the terminating null wide character. If the precision is specified, no more than that many wide characters shall be written. If the precision is not specified, or is greater than the size of the array, the application shall ensure that the array contains a null wide character.
33305		
33306		
33307		
33308		
33309		
33310		
33311		
33312		
33313		If an <code>l</code> (ell) qualifier is present, the application shall ensure that the argument is a pointer to an array of type wchar_t . Wide characters from the array shall be written up to (but not including) a terminating null wide character. If no precision is specified, or is greater than the size of the array, the application shall ensure that the array contains a null wide character. If a precision is specified, no more than that many wide characters shall be written.
33314		
33315		
33316		
33317		
33318		

33319	p	The application shall ensure that the argument is a pointer to void . The value of the
33320		pointer shall be converted to a sequence of printable wide characters in an
33321		implementation-defined manner.
33322	n	The application shall ensure that the argument is a pointer to an integer into which is
33323		written the number of wide characters written to the output so far by this call to one of
33324		the <i>fwprintf()</i> functions. No argument shall be converted, but one shall be consumed. If
33325		the conversion specification includes any flags, a field width, or a precision, the
33326		behavior is undefined.
33327	XSI	C Equivalent to <code>lc</code> .
33328	XSI	S Equivalent to <code>ls</code> .
33329	%	Output a ' % ' wide character; no argument shall be converted. The entire conversion
33330		specification shall be %%.
33331		If a conversion specification does not match one of the above forms, the behavior is undefined.
33332		In no case does a nonexistent or small field width cause truncation of a field; if the result of a
33333		conversion is wider than the field width, the field shall be expanded to contain the conversion
33334		result. Characters generated by <i>fwprintf()</i> and <i>wprintf()</i> shall be printed as if <i>fputwc()</i> had been
33335		called.
33336		For a and A conversions, if FLT_RADIX is not a power of 2 and the result is not exactly
33337		representable in the given precision, the result should be one of the two adjacent numbers in
33338		hexadecimal floating style with the given precision, with the extra stipulation that the error
33339		should have a correct sign for the current rounding direction.
33340		For e, E, f, F, g, and G conversion specifiers, if the number of significant decimal digits is at
33341		most DECIMAL_DIG, then the result should be correctly rounded. If the number of significant
33342		decimal digits is more than DECIMAL_DIG but the source value is exactly representable with
33343		DECIMAL_DIG digits, then the result should be an exact representation with trailing zeros.
33344		Otherwise, the source value is bounded by two adjacent decimal strings $L < U$, both having
33345		DECIMAL_DIG significant digits; the value of the resultant decimal string D should satisfy $L \leq$
33346		$D \leq U$, with the extra stipulation that the error should have a correct sign for the current
33347		rounding direction.
33348	CX	The last data modification and last file status change timestamps of the file shall be marked for
33349		update between the call to a successful execution of <i>fwprintf()</i> or <i>wprintf()</i> and the next
33350		successful completion of a call to <i>fflush()</i> or <i>fclose()</i> on the same stream, or a call to <i>exit()</i> or
33351		<i>abort()</i> .
33352		RETURN VALUE
33353		Upon successful completion, these functions shall return the number of wide characters
33354		transmitted, excluding the terminating null wide character in the case of <i>swprintf()</i> , or a negative
33355	CX	value if an output error was encountered, and set <i>errno</i> to indicate the error.
33356		If <i>n</i> or more wide characters were requested to be written, <i>swprintf()</i> shall return a negative
33357	CX	value, and set <i>errno</i> to indicate the error.
33358		ERRORS
33359		For the conditions under which <i>fwprintf()</i> and <i>wprintf()</i> fail and may fail, refer to <i>fputwc()</i> .
33360		In addition, all forms of <i>fwprintf()</i> shall fail if:
33361	CX	[EILSEQ] A wide-character code that does not correspond to a valid character has been
33362		detected.

33363 In addition, all forms of *fwprintf()* may fail if:

33364 CX [EINVAL] There are insufficient arguments.

33365 In addition, *fwprintf()* and *wprintf()* may fail if:

33366 CX [ENOMEM] Insufficient storage space is available.

33367 The *swprintf()* shall fail if:

33368 CX [EOVERFLOW] The value of *n* is greater than {INT_MAX} or the number of bytes needed to
33369 hold the output excluding the terminating null is greater than {INT_MAX}.

33370 EXAMPLES

33371 To print the language-independent date and time format, the following statement could be used:

33372 `wprintf(format, weekday, month, day, hour, min);`

33373 For American usage, *format* could be a pointer to the wide-character string:

33374 `L"%s, %s %d, %d:%.2d\n"`

33375 producing the message:

33376 `Sunday, July 3, 10:02`

33377 whereas for German usage, *format* could be a pointer to the wide-character string:

33378 `L"%1$s, %3$d. %2$s, %4$d:%5$.2d\n"`

33379 producing the message:

33380 `Sonntag, 3. Juli, 10:02`

33381 APPLICATION USAGE

33382 None.

33383 RATIONALE

33384 None.

33385 FUTURE DIRECTIONS

33386 None.

33387 SEE ALSO

33388 [Section 2.5](#) (on page 495), [btowc\(\)](#), [fputwc\(\)](#), [fwscanf\(\)](#), [mbrtowc\(\)](#), [setlocale\(\)](#)

33389 [XBD Chapter 7](#) (on page 135), [<stdio.h>](#), [<wchar.h>](#)

33390 CHANGE HISTORY

33391 First released in Issue 5. Included for alignment with ISO/IEC 9899:1990/Amendment 1:1995
33392 (E).

33393 Issue 6

33394 The Open Group Corrigendum U040/1 is applied to the RETURN VALUE section, describing
33395 the case if *n* or more wide characters are requested to be written using *swprintf()*.

33396 The normative text is updated to avoid use of the term “must” for application requirements.

33397 The following changes are made for alignment with the ISO/IEC 9899:1999 standard:

- 33398 • The prototypes for *fwprintf()*, *swprintf()*, and *wprintf()* are updated.
- 33399 • The DESCRIPTION is updated.

33400	• The <code>hh</code>, <code>ll</code>, <code>j</code>, <code>t</code>, and <code>z</code> length modifiers are added.
33401	• The <code>a</code>, <code>A</code>, and <code>F</code> conversion characters are added.
33402	• XSI shading is removed from the description of character string representations of infinity and NaN floating-point values.
33403	
33404	The DESCRIPTION is updated to use the terms “conversion specifier” and “conversion specification” consistently.
33405	
33406	ISO/IEC 9899:1999 standard, Technical Corrigendum 1 is incorporated.
33407	Issue 7
33408	Austin Group Interpretation 1003.1-2001 #161 is applied, updating the DESCRIPTION of the <code>0</code> flag.
33409	
33410	Austin Group Interpretation 1003.1-2001 #170 is applied.
33411	ISO/IEC 9899:1999 standard, Technical Corrigendum 2 #68 (SD5-XSH-ERN-70) is applied, revising the description of <code>g</code> and <code>G</code> .
33412	
33413	Functionality relating to the “ <code>%n\$</code> ” form of conversion specification and the <code><apostrophe></code> flag is moved from the XSI option to the Base.
33414	
33415	The <code>[Eoverflow]</code> error is added for <code>swprintf()</code> .
33416	Changes are made related to support for finegrained timestamps.
33417	POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0226 [302] and XSH/TC1-2008/0227 + [14] are applied.
33418	

NAME

fwrite — binary output

SYNOPSIS

#include <stdio.h>

```
size_t fwrite(const void *restrict ptr, size_t size, size_t nitems,
FILE *restrict stream);
```

DESCRIPTION

CX The functionality described on this reference page is aligned with the ISO C standard. Any conflict between the requirements described here and the ISO C standard is unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

The *fwrite()* function shall write, from the array pointed to by *ptr*, up to *nitems* elements whose size is specified by *size*, to the stream pointed to by *stream*. For each object, *size* calls shall be made to the *fputc()* function, taking the values (in order) from an array of **unsigned char** exactly overlaying the object. The file-position indicator for the stream (if defined) shall be advanced by the number of bytes successfully written. If an error occurs, the resulting value of the file-position indicator for the stream is unspecified.

CX The last data modification and last file status change timestamps of the file shall be marked for update between the successful execution of *fwrite()* and the next successful completion of a call to *fflush()* or *fclose()* on the same stream, or a call to *exit()* or *abort()*.

RETURN VALUE

The *fwrite()* function shall return the number of elements successfully written, which may be less than *nitems* if a write error is encountered. If *size* or *nitems* is 0, *fwrite()* shall return 0 and the state of the stream remains unchanged. Otherwise, if a write error occurs, the error indicator for the stream shall be set, and *errno* shall be set to indicate the error.

ERRORSRefer to *fputc()*.**EXAMPLES**

None.

APPLICATION USAGE

Because of possible differences in element length and byte ordering, files written using *fwrite()* are application-dependent, and possibly cannot be read using *fread()* by a different application or by the same application on a different processor.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSOSection 2.5 (on page 495), *ferror()*, *fopen()*, *fprintf()*, *putc()*, *puts()*, *write()*

XBD <stdio.h>

CHANGE HISTORY

First released in Issue 1. Derived from Issue 1 of the SVID.

33460 **Issue 6**

33461 Extensions beyond the ISO C standard are marked.

33462 The following changes are made for alignment with the ISO/IEC 9899:1999 standard:

- 33463 • The *fwrite()* prototype is updated.
- 33464 • The DESCRIPTION is updated to clarify how the data is written out using *fputc()*.

33465 **Issue 7**

33466 Changes are made related to support for finegrained timestamps.

33467 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0228 [14] is applied.

+

NAME

fwscanf, swscanf, wscanf — convert formatted wide-character input

SYNOPSIS

```
#include <stdio.h>
#include <wchar.h>

int fwscanf(FILE *restrict stream, const wchar_t *restrict format, ...);
int swscanf(const wchar_t *restrict ws,
            const wchar_t *restrict format, ...);
int wscanf(const wchar_t *restrict format, ...);
```

DESCRIPTION

The functionality described on this reference page is aligned with the ISO C standard. Any conflict between the requirements described here and the ISO C standard is unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

The *fwscanf()* function shall read from the named input *stream*. The *wscanf()* function shall read from the standard input stream *stdin*. The *swscanf()* function shall read from the wide-character string *ws*. Each function reads wide characters, interprets them according to a format, and stores the results in its arguments. Each expects, as arguments, a control wide-character string *format* described below, and a set of *pointer* arguments indicating where the converted input should be stored. The result is undefined if there are insufficient arguments for the format. If the *format* is exhausted while arguments remain, the excess arguments are evaluated but are otherwise ignored.

Conversions can be applied to the *n*th argument after the *format* in the argument list, rather than to the next unused argument. In this case, the conversion specifier wide character % (see below) is replaced by the sequence "%n\$", where *n* is a decimal integer in the range [1,{NL_ARGMAX}]. This feature provides for the definition of *format* wide-character strings that select arguments in an order appropriate to specific languages. In *format* wide-character strings containing the "%n\$" form of conversion specifications, it is unspecified whether numbered arguments in the argument list can be referenced from the *format* wide-character string more than once.

The *format* can contain either form of a conversion specification—that is, % or "%n\$" — but the two forms cannot normally be mixed within a single *format* wide-character string. The only exception to this is that %% or %* can be mixed with the "%n\$" form. When numbered argument specifications are used, specifying the *N*th argument requires that all the leading arguments, from the first to the (*N*−1)th, are pointers.

The *fwscanf()* function in all its forms allows for detection of a language-dependent radix character in the input string, encoded as a wide-character value. The radix character is defined in the current locale (category *LC_NUMERIC*). In the POSIX locale, or in a locale where the radix character is not defined, the radix character shall default to a <period> ('.').

The *format* is a wide-character string composed of zero or more directives. Each directive is composed of one of the following: one or more white-space wide characters (<space>, <tab>, <newline>, <vertical-tab>, or <form-feed>); an ordinary wide character (neither '%' nor a white-space character); or a conversion specification.

Each conversion specification is introduced by the '%' or by the character sequence "%n\$", after which the following appear in sequence:

- An optional assignment-suppressing character '*'.

33513		• An optional non-zero decimal integer that specifies the maximum field width.
33514	CX	• An optional assignment-allocation character 'm'.
33515		• An optional length modifier that specifies the size of the receiving object.
33516		• A conversion specifier wide character that specifies the type of conversion to be applied.
33517		The valid conversion specifiers are described below.
33518		The <i>fwscanf()</i> functions shall execute each directive of the format in turn. If a directive fails, as
33519		detailed below, the function shall return. Failures are described as input failures (due to the
33520		unavailability of input bytes) or matching failures (due to inappropriate input).
33521		A directive composed of one or more white-space wide characters is executed by reading input
33522		until no more valid input can be read, or up to the first wide character which is not a white-
33523		space wide character, which remains unread.
33524		A directive that is an ordinary wide character shall be executed as follows. The next wide
33525		character is read from the input and compared with the wide character that comprises the
33526		directive; if the comparison shows that they are not equivalent, the directive shall fail, and the
33527		differing and subsequent wide characters remain unread. Similarly, if end-of-file, an encoding
33528		error, or a read error prevents a wide character from being read, the directive shall fail.
33529		A directive that is a conversion specification defines a set of matching input sequences, as
33530		described below for each conversion wide character. A conversion specification is executed in
33531		the following steps.
33532		Input white-space wide characters (as specified by <i>iswspace()</i>) shall be skipped, unless the
33533		conversion specification includes a [, c, or n conversion specifier.
33534		An item shall be read from the input, unless the conversion specification includes an n
33535		conversion specifier wide character. An input item is defined as the longest sequence of input
33536		wide characters, not exceeding any specified field width, which is an initial subsequence of a
33537		matching sequence. The first wide character, if any, after the input item shall remain unread. If
33538		the length of the input item is zero, the execution of the conversion specification shall fail; this
33539		condition is a matching failure, unless end-of-file, an encoding error, or a read error prevented
33540		input from the stream, in which case it is an input failure.
33541		Except in the case of a % conversion specifier, the input item (or, in the case of a %n conversion
33542		specification, the count of input wide characters) shall be converted to a type appropriate to the
33543		conversion wide character. If the input item is not a matching sequence, the execution of the
33544		conversion specification shall fail; this condition is a matching failure. Unless assignment
33545		suppression was indicated by a '*', the result of the conversion shall be placed in the object
33546		pointed to by the first argument following the <i>format</i> argument that has not already received a
33547	CX	conversion result if the conversion specification is introduced by %, or in the <i>n</i> th argument if
33548		introduced by the wide-character sequence "%n\$". If this object does not have an appropriate
33549		type, or if the result of the conversion cannot be represented in the space provided, the behavior
33550		is undefined.
33551	CX	The %c, %s, and %[conversion specifiers shall accept an optional assignment-allocation
33552		character 'm', which shall cause a memory buffer to be allocated to hold the wide-character
33553		string converted including a terminating null wide character. In such a case, the argument
33554		corresponding to the conversion specifier should be a reference to a pointer value that will
33555		receive a pointer to the allocated buffer. The system shall allocate a buffer as if <i>malloc()</i> had been
33556		called. The application shall be responsible for freeing the memory after usage. If there is
33557		insufficient memory to allocate a buffer, the function shall set <i>errno</i> to [ENOMEM] and a
33558		conversion error shall result. If the function returns EOF, any memory successfully allocated for

parameters using assignment-allocation character 'm' by this call shall be freed before the function returns.

The length modifiers and their meanings are:

hh Specifies that a following d, i, o, u, x, X, or n conversion specifier applies to an argument with type pointer to **signed char** or **unsigned char**.

h Specifies that a following d, i, o, u, x, X, or n conversion specifier applies to an argument with type pointer to **short** or **unsigned short**.

l (ell) Specifies that a following d, i, o, u, x, X, or n conversion specifier applies to an argument with type pointer to **long** or **unsigned long**; that a following a, A, e, E, f, F, g, or G conversion specifier applies to an argument with type pointer to **double**; or that a following c, s, or [conversion specifier applies to an argument with type pointer to **wchar_t**. If the 'm' assignment-allocation character is specified, the conversion applies to an argument with the type pointer to a pointer to **wchar_t**.

ll (ell-ell) Specifies that a following d, i, o, u, x, X, or n conversion specifier applies to an argument with type pointer to **long long** or **unsigned long long**.

j Specifies that a following d, i, o, u, x, X, or n conversion specifier applies to an argument with type pointer to **intmax_t** or **uintmax_t**.

z Specifies that a following d, i, o, u, x, X, or n conversion specifier applies to an argument with type pointer to **size_t** or the corresponding signed integer type.

t Specifies that a following d, i, o, u, x, X, or n conversion specifier applies to an argument with type pointer to **ptrdiff_t** or the corresponding **unsigned** type.

L Specifies that a following a, A, e, E, f, F, g, or G conversion specifier applies to an argument with type pointer to **long double**.

If a length modifier appears with any conversion specifier other than as specified above, the behavior is undefined.

The following conversion specifier wide characters are valid:

d Matches an optionally signed decimal integer, whose format is the same as expected for the subject sequence of *wcstol()* with the value 10 for the *base* argument. In the absence of a size modifier, the application shall ensure that the corresponding argument is a pointer to **int**.

i Matches an optionally signed integer, whose format is the same as expected for the subject sequence of *wcstol()* with 0 for the *base* argument. In the absence of a size modifier, the application shall ensure that the corresponding argument is a pointer to **int**.

o Matches an optionally signed octal integer, whose format is the same as expected for the subject sequence of *wcstoul()* with the value 8 for the *base* argument. In the absence of a size modifier, the application shall ensure that the corresponding argument is a pointer to **unsigned**.

u Matches an optionally signed decimal integer, whose format is the same as expected for the subject sequence of *wcstoul()* with the value 10 for the *base* argument. In the absence of a size modifier, the application shall ensure that the corresponding argument is a pointer to **unsigned**.

33602	x	Matches an optionally signed hexadecimal integer, whose format is the same as expected for the subject sequence of <i>wcstoul()</i> with the value 16 for the <i>base</i> argument.
33603		
33604		In the absence of a size modifier, the application shall ensure that the corresponding argument is a pointer to unsigned .
33605		
33606	a, e, f, g	
33607		Matches an optionally signed floating-point number, infinity, or NaN whose format is the same as expected for the subject sequence of <i>wcstod()</i> . In the absence of a size modifier, the application shall ensure that the corresponding argument is a pointer to float .
33608		
33609		
33610		
33611		If the <i>fwprintf()</i> family of functions generates character string representations for infinity and NaN (a symbolic entity encoded in floating-point format) to support IEEE Std 754-1985, the <i>fwscanf()</i> family of functions shall recognize them as input.
33612		
33613		
33614	s	Matches a sequence of non-white-space wide characters. If no l (ell) qualifier is present, characters from the input field shall be converted as if by repeated calls to the <i>wcrtomb()</i> function, with the conversion state described by an mbstate_t object initialized to zero before the first wide character is converted. If the 'm' assignment-allocation character is not specified, the application shall ensure that the corresponding argument is a pointer to a character array large enough to accept the sequence and the terminating null character, which shall be added automatically. Otherwise, the application shall ensure that the corresponding argument is a pointer to a pointer to a wchar_t .
33615		
33616		
33617		
33618		
33619		
33620	CX	
33621		
33622		
33623		If the l (ell) qualifier is present and the 'm' assignment-allocation character is not specified, the application shall ensure that the corresponding argument is a pointer to an array of wchar_t large enough to accept the sequence and the terminating null wide character, which shall be added automatically. If the l (ell) qualifier is present and the 'm' assignment-allocation character is present, the application shall ensure that the corresponding argument is a pointer to a pointer to a wchar_t .
33624		
33625		
33626	CX	
33627		
33628		
33629	[	Matches a non-empty sequence of wide characters from a set of expected wide characters (the <i>scanset</i>). If no l (ell) qualifier is present, wide characters from the input field shall be converted as if by repeated calls to the <i>wcrtomb()</i> function, with the conversion state described by an mbstate_t object initialized to zero before the first wide character is converted. If the 'm' assignment-allocation character is not specified, the application shall ensure that the corresponding argument is a pointer to a character array large enough to accept the sequence and the terminating null character, which shall be added automatically. Otherwise, the application shall ensure that the corresponding argument is a pointer to a pointer to a wchar_t .
33630		
33631		
33632		
33633		
33634		
33635		
33636	CX	
33637		
33638		If an l (ell) qualifier is present and the 'm' assignment-allocation character is not specified, the application shall ensure that the corresponding argument is a pointer to an array of wchar_t large enough to accept the sequence and the terminating null wide character. If an l (ell) qualifier is present and the 'm' assignment-allocation character is specified, the application shall ensure that the corresponding argument is a pointer to a pointer to a wchar_t .
33639		
33640		
33641	CX	
33642		
33643		
33644		The conversion specification includes all subsequent wide characters in the <i>format</i> string up to and including the matching <right-square-bracket> ('] '). The wide characters between the square brackets (the <i>scanlist</i>) comprise the <i>scanset</i> , unless the wide character after the <left-square-bracket> is a <circumflex> (' ^ '), in which case the <i>scanset</i> contains all wide characters that do not appear in the <i>scanlist</i> between the <circumflex> and the <right-square-bracket>. If the conversion specification begins
33645		
33646		
33647		
33648		
33649		

33650			with "[]" or "[^]", the <right-square-bracket> is included in the scanlist and the
33651			next <right-square-bracket> is the matching <right-square-bracket> that ends the
33652			conversion specification; otherwise, the first <right-square-bracket> is the one that ends
33653			the conversion specification. If a '-' is in the scanlist and is not the first wide character,
33654			nor the second where the first wide character is a '^', nor the last wide character, the
33655			behavior is implementation-defined.
33656	c		Matches a sequence of wide characters of exactly the number specified by the field
33657			width (1 if no field width is present in the conversion specification).
33658			If no l (ell) length modifier is present, characters from the input field shall be converted
33659			as if by repeated calls to the <i>wcrtomb()</i> function, with the conversion state described by
33660			an mbstate_t object initialized to zero before the first wide character is converted. No
33661			null character is added. If the 'm' assignment-allocation character is not specified, the
33662			application shall ensure that the corresponding argument is a pointer to the initial
33663	CX		element of a character array large enough to accept the sequence. Otherwise, the
33664			application shall ensure that the corresponding argument is a pointer to a pointer to a
33665			char .
33666			No null wide character is added. If an l (ell) length modifier is present and the 'm'
33667			assignment-allocation character is not specified, the application shall ensure that the
33668			corresponding argument shall be a pointer to the initial element of an array of wchar_t
33669	CX		large enough to accept the sequence. If an l (ell) qualifier is present and the 'm'
33670			assignment-allocation character is specified, the application shall ensure that the
33671			corresponding argument is a pointer to a pointer to a wchar_t .
33672	p		Matches an implementation-defined set of sequences, which shall be the same as the set
33673			of sequences that is produced by the %p conversion specification of the corresponding
33674			<i>fwprintf()</i> functions. The application shall ensure that the corresponding argument is a
33675			pointer to a pointer to void . The interpretation of the input item is implementation-
33676			defined. If the input item is a value converted earlier during the same program
33677			execution, the pointer that results shall compare equal to that value; otherwise, the
33678			behavior of the %p conversion is undefined.
33679	n		No input is consumed. The application shall ensure that the corresponding argument is
33680			a pointer to the integer into which is to be written the number of wide characters read
33681			from the input so far by this call to the <i>fwscanf()</i> functions. Execution of a %n
33682			conversion specification shall not increment the assignment count returned at the
33683			completion of execution of the function. No argument shall be converted, but one shall
33684			be consumed. If the conversion specification includes an assignment-suppressing wide
33685			character or a field width, the behavior is undefined.
33686	XSI	C	Equivalent to lc .
33687	XSI	S	Equivalent to ls .
33688	%		Matches a single '%' wide character; no conversion or assignment shall occur. The
33689			complete conversion specification shall be %%.
33690			If a conversion specification is invalid, the behavior is undefined.
33691			The conversion specifiers A, E, F, G, and X are also valid and shall be equivalent to, respectively,
33692			a, e, f, g, and x.
33693			If end-of-file is encountered during input, conversion is terminated. If end-of-file occurs before
33694			any wide characters matching the current conversion specification (except for %n) have been
33695			read (other than leading white-space, where permitted), execution of the current conversion

33696 specification shall terminate with an input failure. Otherwise, unless execution of the current
 33697 conversion specification is terminated with a matching failure, execution of the following
 33698 conversion specification (if any) shall be terminated with an input failure.

33699 Reaching the end of the string in *swscanf()* shall be equivalent to encountering end-of-file for
 33700 *fwscanf()*.

33701 If conversion terminates on a conflicting input, the offending input shall be left unread in the
 33702 input. Any trailing white space (including <newline>) shall be left unread unless matched by a
 33703 conversion specification. The success of literal matches and suppressed assignments is only
 33704 directly determinable via the %n conversion specification.

33705 CX The *fwscanf()* and *wscanf()* functions may mark the last data access timestamp of the file
 33706 associated with *stream* for update. The last data access timestamp shall be marked for update by
 33707 the first successful execution of *fgetc()*, *fgetws()*, *fwscanf()*, *getc()*, *getwchar()*, *vfwscanf()*,
 33708 *vwscanf()*, or *wscanf()* using *stream* that returns data not supplied by a prior call to *ungetc()*.

33709 RETURN VALUE

33710 Upon successful completion, these functions shall return the number of successfully matched
 33711 and assigned input items; this number can be zero in the event of an early matching failure. If
 33712 the input ends before the first matching failure or conversion, EOF shall be returned. If any
 33713 CX error occurs, EOF shall be returned, and *errno* shall be set to indicate the error. If a read error
 33714 occurs, the error indicator for the stream shall be set.

33715 ERRORS

33716 For the conditions under which the *fwscanf()* functions shall fail and may fail, refer to *fgetc()*.

33717 In addition, the *fwscanf()* function shall fail if:

33718 CX [EILSEQ] Input byte sequence does not form a valid character.

33719 [ENOMEM] Insufficient storage space is available.

33720 In addition, the *fwscanf()* function may fail if:

33721 CX [EINVAL] There are insufficient arguments.

33722 EXAMPLES

33723 The call:

```
33724 int i, n; float x; char name[50];
33725 n = wscanf(L"%d%f%s", &i, &x, name);
```

33726 with the input line:

```
33727 25 54.32E-1 Hamster
```

33728 assigns to *n* the value 3, to *i* the value 25, to *x* the value 5.432, and *name* contains the string
 33729 "Hamster".

33730 The call:

```
33731 int i; float x; char name[50];
33732 (void) wscanf(L"%2d%f%*d %[0123456789]", &i, &x, name);
```

33733 with input:

```
33734 56789 0123 56a72
```

33735 assigns 56 to *i*, 789.0 to *x*, skips 0123, and places the string "56\0" in *name*. The next call to
 33736 *getchar()* shall return the character 'a'.

APPLICATION USAGE

In format strings containing the ' % ' form of conversion specifications, each argument in the argument list is used exactly once.

For functions that allocate memory as if by *malloc()*, the application should release such memory when it is no longer required by a call to *free()*. For *fwscanf()*, this is memory allocated via use of the ' m ' assignment-allocation character.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

Section 2.5 (on page 495), *getwc()*, *fwprintf()*, *setlocale()*, *wcstod()*, *wcstol()*, *wcstoul()*, *wcrtomb()*

XBD Chapter 7 (on page 135), *<stdio.h>*, *<wchar.h>*

CHANGE HISTORY

First released in Issue 5. Included for alignment with ISO/IEC 9899:1990/Amendment 1:1995 (E).

Issue 6

The normative text is updated to avoid use of the term “must” for application requirements.

The following changes are made for alignment with the ISO/IEC 9899:1999 standard:

- The prototypes for *fwscanf()* and *swscanf()* are updated.
- The DESCRIPTION is updated.
- The hh, ll, j, t, and z length modifiers are added.
- The a, A, and F conversion characters are added.

The DESCRIPTION is updated to use the terms “conversion specifier” and “conversion specification” consistently.

Issue 7

Austin Group Interpretation 1003.1-2001 #170 is applied.

SD5-XSH-ERN-132 is applied, adding the assignment-allocation character ' m '.

Functionality relating to the " % n \$ " form of conversion specification is moved from the XSI option to the Base.

Changes are made related to support for finegrained timestamps.

The APPLICATION USAGE section is updated to clarify that memory is allocated as if by *malloc()*.

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0229 [302] and XSH/TC1-2008/0230 + [14] are applied.

NAME

gai_strerror — address and name information error description

SYNOPSIS

```
#include <netdb.h>
```

```
const char *gai_strerror(int ecode);
```

DESCRIPTION

The *gai_strerror()* function shall return a text string describing an error value for the *getaddrinfo()* and *getnameinfo()* functions listed in the **<netdb.h>** header.

When the *ecode* argument is one of the following values listed in the **<netdb.h>** header:

[EAI_AGAIN]	[EAI_NONAME]
[EAI_BADFLAGS]	[EAI_OVERFLOW]
[EAI_FAIL]	[EAI_SERVICE]
[EAI_FAMILY]	[EAI_SOCKTYPE]
[EAI_MEMORY]	[EAI_SYSTEM]

the function return value shall point to a string describing the error. If the argument is not one of those values, the function shall return a pointer to a string whose contents indicate an unknown error.

RETURN VALUE

Upon successful completion, *gai_strerror()* shall return a pointer to an implementation-defined string.

ERRORS

No errors are defined.

EXAMPLES

None.

APPLICATION USAGE

None.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

freeaddrinfo()

XBD **<netdb.h>**

CHANGE HISTORY

First released in Issue 6. Derived from the XNS, Issue 5.2 specification.

The Open Group Base Resolution bwg2001-009 is applied, which changes the return type from **char *** to **const char ***. This is for coordination with the IPnG Working Group.

IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/22 is applied, adding the [EAI_OVERFLOW] error code.

33811 NAME

33812 getaddrinfo — get address information

33813 SYNOPSIS

```
33814 #include <sys/socket.h>
33815 #include <netdb.h>

33816 int getaddrinfo(const char *restrict nodename,
33817                const char *restrict servname,
33818                const struct addrinfo *restrict hints,
33819                struct addrinfo **restrict res);
```

33820 DESCRIPTION

33821 Refer to *freeaddrinfo()*.

33822 **NAME**

33823 getc — get a byte from a stream

33824 **SYNOPSIS**

33825 #include <stdio.h>

33826 int getc(FILE *stream);

33827 **DESCRIPTION**

33828 CX The functionality described on this reference page is aligned with the ISO C standard. Any
33829 conflict between the requirements described here and the ISO C standard is unintentional. This
33830 volume of POSIX.1-2008 defers to the ISO C standard.

33831 The *getc()* function shall be equivalent to *fgetc()*, except that if it is implemented as a macro it
33832 may evaluate *stream* more than once, so the argument should never be an expression with side-
33833 effects.

33834 **RETURN VALUE**33835 Refer to *fgetc()*.33836 **ERRORS**33837 Refer to *fgetc()*.33838 **EXAMPLES**

33839 None.

33840 **APPLICATION USAGE**

33841 If the integer value returned by *getc()* is stored into a variable of type **char** and then compared
33842 against the integer constant EOF, the comparison may never succeed, because sign-extension of
33843 a variable of type **char** on widening to integer is implementation-defined.

33844 Since it may be implemented as a macro, *getc()* may treat incorrectly a *stream* argument with
33845 side-effects. In particular, *getc(*f++)* does not necessarily work as expected. Therefore, use of this
33846 function should be preceded by "#undef getc" in such situations; *fgetc()* could also be used.

33847 **RATIONALE**

33848 None.

33849 **FUTURE DIRECTIONS**

33850 None.

33851 **SEE ALSO**33852 Section 2.5 (on page 495), *fgetc()*

33853 XBD <stdio.h>

33854 **CHANGE HISTORY**

33855 First released in Issue 1. Derived from Issue 1 of the SVID.

33856 **Issue 7**

33857 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0231 [14] is applied.

+

NAME

getc_unlocked, getchar_unlocked, putc_unlocked, putchar_unlocked — stdio with explicit client locking

SYNOPSIS

```
#include <stdio.h>

int getc_unlocked(FILE *stream);
int getchar_unlocked(void);
int putc_unlocked(int c, FILE *stream);
int putchar_unlocked(int c);
```

DESCRIPTION

Versions of the functions *getc()*, *getchar()*, *putc()*, and *putchar()* respectively named *getc_unlocked()*, *getchar_unlocked()*, *putc_unlocked()*, and *putchar_unlocked()* shall be provided which are functionally equivalent to the original versions, with the exception that they are not required to be implemented in a thread-safe manner. They may only safely be used within a scope protected by *flockfile()* (or *ftrylockfile()*) and *funlockfile()*. These functions may safely be used in a multi-threaded program if and only if they are called while the invoking thread owns the (FILE *) object, as is the case after a successful call to the *flockfile()* or *ftrylockfile()* functions.

If *getc_unlocked()* or *putc_unlocked()* are implemented as macros they may evaluate *stream* more than once, so the *stream* argument should never be an expression with side-effects.

RETURN VALUE

See *getc()*, *getchar()*, *putc()*, and *putchar()*.

ERRORS

See *getc()*, *getchar()*, *putc()*, and *putchar()*.

EXAMPLES

None.

APPLICATION USAGE

Since they may be implemented as macros, *getc_unlocked()* and *putc_unlocked()* may treat incorrectly a *stream* argument with side-effects. In particular, *getc_unlocked(*f++)* and *putc_unlocked(c,*f++)* do not necessarily work as expected. Therefore, use of these functions in such situations should be preceded by the following statement as appropriate:

```
#undef getc_unlocked
#undef putc_unlocked
```

RATIONALE

Some I/O functions are typically implemented as macros for performance reasons (for example, *putc()* and *getc()*). For safety, they need to be synchronized, but it is often too expensive to synchronize on every character. Nevertheless, it was felt that the safety concerns were more important; consequently, the *getc()*, *getchar()*, *putc()*, and *putchar()* functions are required to be thread-safe. However, unlocked versions are also provided with names that clearly indicate the unsafe nature of their operation but can be used to exploit their higher performance. These unlocked versions can be safely used only within explicitly locked program regions, using exported locking primitives. In particular, a sequence such as:

```
flockfile(fileptr);
putc_unlocked('1', fileptr);
putc_unlocked('\n', fileptr);
fprintf(fileptr, "Line 2\n");
funlockfile(fileptr);
```


is permissible, and results in the text sequence:

```
1
Line 2
```

being printed without being interspersed with output from other threads.

It would be wrong to have the standard names such as *getc()*, *putc()*, and so on, map to the “faster, but unsafe” rather than the “slower, but safe” versions. In either case, you would still want to inspect all uses of *getc()*, *putc()*, and so on, by hand when converting existing code. Choosing the safe bindings as the default, at least, results in correct code and maintains the “atomicity at the function” invariant. To do otherwise would introduce gratuitous synchronization errors into converted code. Other routines that modify the *stdio* (**FILE** *) structures or buffers are also safely synchronized.

Note that there is no need for functions of the form *getc_locked()*, *putc_locked()*, and so on, since this is the functionality of *getc()*, *putc()*, *et al.* It would be inappropriate to use a feature test macro to switch a macro definition of *getc()* between *getc_locked()* and *getc_unlocked()*, since the ISO C standard requires an actual function to exist, a function whose behavior could not be changed by the feature test macro. Also, providing both the *xxx_locked()* and *xxx_unlocked()* forms leads to the confusion of whether the suffix describes the behavior of the function or the circumstances under which it should be used.

Three additional routines, *flockfile()*, *ftrylockfile()*, and *funlockfile()* (which may be macros), are provided to allow the user to delineate a sequence of I/O statements that are executed synchronously.

The *ungetc()* function is infrequently called relative to the other functions/macros so no unlocked variation is needed.

FUTURE DIRECTIONS

None.

SEE ALSO

[Section 2.5](#) (on page 495), *flockfile()*, *getc()*, *getchar()*, *putc()*, *putchar()*

XBD [<stdio.h>](#)

CHANGE HISTORY

First released in Issue 5. Included for alignment with the POSIX Threads Extension.

Issue 6

These functions are marked as part of the Thread-Safe Functions option.

The Open Group Corrigendum U030/2 is applied, adding APPLICATION USAGE describing how applications should be written to avoid the case when the functions are implemented as macros.

Issue 7

The *getc_unlocked()*, *getchar_unlocked()*, *putc_unlocked()*, and *putchar_unlocked()* functions are moved from the Thread-Safe Functions option to the Base.

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0232 [395], XSH/TC1-2008/0233 [395], + XSH/TC1-2008/0234 [395], and XSH/TC1-2008/0235 [14] are applied.

33944 **NAME**33945 getchar — get a byte from a *stdin* stream33946 **SYNOPSIS**

33947 #include <stdio.h>

33948 int getchar(void);

33949 **DESCRIPTION**

33950 **CX** The functionality described on this reference page is aligned with the ISO C standard. Any
33951 conflict between the requirements described here and the ISO C standard is unintentional. This
33952 volume of POSIX.1-2008 defers to the ISO C standard.

33953 The *getchar()* function shall be equivalent to *getc(stdin)*.33954 **RETURN VALUE**33955 Refer to *fgetc()*.33956 **ERRORS**33957 Refer to *fgetc()*.33958 **EXAMPLES**

33959 None.

33960 **APPLICATION USAGE**

33961 If the integer value returned by *getchar()* is stored into a variable of type **char** and then
33962 compared against the integer constant EOF, the comparison may never succeed, because sign-
33963 extension of a variable of type **char** on widening to integer is implementation-defined.

33964 **RATIONALE**

33965 None.

33966 **FUTURE DIRECTIONS**

33967 None.

33968 **SEE ALSO**33969 Section 2.5 (on page 495), *getc()*

33970 XBD <stdio.h>

33971 **CHANGE HISTORY**

33972 First released in Issue 1. Derived from Issue 1 of the SVID.

33973 **Issue 7**

33974 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0236 [14] is applied.

+

33975 **NAME**

33976 getchar_unlocked — stdio with explicit client locking

33977 **SYNOPSIS**

33978 CX #include <stdio.h>

33979 int getchar_unlocked(void);

33980 **DESCRIPTION**33981 Refer to *getc_unlocked()*.

NAME

getcwd — get the pathname of the current working directory

SYNOPSIS

```
#include <unistd.h>

char *getcwd(char *buf, size_t size);
```

DESCRIPTION

The *getcwd()* function shall place an absolute pathname of the current working directory in the array pointed to by *buf*, and return *buf*. The pathname shall contain no components that are dot or dot-dot, or are symbolic links.

If there are multiple pathnames that *getcwd()* could place in the array pointed to by *buf*, one beginning with a single <slash> character and one or more beginning with two <slash> characters, then *getcwd()* shall place the pathname beginning with a single <slash> character in the array. The pathname shall not contain any unnecessary <slash> characters after the leading one or two <slash> characters.

The *size* argument is the size in bytes of the character array pointed to by the *buf* argument. If *buf* is a null pointer, the behavior of *getcwd()* is unspecified.

RETURN VALUE

Upon successful completion, *getcwd()* shall return the *buf* argument. Otherwise, *getcwd()* shall return a null pointer and set *errno* to indicate the error. The contents of the array pointed to by *buf* are then undefined.

ERRORS

The *getcwd()* function shall fail if:

[EINVAL]	The <i>size</i> argument is 0.
[ERANGE]	The <i>size</i> argument is greater than 0, but is smaller than the length of the string +1.

The *getcwd()* function may fail if:

[EACCES]	Search permission was denied for the current directory, or read or search permission was denied for a directory above the current directory in the file hierarchy.
[ENOMEM]	Insufficient storage space is available.

EXAMPLES

The following example uses {PATH_MAX} as the initial buffer size (unless it is indeterminate or very large), and calls *getcwd()* with progressively larger buffers until it does not give an [ERANGE] error.

```
#include <stdlib.h>
#include <errno.h>
#include <unistd.h>

...

long path_max;
size_t size;
char *buf;
char *ptr;

path_max = pathconf(".", _PC_PATH_MAX);
if (path_max == -1)
```

```

34026         size = 1024;
34027     else if (path_max > 10240)
34028         size = 10240;
34029     else
34030         size = path_max;
34031     for (buf = ptr = NULL; ptr == NULL; size *= 2)
34032     {
34033         if ((buf = realloc(buf, size)) == NULL)
34034         {
34035             ... handle error ...
34036         }
34037         ptr = getcwd(buf, size);
34038         if (ptr == NULL && errno != ERANGE)
34039         {
34040             ... handle error ...
34041         }
34042     }
34043     ...
34044     free (buf);

```

APPLICATION USAGE

If the pathname obtained from *getcwd()* is longer than {PATH_MAX} bytes, it could produce an [ENAMETOOLONG] error if passed to *chdir()*. Therefore, in order to return to that directory it may be necessary to break the pathname into sections shorter than {PATH_MAX} bytes and call *chdir()* on each section in turn (the first section being an absolute pathname and subsequent sections being relative pathnames). A simpler way to handle saving and restoring the working directory when it may be deeper than {PATH_MAX} bytes in the file hierarchy is to use a file descriptor and *fchdir()*, rather than *getcwd()* and *chdir()*. However, the two methods do have some differences. The *fchdir()* approach causes the program to restore a working directory even if it has been renamed in the meantime, whereas the *chdir()* approach restores to a directory with the same name as the original, even if the directories were renamed in the meantime. Since the *fchdir()* approach does not access parent directories, it can succeed when *getcwd()* would fail due to permissions problems. In applications conforming to earlier versions of this standard, it was not possible to use the *fchdir()* approach when the working directory is searchable but not readable, as the only way to open a directory was with O_RDONLY, whereas the *getcwd()* approach can succeed in this case.

RATIONALE

Having *getcwd()* take no arguments and instead use the *malloc()* function to produce space for the returned argument was considered. The advantage is that *getcwd()* knows how big the working directory pathname is and can allocate an appropriate amount of space. But the programmer would have to use the *free()* function to free the resulting object, or each use of *getcwd()* would further reduce the available memory. Finally, *getcwd()* is taken from the SVID where it has the two arguments used in this volume of POSIX.1-2008.

The older function *getwd()* was rejected for use in this context because it had only a buffer argument and no *size* argument, and thus had no way to prevent overwriting the buffer, except to depend on the programmer to provide a large enough buffer.

On some implementations, if *buf* is a null pointer, *getcwd()* may obtain *size* bytes of memory using *malloc()*. In this case, the pointer returned by *getcwd()* may be used as the argument in a subsequent call to *free()*. Invoking *getcwd()* with *buf* as a null pointer is not recommended in conforming applications.

Earlier implementations of *getcwd()* sometimes generated pathnames like ".../.../subdirname" internally, using them to explore the path of ancestor directories back to the root. If one of these internal pathnames exceeded {PATH_MAX} in length, the implementation could fail with *errno* set to [ENAMETOOLONG]. This is no longer allowed.

If a program is operating in a directory where some (grand)parent directory does not permit reading, *getcwd()* may fail, as in most implementations it must read the directory to determine the name of the file. This can occur if search, but not read, permission is granted in an intermediate directory, or if the program is placed in that directory by some more privileged process (for example, login). Including the [EACCES] error condition makes the reporting of the error consistent and warns the application developer that *getcwd()* can fail for reasons beyond the control of the application developer or user. Some implementations can avoid this occurrence (for example, by implementing *getcwd()* using *pwd*, where *pwd* is a set-user-root process), thus the error was made optional. Since this volume of POSIX.1-2008 permits the addition of other errors, this would be a common addition and yet one that applications could not be expected to deal with without this addition.

FUTURE DIRECTIONS

None.

SEE ALSO

malloc()

XBD <unistd.h>

CHANGE HISTORY

First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The [ENOMEM] optional error condition is added.

Issue 7

Austin Group Interpretation 1003.1-2001 #140 is applied, changing the text for consistency with the *pwd* utility, adding text to address the case where the current directory is deeper in the file hierarchy than {PATH_MAX} bytes, and adding the requirements relating to pathnames beginning with two <slash> characters.

34106 **NAME**34107 `getdate` — convert user format date and time34108 **SYNOPSIS**

```
34109 XSI      #include <time.h>
34110      struct tm *getdate(const char *string);
```

34111 **DESCRIPTION**

34112 The `getdate()` function shall convert a string representation of a date or time into a broken-down
 34113 time.

34114 The external variable or macro `getdate_err`, which has type **int**, is used by `getdate()` to return error
 34115 values. It is unspecified whether `getdate_err` is a macro or an identifier declared with external
 34116 linkage, and whether or not it is a modifiable lvalue. If a macro definition is suppressed in order
 34117 to access an actual object, or a program defines an identifier with the name `getdate_err`, the
 34118 behavior is undefined.

34119 Templates are used to parse and interpret the input string. The templates are contained in a text
 34120 file identified by the environment variable `DATMSK`. The `DATMSK` variable should be set to
 34121 indicate the full pathname of the file that contains the templates. The first line in the template
 34122 that matches the input specification is used for interpretation and conversion into the internal
 34123 time format.

34124 The following conversion specifications shall be supported:

34125	<code>%%</code>	Equivalent to <code>%</code> .
34126	<code>%a</code>	Abbreviated weekday name.
34127	<code>%A</code>	Full weekday name.
34128	<code>%b</code>	Abbreviated month name.
34129	<code>%B</code>	Full month name.
34130	<code>%c</code>	Locale's appropriate date and time representation.
34131	<code>%C</code>	Century number [00,99]; leading zeros are permitted but not required.
34132	<code>%d</code>	Day of month [01,31]; the leading 0 is optional.
34133	<code>%D</code>	Date as <code>%m/%d/%y</code> .
34134	<code>%e</code>	Equivalent to <code>%d</code> .
34135	<code>%h</code>	Abbreviated month name.
34136	<code>%H</code>	Hour [00,23].
34137	<code>%I</code>	Hour [01,12].
34138	<code>%m</code>	Month number [01,12].
34139	<code>%M</code>	Minute [00,59].
34140	<code>%n</code>	Equivalent to <code><newline></code> .
34141	<code>%p</code>	Locale's equivalent of either AM or PM.
34142	<code>%r</code>	The locale's appropriate representation of time in AM and PM notation. In the POSIX 34143 locale, this shall be equivalent to <code>%I:%M:%S %p</code> .

34144	%R	Time as %H:%M.
34145	%S	Seconds [00,60]. The range goes to 60 (rather than stopping at 59) to allow positive leap seconds to be expressed. Since leap seconds cannot be predicted by any algorithm, leap second data must come from some external source.
34146		
34147		
34148	%t	Equivalent to <tab>.
34149	%T	Time as %H:%M:%S.
34150	%w	Weekday number (Sunday = [0,6]).
34151	%x	Locale's appropriate date representation.
34152	%X	Locale's appropriate time representation.
34153	%y	Year within century. When a century is not otherwise specified, values in the range [69,99] shall refer to years 1969 to 1999 inclusive, and values in the range [00,68] shall refer to years 2000 to 2068 inclusive.
34154		
34155		
34156		Note: It is expected that in a future version of this standard the default century inferred from a 2-digit year will change. (This would apply to all commands accepting a 2-digit year as input.)
34157		
34158		
34159	%Y	Year as "ccyy" (for example, 2001).
34160	%Z	Timezone name or no characters if no timezone exists. If the timezone supplied by %Z is not the timezone that <i>getdate()</i> expects, an invalid input specification error shall result. The <i>getdate()</i> function calculates an expected timezone based on information supplied to the function (such as the hour, day, and month).
34161		
34162		
34163		
34164		The match between the template and input specification performed by <i>getdate()</i> shall be case-insensitive.
34165		
34166		The month and weekday names can consist of any combination of upper and lowercase letters. The process can request that the input date or time specification be in a specific language by setting the <i>LC_TIME</i> category (see <i>setlocale()</i>).
34167		
34168		
34169		Leading zeros are not necessary for the descriptors that allow leading zeros. However, at most two digits are allowed for those descriptors, including leading zeros. Extra white space in either the template file or in <i>string</i> shall be ignored.
34170		
34171		
34172		The results are undefined if the conversion specifications %c, %x, and %X include unsupported conversion specifications.
34173		
34174		The following rules apply for converting the input specification into the internal format:
34175		• If %Z is being scanned, then <i>getdate()</i> shall initialize the broken-down time to be the current time in the scanned timezone. Otherwise, it shall initialize the broken-down time based on the current local time as if <i>localtime()</i> had been called.
34176		
34177		
34178		• If only the weekday is given, the day chosen shall be the day, starting with today and moving into the future, which first matches the named day.
34179		
34180		• If only the month (and no year) is given, the month chosen shall be the month, starting with the current month and moving into the future, which first matches the named month. The first day of the month shall be assumed if no day is given.
34181		
34182		
34183		• If no hour, minute, and second are given, the current hour, minute, and second shall be assumed.
34184		

- If no date is given, the hour chosen shall be the hour, starting with the current hour and moving into the future, which first matches the named hour.

If a conversion specification in the DATEMSK file does not correspond to one of the conversion specifications above, the behavior is unspecified.

The `getdate()` function need not be thread-safe.

RETURN VALUE

Upon successful completion, `getdate()` shall return a pointer to a **struct tm**. Otherwise, it shall return a null pointer and set `getdate_err` to indicate the error.

ERRORS

The `getdate()` function shall fail in the following cases, setting `getdate_err` to the value shown in the list below. Any changes to `errno` are unspecified.

1. The `DATEMSK` environment variable is null or undefined.
2. The template file cannot be opened for reading.
3. Failed to get file status information.
4. The template file is not a regular file.
5. An I/O error is encountered while reading the template file.
6. Memory allocation failed (not enough memory available).
7. There is no line in the template that matches the input.
8. Invalid input specification. For example, February 31; or a time is specified that cannot be represented in a **time_t** (representing the time in seconds since the Epoch).

EXAMPLES

1. The following example shows the possible contents of a template:

```
%m
%A %B %d, %Y, %H:%M:%S
%A
%B
%m/%d/%y %I %p
%d, %m, %Y %H:%M
at %A the %dst of %B in %Y
run job at %I %p, %B %dnd
%A den %d. %B %Y %H.%M Uhr
```

2. The following are examples of valid input specifications for the template in Example 1:

```
getdate("10/1/87 4 PM");
getdate("Friday");
getdate("Friday September 18, 1987, 10:30:30");
getdate("24,9,1986 10:30");
getdate("at monday the 1st of december in 1986");
getdate("run job at 3 PM, december 2nd");
```

If the `LC_TIME` category is set to a German locale that includes *freitag* as a weekday name and *oktober* as a month name, the following would be valid:

```
getdate("freitag den 10. oktober 1986 10.30 Uhr");
```

3. The following example shows how local date and time specification can be defined in the template:

Invocation	Line in Template
getdate("11/27/86")	%m/%d/%y
getdate("27.11.86")	%d.%m.%y
getdate("86-11-27")	%y-%m-%d
getdate("Friday 12:00:00")	%A %H:%M:%S

4. The following examples help to illustrate the above rules assuming that the current date is Mon Sep 22 12:19:47 EDT 1986 and the *LC_TIME* category is set to the default C locale:

Input	Line in Template	Date
Mon	%a	Mon Sep 22 12:19:47 EDT 1986
Sun	%a	Sun Sep 28 12:19:47 EDT 1986
Fri	%a	Fri Sep 26 12:19:47 EDT 1986
September	%B	Mon Sep 1 12:19:47 EDT 1986
January	%B	Thu Jan 1 12:19:47 EST 1987
December	%B	Mon Dec 1 12:19:47 EST 1986
Sep Mon	%b %a	Mon Sep 1 12:19:47 EDT 1986
Jan Fri	%b %a	Fri Jan 2 12:19:47 EST 1987
Dec Mon	%b %a	Mon Dec 1 12:19:47 EST 1986
Jan Wed 1989	%b %a %Y	Wed Jan 4 12:19:47 EST 1989
Fri 9	%a %H	Fri Sep 26 09:00:00 EDT 1986
Feb 10:30	%b %H:%S	Sun Feb 1 10:00:30 EST 1987
10:30	%H:%M	Tue Sep 23 10:30:00 EDT 1986
13:30	%H:%M	Mon Sep 22 13:30:00 EDT 1986

APPLICATION USAGE

Although historical versions of *getdate()* did not require that **<time.h>** declare the external variable *getdate_err*, this volume of POSIX.1-2008 does require it. The standard developers encourage applications to remove declarations of *getdate_err* and instead incorporate the declaration by including **<time.h>**.

Applications should use %Y (4-digit years) in preference to %y (2-digit years).

RATIONALE

In standard locales, the conversion specifications %c, %x, and %X do not include unsupported conversion specifiers and so the text regarding results being undefined is not a problem in that case.

FUTURE DIRECTIONS

None.

SEE ALSO

ctime(), *localtime()*, *setlocale()*, *strftime()*, *times()*

XBD **<time.h>**

CHANGE HISTORY

First released in Issue 4, Version 2.

Issue 5

- 34267 Moved from X/OPEN UNIX extension to BASE.
- 34268
- 34269 The last paragraph of the DESCRIPTION is added.
- 34270 The %C conversion specification is added, and the exact meaning of the %y conversion
- 34271 specification is clarified in the DESCRIPTION.
- 34272 A note indicating that this function need not be reentrant is added to the DESCRIPTION.
- 34273 The %R conversion specification is changed to follow historical practice.

Issue 6

- 34274 The DESCRIPTION is updated to refer to “seconds since the Epoch” rather than “seconds since
- 34275 00:00:00 UTC (Coordinated Universal Time), January 1 1970” for consistency with other *time*
- 34276 functions.
- 34277
- 34278 The description of %S is updated so that the valid range is [00,60] rather than [00,61].
- 34279 The DESCRIPTION is updated to refer to conversion specifications instead of field descriptors
- 34280 for consistency with other functions.

Issue 7

- 34281 Austin Group Interpretation 1003.1-2001 #156 is applied.
- 34282
- 34283 The description of the *getdate_err* value is expanded.

NAME

getdelim, getline — read a delimited record from *stream*

SYNOPSIS

```
CX      #include <stdio.h>

ssize_t getdelim(char **restrict lineptr, size_t *restrict n,
                 int delimiter, FILE *restrict stream);
ssize_t getline(char **restrict lineptr, size_t *restrict n,
                FILE *restrict stream);
```

DESCRIPTION

The *getdelim()* function shall read from *stream* until it encounters a character matching the *delimiter* character. The *delimiter* argument is an **int**, the value of which the application shall ensure is a character representable as an **unsigned char** of equal value that terminates the read process. If the *delimiter* argument has any other value, the behavior is undefined.

The application shall ensure that **lineptr* is a valid argument that could be passed to the *free()* function. If **n* is non-zero, the application shall ensure that **lineptr* either points to an object of size at least **n* bytes, or is a null pointer.

The size of the object pointed to by **lineptr* shall be increased to fit the incoming line, if it isn't already large enough, including room for the delimiter and a terminating NUL. The characters read, including any delimiter, shall be stored in the string pointed to by the *lineptr* argument, and a terminating NUL added when the delimiter or end of file is encountered.

The *getline()* function shall be equivalent to the *getdelim()* function with the *delimiter* character equal to the <newline> character.

The *getdelim()* and *getline()* functions may mark the last data access timestamp of the file associated with *stream* for update. The last data access timestamp shall be marked for update by the first successful execution of *fgetc()*, *fgets()*, *fread()*, *fscanf()*, *getc()*, *getchar()*, *getdelim()*, *getline()*, *gets()*, or *scanf()* using *stream* that returns data not supplied by a prior call to *ungetc()*.

RETURN VALUE

Upon successful completion, the *getline()* and *getdelim()* functions shall return the number of characters written into the buffer, including the delimiter character if one was encountered before EOF, but excluding the terminating NUL character. If no characters were read, and the end-of-file indicator for the stream is set, or if the stream is at end-of-file, the end-of-file indicator for the stream shall be set and the function shall return -1. If an error occurs, the error indicator for the stream shall be set, and the function shall return -1 and set *errno* to indicate the error.

ERRORS

For the conditions under which the *getdelim()* and *getline()* functions shall fail and may fail, refer to *fgetc()*.

In addition, these functions shall fail if:

[EINVAL] *lineptr* or *n* is a null pointer.

[ENOMEM] Insufficient memory is available.

These functions may fail if:

[EOVERFLOW] More than {SSIZE_MAX} characters were read without encountering the *delimiter* character.

EXAMPLES

```

#include <stdio.h>
#include <stdlib.h>

int main(void)
{
    FILE *fp;
    char *line = NULL;
    size_t len = 0;
    ssize_t read;
    fp = fopen("/etc/motd", "r");
    if (fp == NULL)
        exit(1);
    while ((read = getline(&line, &len, fp)) != -1) {
        printf("Retrieved line of length %zu :\\n", read);
        printf("%s", line);
    }
    if (ferror(fp)) {
        /* handle error */
    }
    free(line);
    fclose(fp);
    return 0;
}

```

APPLICATION USAGE

Setting **lineptr* to a null pointer and **n* to zero are allowed and a recommended way to start parsing a file.

The *ferror()* or *feof()* functions should be used to distinguish between an error condition and an end-of-file condition.

Although a NUL terminator is always supplied after the line, note that *strlen(*lineptr)* will be smaller than the return value if the line contains embedded NUL characters.

RATIONALE

These functions are widely used to solve the problem that the *fgets()* function has with long lines. The functions automatically enlarge the target buffers if needed. These are especially useful since they reduce code needed for applications.

FUTURE DIRECTIONS

None.

SEE ALSO

[Section 2.5](#) (on page 495), *fgetc()*, *fgets()*, *free()*

XBD [<stdio.h>](#)

CHANGE HISTORY

First released in Issue 7.

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0237 [14] is applied.

+

34368 **NAME**

34369 getegid — get the effective group ID

34370 **SYNOPSIS**

34371 #include <unistd.h>

34372 gid_t getegid(void);

34373 **DESCRIPTION**34374 The *getegid()* function shall return the effective group ID of the calling process.34375 **RETURN VALUE**34376 The *getegid()* function shall always be successful and no return value is reserved to indicate an
34377 error.34378 **ERRORS**

34379 No errors are defined.

34380 **EXAMPLES**

34381 None.

34382 **APPLICATION USAGE**

34383 None.

34384 **RATIONALE**

34385 None.

34386 **FUTURE DIRECTIONS**

34387 None.

34388 **SEE ALSO**34389 *geteuid()*, *getgid()*, *getuid()*, *setegid()*, *seteuid()*, *setgid()*, *setregid()*, *setreuid()*, *setuid()*

34390 XBD <sys/types.h>, <unistd.h>

34391 **CHANGE HISTORY**

34392 First released in Issue 1. Derived from Issue 1 of the SVID.

34393 **Issue 6**

34394 In the SYNOPSIS, the optional include of the <sys/types.h> header is removed.

34395 The following new requirements on POSIX implementations derive from alignment with the
34396 Single UNIX Specification:

- 34397
- The requirement to include <sys/types.h> has been removed. Although <sys/types.h> was
34398 required for conforming implementations of previous POSIX specifications, it was not
34399 required for UNIX applications.

34400 **NAME**

34401 getenv — get value of an environment variable

34402 **SYNOPSIS**

34403 #include <stdlib.h>

34404 char *getenv(const char *name);

34405 **DESCRIPTION**

34406 CX The functionality described on this reference page is aligned with the ISO C standard. Any
 34407 conflict between the requirements described here and the ISO C standard is unintentional. This
 34408 volume of POSIX.1-2008 defers to the ISO C standard.

34409 The *getenv()* function shall search the environment of the calling process (see XBD [Chapter 8](#), on
 34410 page 173) for the environment variable *name* if it exists and return a pointer to the value of the
 34411 environment variable. If the specified environment variable cannot be found, a null pointer shall
 34412 be returned. The application shall ensure that it does not modify the string pointed to by the
 34413 *getenv()* function.

34414 CX The returned string pointer might be invalidated or the string content might be overwritten by a
 34415 CX subsequent call to *getenv()*, *setenv()*, *unsetenv()*, or (if supported) *putenv()* but they shall not be
 34416 affected by a call to any other function in this volume of POSIX.1-2008.

34417 CX The *getenv()* function need not be thread-safe. -

34418 **RETURN VALUE**

34419 Upon successful completion, *getenv()* shall return a pointer to a string containing the *value* for
 34420 the specified *name*. If the specified *name* cannot be found in the environment of the calling
 34421 process, a null pointer shall be returned.

34422 **ERRORS**

34423 No errors are defined.

34424 **EXAMPLES**34425 **Getting the Value of an Environment Variable**34426 The following example gets the value of the *HOME* environment variable.

34427 #include <stdlib.h>

34428 ...

34429 const char *name = "HOME";

34430 char *value;

34431 value = getenv(name);

34432 **APPLICATION USAGE**

34433 None.

34434 **RATIONALE**

34435 The *clearenv()* function was considered but rejected. The *putenv()* function has now been
 34436 included for alignment with the Single UNIX Specification.

34437 The *getenv()* function is inherently not thread-safe because it returns a value pointing to static
 34438 data.

34439 Conforming applications are required not to directly modify the pointers to which *environ*
 34440 points, but to use only the *setenv()*, *unsetenv()*, and *putenv()* functions, or assignment to *environ*
 34441 itself, to manipulate the process environment. This constraint allows the implementation to
 34442 properly manage the memory it allocates. This enables the implementation to free any space it

has allocated to strings (and perhaps the pointers to them) stored in *environ* when *unsetenv()* is called. A C runtime start-up procedure (that which invokes *main()* and perhaps initializes *environ*) can also initialize a flag indicating that none of the environment has yet been copied to allocated storage, or that the separate table has not yet been initialized. If the application switches to a complete new environment by assigning a new value to *environ*, this can be detected by *getenv()*, *setenv()*, *unsetenv()*, or *putenv()* and the implementation can at that point reinitialize based on the new environment. (This may include copying the environment strings into a new array and assigning *environ* to point to it.)

In fact, for higher performance of *getenv()*, implementations that do not provide *putenv()* could also maintain a separate copy of the environment in a data structure that could be searched much more quickly (such as an indexed hash table, or a binary tree), and update both it and the linear list at *environ* when *setenv()* or *unsetenv()* is invoked. On implementations that do provide *putenv()*, such a copy might still be worthwhile but would need to allow for the fact that applications can directly modify the content of environment strings added with *putenv()*. For example, if an environment string found by searching the copy is one that was added using *putenv()*, the implementation would need to check that the string in *environ* still has the same name (and value, if the copy includes values), and whenever searching the copy produces no match the implementation would then need to search each environment string in *environ* that was added using *putenv()* in case any of them have changed their names and now match. Thus, each use of *putenv()* to add to the environment would reduce the speed advantage of having the copy.

Performance of *getenv()* can be important for applications which have large numbers of environment variables. Typically, applications like this use the environment as a resource database of user-configurable parameters. The fact that these variables are in the user's shell environment usually means that any other program that uses environment variables (such as *ls*, which attempts to use *COLUMNS*), or really almost any utility (*LANG*, *LC_ALL*, and so on) is similarly slowed down by the linear search through the variables.

An implementation that maintains separate data structures, or even one that manages the memory it consumes, is not currently required as it was thought it would reduce consensus among implementors who do not want to change their historical implementations.

FUTURE DIRECTIONS

A future version may add one or more functions to access and modify the environment in a thread-safe manner.

SEE ALSO

exec, *putenv()*, *setenv()*, *unsetenv()*

XBD Chapter 8 (on page 173), `<stdlib.h>`

CHANGE HISTORY

First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 5

Normative text previously in the APPLICATION USAGE section is moved to the RETURN VALUE section.

A note indicating that this function need not be reentrant is added to the DESCRIPTION.

Issue 6

The following changes were made to align with the IEEE P1003.1a draft standard:

- 34487 • References added to the new *setenv()* and *unsetenv()* functions.
- 34488 The normative text is updated to avoid use of the term “must” for application requirements.
- 34489 **Issue 7**
- 34490 Austin Group Interpretation 1003.1-2001 #062 is applied, clarifying that a call to *putenv()* may
- 34491 also cause the string to be overwritten.
- 34492 Austin Group Interpretation 1003.1-2001 #148 is applied, adding the FUTURE DIRECTIONS.
- 34493 Austin Group Interpretation 1003.1-2001 #156 is applied.
- 34494 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0238 [75,428], XSH/TC1-2008/0239 +
- 34495 [167], and XSH/TC1-2008/0240 [167] are applied.

34496 **NAME**

34497 geteuid — get the effective user ID

34498 **SYNOPSIS**

34499 #include <unistd.h>

34500 uid_t geteuid(void);

34501 **DESCRIPTION**34502 The *geteuid()* function shall return the effective user ID of the calling process.34503 **RETURN VALUE**34504 The *geteuid()* function shall always be successful and no return value is reserved to indicate an
34505 error.34506 **ERRORS**

34507 No errors are defined.

34508 **EXAMPLES**

34509 None.

34510 **APPLICATION USAGE**

34511 None.

34512 **RATIONALE**

34513 None.

34514 **FUTURE DIRECTIONS**

34515 None.

34516 **SEE ALSO**34517 *getegid()*, *getgid()*, *getuid()*, *setegid()*, *seteuid()*, *setgid()*, *setregid()*, *setreuid()*, *setuid()*

34518 XBD <sys/types.h>, <unistd.h>

34519 **CHANGE HISTORY**

34520 First released in Issue 1. Derived from Issue 1 of the SVID.

34521 **Issue 6**

34522 In the SYNOPSIS, the optional include of the <sys/types.h> header is removed.

34523 The following new requirements on POSIX implementations derive from alignment with the
34524 Single UNIX Specification:

- 34525
- The requirement to include <sys/types.h> has been removed. Although <sys/types.h> was
34526 required for conforming implementations of previous POSIX specifications, it was not
34527 required for UNIX applications.

34528 **NAME**

34529 getgid — get the real group ID

34530 **SYNOPSIS**

34531 #include <unistd.h>

34532 gid_t getgid(void);

34533 **DESCRIPTION**34534 The *getgid()* function shall return the real group ID of the calling process.34535 **RETURN VALUE**34536 The *getgid()* function shall always be successful and no return value is reserved to indicate an
34537 error.34538 **ERRORS**

34539 No errors are defined.

34540 **EXAMPLES**

34541 None.

34542 **APPLICATION USAGE**

34543 None.

34544 **RATIONALE**

34545 None.

34546 **FUTURE DIRECTIONS**

34547 None.

34548 **SEE ALSO**34549 *getegid(), geteuid(), getuid(), setegid(), seteuid(), setgid(), setregid(), setreuid(), setuid()*34550 XBD [**<sys/types.h>**](#), [**<unistd.h>**](#)34551 **CHANGE HISTORY**

34552 First released in Issue 1. Derived from Issue 1 of the SVID.

34553 **Issue 6**34554 In the SYNOPSIS, the optional include of the **<sys/types.h>** header is removed.34555 The following new requirements on POSIX implementations derive from alignment with the
34556 Single UNIX Specification:

- 34557
- The requirement to include **<sys/types.h>** has been removed. Although **<sys/types.h>** was
34558 required for conforming implementations of previous POSIX specifications, it was not
34559 required for UNIX applications.

34560 **NAME**

34561 getgrent — get the group database entry

34562 **SYNOPSIS**

34563 XSI #include <grp.h>

34564 struct group *getgrent(void);

34565 **DESCRIPTION**

34566 Refer to *endgrent()*.

NAME

getgrgid, getgrgid_r — get group database entry for a group ID

SYNOPSIS

```
#include <grp.h>

struct group *getgrgid(gid_t gid);
int getgrgid_r(gid_t gid, struct group *grp, char *buffer,
               size_t bufsize, struct group **result);
```

DESCRIPTION

The *getgrgid()* function shall search the group database for an entry with a matching *gid*.

The *getgrgid()* function need not be thread-safe.

The *getgrgid_r()* function shall update the **group** structure pointed to by *grp* and store a pointer to that structure at the location pointed to by *result*. The structure shall contain an entry from the group database with a matching *gid*. Storage referenced by the group structure is allocated from the memory provided with the *buffer* parameter, which is *bufsize* bytes in size. A call to *sysconf(_SC_GETGR_R_SIZE_MAX)* returns either *-1* without changing *errno* or an initial value suggested for the size of this buffer. A null pointer shall be returned at the location pointed to by *result* on error or if the requested entry is not found.

RETURN VALUE

Upon successful completion, *getgrgid()* shall return a pointer to a **struct group** with the structure defined in **<grp.h>** with a matching entry if one is found. The *getgrgid()* function shall return a null pointer if either the requested entry was not found, or an error occurred. On error, *errno* shall be set to indicate the error.

The application shall not modify the structure to which the return value points, nor any storage areas pointed to by pointers within the structure. The returned pointer, and pointers within the structure, might be invalidated or the structure or the storage areas might be overwritten by a subsequent call to *getgrent()*, *getgrgid()*, or *getgrnam()*.

If successful, the *getgrgid_r()* function shall return zero; otherwise, an error number shall be returned to indicate the error.

ERRORS

The *getgrgid()* and *getgrgid_r()* functions may fail if:

[EIO]	An I/O error has occurred.
[EINTR]	A signal was caught during <i>getgrgid()</i> .
[EMFILE]	All file descriptors available to the process are currently open.
[ENFILE]	The maximum allowable number of files is currently open in the system.
- The *getgrgid_r()* function may fail if:
- | | |
|----------|--|
| [ERANGE] | Insufficient storage was supplied via <i>buffer</i> and <i>bufsize</i> to contain the data to be referenced by the resulting group structure. |
|----------|--|

EXAMPLES

Note that *sysconf*(_SC_GETGR_R_SIZE_MAX) may return -1 if there is no hard limit on the size of the buffer needed to store all the groups returned. This example shows how an application can allocate a buffer of sufficient size to work with *getgrgid_r*().

```

long int initlen = sysconf(_SC_GETGR_R_SIZE_MAX);
size_t len;
if (initlen == -1)
    /* Default initial length. */
    len = 1024;
else
    len = (size_t) initlen;
struct group result;
struct group *resultp;
char *buffer = malloc(len);
if (buffer == NULL)
    ...handle error...
int e;
while ((e = getgrgid_r(42, &result, buffer, len, &resultp)) == ERANGE)
{
    size_t newlen = 2 * len;
    if (newlen < len)
        ...handle error...
    len = newlen;
    char *newbuffer = realloc(buffer, len);
    if (newbuffer == NULL)
        ...handle error...
    buffer = newbuffer;
}
if (e != 0)
    ...handle error...
free (buffer);

```

Finding an Entry in the Group Database

The following example uses *getgrgid*() to search the group database for a group ID that was previously stored in a **stat** structure, then prints out the group name if it is found. If the group is not found, the program prints the numeric value of the group for the entry.

```

#include <sys/types.h>
#include <grp.h>
#include <stdio.h>
...
struct stat statbuf;
struct group *grp;
...
if ((grp = getgrgid(statbuf.st_gid)) != NULL)
    printf(" %-8.8s", grp->gr_name);
else
    printf(" %-8d", statbuf.st_gid);
...

```

APPLICATION USAGE

Applications wishing to check for error situations should set *errno* to 0 before calling *getgrgid()*. If *errno* is set on return, an error occurred.

The *getgrgid_r()* function is thread-safe and shall return values in a user-supplied buffer instead of possibly using a static data area that may be overwritten by each call.

Portable applications should take into account that it is usual for an implementation to return -1 from *sysconf()* indicating that there is no maximum for *_SC_GETGR_R_SIZE_MAX*.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

endgrent(), *getgrnam()*, *sysconf()*

XBD **<grp.h>**, **<sys/types.h>**

CHANGE HISTORY

First released in Issue 1. Derived from System V Release 2.0.

Issue 5

Normative text previously in the APPLICATION USAGE section is moved to the RETURN VALUE section.

The *getgrgid_r()* function is included for alignment with the POSIX Threads Extension.

A note indicating that the *getgrgid()* function need not be reentrant is added to the DESCRIPTION.

Issue 6

The *getgrgid_r()* function is marked as part of the Thread-Safe Functions option.

The Open Group Corrigendum U028/3 is applied, correcting text in the DESCRIPTION describing matching the *gid*.

In the DESCRIPTION, the note about reentrancy is expanded to cover thread-safety.

In the SYNOPSIS, the optional include of the **<sys/types.h>** header is removed.

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The requirement to include **<sys/types.h>** has been removed. Although **<sys/types.h>** was required for conforming implementations of previous POSIX specifications, it was not required for UNIX applications.
- In the RETURN VALUE section, the requirement to set *errno* on error is added.
- The [EIO], [EINTR], [EMFILE], and [ENFILE] optional error conditions are added.

The APPLICATION USAGE section is updated to include a note on the thread-safe function and its avoidance of possibly using a static data area.

IEEE PASC Interpretation 1003.1 #116 is applied, changing the description of the size of the buffer from *bufsize* characters to bytes.

34690 **Issue 7**

34691 Austin Group Interpretation 1003.1-2001 #156 is applied.

34692 SD5-XBD-ERN-4 is applied, changing the definition of the [EMFILE] error.

34693 SD5-XSH-ERN-166 is applied.

34694 The *getgrgid_r()* function is moved from the Thread-Safe Functions option to the Base.

34695 A minor addition is made to the EXAMPLES section, reminding the application developer to
34696 free memory allocated as if by *malloc()*.

34697 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0241 [75] is applied.

+

NAME

getgrnam, getgrnam_r — search group database for a name

SYNOPSIS

```
#include <grp.h>

struct group *getgrnam(const char *name);
int getgrnam_r(const char *name, struct group *grp, char *buffer,
               size_t bufsize, struct group **result);
```

DESCRIPTION

The *getgrnam()* function shall search the group database for an entry with a matching *name*.

The *getgrnam()* function need not be thread-safe.

The *getgrnam_r()* function shall update the **group** structure pointed to by *grp* and store a pointer to that structure at the location pointed to by *result*. The structure shall contain an entry from the group database with a matching *name*. Storage referenced by the **group** structure is allocated from the memory provided with the *buffer* parameter, which is *bufsize* bytes in size. A call to *sysconf(_SC_GETGR_R_SIZE_MAX)* returns either *-1* without changing *errno* or an initial value suggested for the size of this buffer. A null pointer is returned at the location pointed to by *result* on error or if the requested entry is not found.

RETURN VALUE

The *getgrnam()* function shall return a pointer to a **struct group** with the structure defined in **<grp.h>** with a matching entry if one is found. The *getgrnam()* function shall return a null pointer if either the requested entry was not found, or an error occurred. On error, *errno* shall be set to indicate the error.

The application shall not modify the structure to which the return value points, nor any storage areas pointed to by pointers within the structure. The returned pointer, and pointers within the structure, might be invalidated or the structure or the storage areas might be overwritten by a subsequent call to *getgrent()*, *getgrgid()*, or *getgrnam()*.

The *getgrnam_r()* function shall return zero on success or if the requested entry was not found and no error has occurred. If any error has occurred, an error number shall be returned to indicate the error.

ERRORS

The *getgrnam()* and *getgrnam_r()* functions may fail if:

[EIO]	An I/O error has occurred.
[EINTR]	A signal was caught during <i>getgrnam()</i> .
[EMFILE]	All file descriptors available to the process are currently open.
[ENFILE]	The maximum allowable number of files is currently open in the system.
- The *getgrnam_r()* function may fail if:
- | | |
|----------|--|
| [ERANGE] | Insufficient storage was supplied via <i>buffer</i> and <i>bufsize</i> to contain the data to be referenced by the resulting group structure. |
|----------|--|

EXAMPLES

Note that `sysconf(_SC_GETGR_R_SIZE_MAX)` may return `-1` if there is no hard limit on the size of the buffer needed to store all the groups returned. This example shows how an application can allocate a buffer of sufficient size to work with `getgrnam_r()`.

```

long int initlen = sysconf(_SC_GETGR_R_SIZE_MAX);
size_t len;
if (initlen == -1)
    /* Default initial length. */
    len = 1024;
else
    len = (size_t) initlen;
struct group result;
struct group *resultp;
char *buffer = malloc(len);
if (buffer == NULL)
    ...handle error...
int e;
while ((e = getgrnam_r("somegroup", &result, buffer, len, &resultp))
      == ERANGE)
{
    size_t newlen = 2 * len;
    if (newlen < len)
        ...handle error...
    len = newlen;
    char *newbuffer = realloc(buffer, len);
    if (newbuffer == NULL)
        ...handle error...
    buffer = newbuffer;
}
if (e != 0)
    ...handle error...
free (buffer);

```

APPLICATION USAGE

Applications wishing to check for error situations should set `errno` to 0 before calling `getgrnam()`. If `errno` is set on return, an error occurred.

The `getgrnam_r()` function is thread-safe and shall return values in a user-supplied buffer instead of possibly using a static data area that may be overwritten by each call.

Portable applications should take into account that it is usual for an implementation to return `-1` from `sysconf()` indicating that there is no maximum for `_SC_GETGR_R_SIZE_MAX`.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

`endgrent()`, `getgrgid()`, `sysconf()`

XBD `<grp.h>`, `<sys/types.h>`

CHANGE HISTORY

First released in Issue 1. Derived from System V Release 2.0.

Issue 5

Normative text previously in the APPLICATION USAGE section is moved to the RETURN VALUE section.

The `getgrnam_r()` function is included for alignment with the POSIX Threads Extension.

A note indicating that the `getgrnam()` function need not be reentrant is added to the DESCRIPTION.

Issue 6

The `getgrnam_r()` function is marked as part of the Thread-Safe Functions option.

In the DESCRIPTION, the note about reentrancy is expanded to cover thread-safety.

In the SYNOPSIS, the optional include of the `<sys/types.h>` header is removed.

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The requirement to include `<sys/types.h>` has been removed. Although `<sys/types.h>` was required for conforming implementations of previous POSIX specifications, it was not required for UNIX applications.
- In the RETURN VALUE section, the requirement to set `errno` on error is added.
- The [EIO], [EINTR], [EMFILE], and [ENFILE] optional error conditions are added.

The APPLICATION USAGE section is updated to include a note on the thread-safe function and its avoidance of possibly using a static data area.

IEEE PASC Interpretation 1003.1 #116 is applied, changing the description of the size of the buffer from *bufsize* characters to bytes.

Issue 7

Austin Group Interpretation 1003.1-2001 #081 is applied, clarifying the RETURN VALUE section.

Austin Group Interpretation 1003.1-2001 #156 is applied.

SD5-XBD-ERN-4 is applied, changing the definition of the [EMFILE] error.

SD5-XSH-ERN-166 is applied.

The `getgrnam_r()` function is moved from the Thread-Safe Functions option to the Base.

A minor addition is made to the EXAMPLES section, reminding the application developer to free memory allocated as if by `malloc()`.

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0242 [75] is applied.

+

34814 **NAME**

34815 getgroups — get supplementary group IDs

34816 **SYNOPSIS**

34817 #include <unistd.h>

34818 int getgroups(int *gidsetsize*, gid_t *grouplist*[]);34819 **DESCRIPTION**

34820 The *getgroups()* function shall fill in the array *grouplist* with the current supplementary group
 34821 IDs of the calling process. It is implementation-defined whether *getgroups()* also returns the
 34822 effective group ID in the *grouplist* array.

34823 The *gidsetsize* argument specifies the number of elements in the array *grouplist*. The actual
 34824 number of group IDs stored in the array shall be returned. The values of array entries with
 34825 indices greater than or equal to the value returned are undefined.

34826 If *gidsetsize* is 0, *getgroups()* shall return the number of group IDs that it would otherwise return
 34827 without modifying the array pointed to by *grouplist*.

34828 If the effective group ID of the process is returned with the supplementary group IDs, the value
 34829 returned shall always be greater than or equal to one and less than or equal to the value of
 34830 {NGROUPS_MAX}+1.

34831 **RETURN VALUE**

34832 Upon successful completion, the number of supplementary group IDs shall be returned. A
 34833 return value of -1 indicates failure and *errno* shall be set to indicate the error.

34834 **ERRORS**34835 The *getgroups()* function shall fail if:

34836 [EINVAL] The *gidsetsize* argument is non-zero and less than the number of group IDs
 34837 that would have been returned.

34838 **EXAMPLES**34839 **Getting the Supplementary Group IDs of the Calling Process**

34840 The following example places the current supplementary group IDs of the calling process into
 34841 the *group* array.

```
34842       #include <sys/types.h>
34843       #include <unistd.h>
34844       ...
34845       gid_t *group;
34846       int nogroups;
34847       long ngroups_max;

34848       ngroups_max = sysconf(_SC_NGROUPS_MAX) + 1;
34849       group = (gid_t *)malloc(ngroups_max * sizeof(gid_t));
34850       ngroups = getgroups(ngroups_max, group);
```

34851 **APPLICATION USAGE**

34852 None.

34853 **RATIONALE**

34854 The related function *setgroups()* is a privileged operation and therefore is not covered by this
 34855 volume of POSIX.1-2008.

34856 As implied by the definition of supplementary groups, the effective group ID may appear in the

array returned by *getgroups()* or it may be returned only by *getegid()*. Duplication may exist, but the application needs to call *getegid()* to be sure of getting all of the information. Various implementation variations and administrative sequences cause the set of groups appearing in the result of *getgroups()* to vary in order and as to whether the effective group ID is included, even when the set of groups is the same (in the mathematical sense of “set”). (The history of a process and its parents could affect the details of the result.)

Application developers should note that {NGROUPS_MAX} is not necessarily a constant on all implementations.

FUTURE DIRECTIONS

None.

SEE ALSO

getegid(), *setgid()*

XBD <sys/types.h>, <unistd.h>

CHANGE HISTORY

First released in Issue 3. Included for alignment with the POSIX.1-1988 standard.

Issue 5

Normative text previously in the APPLICATION USAGE section is moved to the DESCRIPTION.

Issue 6

In the SYNOPSIS, the optional include of the <sys/types.h> header is removed.

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The requirement to include <sys/types.h> has been removed. Although <sys/types.h> was required for conforming implementations of previous POSIX specifications, it was not required for UNIX applications.
- A return value of 0 is not permitted, because {NGROUPS_MAX} cannot be 0. This is a FIPS requirement.

The following changes were made to align with the IEEE P1003.1a draft standard:

- An explanation is added that the effective group ID may be included in the supplementary group list.

34887 **NAME**

34888 gethostent — network host database functions

34889 **SYNOPSIS**

34890 #include <netdb.h>

34891 struct hostent *gethostent(void);

34892 **DESCRIPTION**

34893 Refer to *endhostent()*.

34894 NAME

34895 gethostid — get an identifier for the current host

34896 SYNOPSIS

```
34897 XSI      #include <unistd.h>  
34898         long gethostid(void);
```

34899 DESCRIPTION

34900 The *gethostid()* function shall retrieve a 32-bit identifier for the current host.

34901 RETURN VALUE

34902 Upon successful completion, *gethostid()* shall return an identifier for the current host.

34903 ERRORS

34904 No errors are defined.

34905 EXAMPLES

34906 None.

34907 APPLICATION USAGE

34908 This volume of POSIX.1-2008 does not define the domain in which the return value is unique.

34909 RATIONALE

34910 None.

34911 FUTURE DIRECTIONS

34912 None.

34913 SEE ALSO

34914 *initstate()*

34915 XBD [<unistd.h>](#)

34916 CHANGE HISTORY

34917 First released in Issue 4, Version 2.

34918 Issue 5

34919 Moved from X/OPEN UNIX extension to BASE.

34920 **NAME**

34921 gethostname — get name of current host

34922 **SYNOPSIS**

34923 #include <unistd.h>

34924 int gethostname(char *name, size_t namelen);

34925 **DESCRIPTION**

34926 The *gethostname()* function shall return the standard host name for the current machine. The
34927 *namelen* argument shall specify the size of the array pointed to by the *name* argument. The
34928 returned name shall be null-terminated, except that if *namelen* is an insufficient length to hold
34929 the host name, then the returned name shall be truncated and it is unspecified whether the
34930 returned name is null-terminated.

34931 Host names are limited to {HOST_NAME_MAX} bytes.

34932 **RETURN VALUE**

34933 Upon successful completion, 0 shall be returned; otherwise, -1 shall be returned.

34934 **ERRORS**

34935 No errors are defined.

34936 **EXAMPLES**

34937 None.

34938 **APPLICATION USAGE**

34939 None.

34940 **RATIONALE**

34941 None.

34942 **FUTURE DIRECTIONS**

34943 None.

34944 **SEE ALSO**34945 *gethostid()*, *uname()*

34946 XBD <unistd.h>

34947 **CHANGE HISTORY**

34948 First released in Issue 6. Derived from the XNS, Issue 5.2 specification.

34949 The Open Group Base Resolution bwg2001-008 is applied, changing the *namelen* parameter from
34950 *socklen_t* to *size_t*.

NAME

getitimer, setitimer — get and set value of interval timer

SYNOPSIS

OB XSI `#include <sys/time.h>`

```
int getitimer(int which, struct itimerval *value);
int setitimer(int which, const struct itimerval *restrict value,
              struct itimerval *restrict ovalue);
```

DESCRIPTION

The *getitimer()* function shall store the current value of the timer specified by *which* into the structure pointed to by *value*. The *setitimer()* function shall set the timer specified by *which* to the value specified in the structure pointed to by *value*, and if *ovalue* is not a null pointer, store the previous value of the timer in the structure pointed to by *ovalue*.

A timer value is defined by the **itimerval** structure, specified in **<sys/time.h>**. If *it_value* is non-zero, it shall indicate the time to the next timer expiration. If *it_interval* is non-zero, it shall specify a value to be used in reloading *it_value* when the timer expires. Setting *it_value* to 0 shall disable a timer, regardless of the value of *it_interval*. Setting *it_interval* to 0 shall disable a timer after its next expiration (assuming *it_value* is non-zero).

Implementations may place limitations on the granularity of timer values. For each interval timer, if the requested timer value requires a finer granularity than the implementation supports, the actual timer value shall be rounded up to the next supported value.

An XSI-conforming implementation provides each process with at least three interval timers, which are indicated by the *which* argument:

ITIMER_PROF	Decrements both in process virtual time and when the system is running on behalf of the process. It is designed to be used by interpreters in statistically profiling the execution of interpreted programs. Each time the ITIMER_PROF timer expires, the SIGPROF signal is delivered.
ITIMER_REAL	Decrements in real time. A SIGALRM signal is delivered when this timer expires.
ITIMER_VIRTUAL	Decrements in process virtual time. It runs only when the process is executing. A SIGVTALRM signal is delivered when it expires.

The interaction between *setitimer()* and *alarm()* or *sleep()* is unspecified.

RETURN VALUE

Upon successful completion, *getitimer()* or *setitimer()* shall return 0; otherwise, -1 shall be returned and *errno* set to indicate the error.

ERRORS

The *setitimer()* function shall fail if:

[EINVAL]	The <i>value</i> argument is not in canonical form. (In canonical form, the number of microseconds is a non-negative integer less than 1 000 000 and the number of seconds is a non-negative integer.)

The *getitimer()* and *setitimer()* functions may fail if:

[EINVAL]	The <i>which</i> argument is not recognized.

34992 **EXAMPLES**

34993 None.

34994 **APPLICATION USAGE**

34995 Applications should use the *timer_gettime()* and *timer_settime()* functions instead of the
34996 obsolescent *gettimer()* and *setitimer()* functions, respectively.

34997 **RATIONALE**

34998 None.

34999 **FUTURE DIRECTIONS**35000 The *gettimer()* and *setitimer()* functions may be removed in a future version.35001 **SEE ALSO**35002 *alarm()*, *exec*, *sleep()*, *timer_getoverrun()*

35003 XBD <signal.h>, <sys/time.h>

35004 **CHANGE HISTORY**

35005 First released in Issue 4, Version 2.

35006 **Issue 5**

35007 Moved from X/OPEN UNIX extension to BASE.

35008 **Issue 6**

35009 The **restrict** keyword is added to the *setitimer()* prototype for alignment with the
35010 ISO/IEC 9899:1999 standard.

35011 **Issue 7**35012 The *gettimer()* and *setitimer()* functions are marked obsolescent.

35013 **NAME**35014 getline — read a delimited record from *stream*35015 **SYNOPSIS**

```
35016 CX      #include <stdio.h>
35017          ssize_t getline(char **restrict lineptr, size_t *restrict n,
35018                          FILE *restrict stream);
```

35019 **DESCRIPTION**35020 Refer to *getdelim()*.

NAME

getlogin, getlogin_r — get login name

SYNOPSIS

```
#include <unistd.h>
```

```
char *getlogin(void);
```

```
int getlogin_r(char *name, size_t namesize);
```

DESCRIPTION

The *getlogin()* function shall return a pointer to a string containing the user name associated by the login activity with the controlling terminal of the current process. If *getlogin()* returns a non-null pointer, then that pointer points to the name that the user logged in under, even if there are several login names with the same user ID.

The *getlogin()* function need not be thread-safe.

The *getlogin_r()* function shall put the name associated by the login activity with the controlling terminal of the current process in the character array pointed to by *name*. The array is *namesize* characters long and should have space for the name and the terminating null character. The maximum size of the login name is {LOGIN_NAME_MAX}.

If *getlogin_r()* is successful, *name* points to the name the user used at login, even if there are several login names with the same user ID.

The *getlogin()* and *getlogin_r()* functions may make use of file descriptors 0, 1, and 2 to find the controlling terminal of the current process, examining each in turn until the terminal is found. If in this case none of these three file descriptors is open to the controlling terminal, these functions may fail. The method used to find the terminal associated with a file descriptor may depend on the file descriptor being open to the actual terminal device, not **/dev/tty**.

RETURN VALUE

Upon successful completion, *getlogin()* shall return a pointer to the login name or a null pointer if the user's login name cannot be found. Otherwise, it shall return a null pointer and set *errno* to indicate the error.

The application shall not modify the string returned. The returned pointer might be invalidated or the string content might be overwritten by a subsequent call to *getlogin()*.

If successful, the *getlogin_r()* function shall return zero; otherwise, an error number shall be returned to indicate the error.

ERRORS

These functions may fail if:

[EMFILE] All file descriptors available to the process are currently open.

[ENFILE] The maximum allowable number of files is currently open in the system.

[ENOTTY] None of the file descriptors 0, 1, or 2 is open to the controlling terminal of the current process.

[ENXIO] The calling process has no controlling terminal.

The *getlogin_r()* function may fail if:

[ERANGE] The value of *namesize* is smaller than the length of the string to be returned including the terminating null character.

EXAMPLES**Getting the User Login Name**

The following example calls the *getlogin()* function to obtain the name of the user associated with the calling process, and passes this information to the *getpwnam()* function to get the associated user database information.

```
#include <unistd.h>
#include <sys/types.h>
#include <pwd.h>
#include <stdio.h>
...
char *lgn;
struct passwd *pw;
...
if ((lgn = getlogin()) == NULL || (pw = getpwnam(lgn)) == NULL) {
    fprintf(stderr, "Get of user information failed.\n"); exit(1);
}
```

APPLICATION USAGE

Three names associated with the current process can be determined: *getpwuid(geteuid())* shall return the name associated with the effective user ID of the process; *getlogin()* shall return the name associated with the current login activity; and *getpwuid(getuid())* shall return the name associated with the real user ID of the process.

The *getlogin_r()* function is thread-safe and returns values in a user-supplied buffer instead of possibly using a static data area that may be overwritten by each call.

RATIONALE

The *getlogin()* function returns a pointer to the user's login name. The same user ID may be shared by several login names. If it is desired to get the user database entry that is used during login, the result of *getlogin()* should be used to provide the argument to the *getpwnam()* function. (This might be used to determine the user's login shell, particularly where a single user has multiple login shells with distinct login names, but the same user ID.)

The information provided by the *cuserid()* function, which was originally defined in the POSIX.1-1988 standard and subsequently removed, can be obtained by the following:

```
getpwuid(geteuid())
```

while the information provided by historical implementations of *cuserid()* can be obtained by:

```
getpwuid(getuid())
```

The thread-safe version of this function places the user name in a user-supplied buffer and returns a non-zero value if it fails. The non-thread-safe version may return the name in a static data area that may be overwritten by each call.

FUTURE DIRECTIONS

None.

SEE ALSO

getpwnam(), *getpwuid()*, *geteuid()*, *getuid()*

XBD *<limits.h>*, *<unistd.h>*

CHANGE HISTORY

First released in Issue 1. Derived from System V Release 2.0.

Issue 5

Normative text previously in the APPLICATION USAGE section is moved to the RETURN VALUE section.

The *getlogin_r()* function is included for alignment with the POSIX Threads Extension.

A note indicating that the *getlogin()* function need not be reentrant is added to the DESCRIPTION.

Issue 6

The *getlogin_r()* function is marked as part of the Thread-Safe Functions option.

In the DESCRIPTION, the note about reentrancy is expanded to cover thread-safety.

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- In the RETURN VALUE section, the requirement to set *errno* on error is added.
- The [EMFILE], [ENFILE], and [ENXIO] optional error conditions are added.

The APPLICATION USAGE section is updated to include a note on the thread-safe function and its avoidance of possibly using a static data area.

Issue 7

Austin Group Interpretation 1003.1-2001 #156 is applied.

SD5-XBD-ERN-4 is applied, changing the definition of the [EMFILE] error.

The *getlogin_r()* function is moved from the Thread-Safe Functions option to the Base.

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0243 [172], XSH/TC1-2008/0244 [75], + and XSH/TC1-2008/0245 [172] are applied.

NAME

getmsg, getpmsg — receive next message from a STREAMS file (**STREAMS**)

SYNOPSIS

```
OB XSR #include <stropts.h>
```

```
int getmsg(int fildes, struct strbuf *restrict ctlptr,
           struct strbuf *restrict dataptr, int *restrict flagsp);
int getpmsg(int fildes, struct strbuf *restrict ctlptr,
            struct strbuf *restrict dataptr, int *restrict bandp,
            int *restrict flagsp);
```

DESCRIPTION

The *getmsg()* function shall retrieve the contents of a message located at the head of the STREAM head read queue associated with a STREAMS file and place the contents into one or more buffers. The message contains either a data part, a control part, or both. The data and control parts of the message shall be placed into separate buffers, as described below. The semantics of each part are defined by the originator of the message.

The *getpmsg()* function shall be equivalent to *getmsg()*, except that it provides finer control over the priority of the messages received. Except where noted, all requirements on *getmsg()* also pertain to *getpmsg()*.

The *fildes* argument specifies a file descriptor referencing a STREAMS-based file.

The *ctlptr* and *dataptr* arguments each point to a **strbuf** structure, in which the *buf* member points to a buffer in which the data or control information is to be placed, and the *maxlen* member indicates the maximum number of bytes this buffer can hold. On return, the *len* member shall contain the number of bytes of data or control information actually received. The *len* member shall be set to 0 if there is a zero-length control or data part and *len* shall be set to -1 if no data or control information is present in the message.

When *getmsg()* is called, *flagsp* should point to an integer that indicates the type of message the process is able to receive. This is described further below.

The *ctlptr* argument is used to hold the control part of the message, and *dataptr* is used to hold the data part of the message. If *ctlptr* (or *dataptr*) is a null pointer or the *maxlen* member is -1, the control (or data) part of the message shall not be processed and shall be left on the STREAM head read queue, and if the *ctlptr* (or *dataptr*) is not a null pointer, *len* shall be set to -1. If the *maxlen* member is set to 0 and there is a zero-length control (or data) part, that zero-length part shall be removed from the read queue and *len* shall be set to 0. If the *maxlen* member is set to 0 and there are more than 0 bytes of control (or data) information, that information shall be left on the read queue and *len* shall be set to 0. If the *maxlen* member in *ctlptr* (or *dataptr*) is less than the control (or data) part of the message, *maxlen* bytes shall be retrieved. In this case, the remainder of the message shall be left on the STREAM head read queue and a non-zero return value shall be provided.

By default, *getmsg()* shall process the first available message on the STREAM head read queue. However, a process may choose to retrieve only high-priority messages by setting the integer pointed to by *flagsp* to RS_HIPRI. In this case, *getmsg()* shall only process the next message if it is a high-priority message. When the integer pointed to by *flagsp* is 0, any available message shall be retrieved. In this case, on return, the integer pointed to by *flagsp* shall be set to RS_HIPRI if a high-priority message was retrieved, or 0 otherwise.

For *getpmsg()*, the flags are different. The *flagsp* argument points to a bitmask with the following mutually-exclusive flags defined: MSG_HIPRI, MSG_BAND, and MSG_ANY. Like *getmsg()*,

getpmsg() shall process the first available message on the STREAM head read queue. A process may choose to retrieve only high-priority messages by setting the integer pointed to by *flagsp* to MSG_HIPRI and the integer pointed to by *bandp* to 0. In this case, *getpmsg()* shall only process the next message if it is a high-priority message. In a similar manner, a process may choose to retrieve a message from a particular priority band by setting the integer pointed to by *flagsp* to MSG_BAND and the integer pointed to by *bandp* to the priority band of interest. In this case, *getpmsg()* shall only process the next message if it is in a priority band equal to, or greater than, the integer pointed to by *bandp*, or if it is a high-priority message. If a process wants to get the first message off the queue, the integer pointed to by *flagsp* should be set to MSG_ANY and the integer pointed to by *bandp* should be set to 0. On return, if the message retrieved was a high-priority message, the integer pointed to by *flagsp* shall be set to MSG_HIPRI and the integer pointed to by *bandp* shall be set to 0. Otherwise, the integer pointed to by *flagsp* shall be set to MSG_BAND and the integer pointed to by *bandp* shall be set to the priority band of the message.

If O_NONBLOCK is not set, *getmsg()* and *getpmsg()* shall block until a message of the type specified by *flagsp* is available at the front of the STREAM head read queue. If O_NONBLOCK is set and a message of the specified type is not present at the front of the read queue, *getmsg()* and *getpmsg()* shall fail and set *errno* to [EAGAIN].

If a hangup occurs on the STREAM from which messages are retrieved, *getmsg()* and *getpmsg()* shall continue to operate normally, as described above, until the STREAM head read queue is empty. Thereafter, they shall return 0 in the *len* members of *ctlptr* and *dataptr*.

RETURN VALUE

Upon successful completion, *getmsg()* and *getpmsg()* shall return a non-negative value. A value of 0 indicates that a full message was read successfully. A return value of MORECTL indicates that more control information is waiting for retrieval. A return value of MOREDATA indicates that more data is waiting for retrieval. A return value of the bitwise-logical OR of MORECTL and MOREDATA indicates that both types of information remain. Subsequent *getmsg()* and *getpmsg()* calls shall retrieve the remainder of the message. However, if a message of higher priority has come in on the STREAM head read queue, the next call to *getmsg()* or *getpmsg()* shall retrieve that higher-priority message before retrieving the remainder of the previous message.

If the high priority control part of the message is consumed, the message shall be placed back on the queue as a normal message of band 0. Subsequent *getmsg()* and *getpmsg()* calls shall retrieve the remainder of the message. If, however, a priority message arrives or already exists on the STREAM head, the subsequent call to *getmsg()* or *getpmsg()* shall retrieve the higher-priority message before retrieving the remainder of the message that was put back.

Upon failure, *getmsg()* and *getpmsg()* shall return -1 and set *errno* to indicate the error.

ERRORS

The *getmsg()* and *getpmsg()* functions shall fail if:

[EAGAIN]	The O_NONBLOCK flag is set and no messages are available.
[EBADF]	The <i>fildev</i> argument is not a valid file descriptor open for reading.
[EBADMSG]	The queued message to be read is not valid for <i>getmsg()</i> or <i>getpmsg()</i> or a pending file descriptor is at the STREAM head.
[EINTR]	A signal was caught during <i>getmsg()</i> or <i>getpmsg()</i> .
[EINVAL]	An illegal value was specified by <i>flagsp</i> , or the STREAM or multiplexer referenced by <i>fildev</i> is linked (directly or indirectly) downstream from a multiplexer.

35219 [ENOSTR] A STREAM is not associated with *fil*des.

35220 In addition, *getmsg()* and *getpmsg()* shall fail if the STREAM head had processed an
 35221 asynchronous error before the call. In this case, the value of *errno* does not reflect the result of
 35222 *getmsg()* or *getpmsg()* but reflects the prior error.

35223 EXAMPLES

35224 Getting Any Message

35225 In the following example, the value of *fd* is assumed to refer to an open STREAMS file. The call
 35226 to *getmsg()* retrieves any available message on the associated STREAM-head read queue,
 35227 returning control and data information to the buffers pointed to by *ctrlbuf* and *databuf*,
 35228 respectively.

```
35229 #include <stropts.h>
35230 ...
35231 int fd;
35232 char ctrlbuf[128];
35233 char databuf[512];
35234 struct strbuf ctrl;
35235 struct strbuf data;
35236 int flags = 0;
35237 int ret;

35238 ctrl.buf = ctrlbuf;
35239 ctrl.maxlen = sizeof(ctrlbuf);

35240 data.buf = databuf;
35241 data.maxlen = sizeof(databuf);

35242 ret = getmsg (fd, &ctrl, &data, &flags);
```

35243 Getting the First Message off the Queue

35244 In the following example, the call to *getpmsg()* retrieves the first available message on the
 35245 associated STREAM-head read queue.

```
35246 #include <stropts.h>
35247 ...
35248 int fd;
35249 char ctrlbuf[128];
35250 char databuf[512];
35251 struct strbuf ctrl;
35252 struct strbuf data;
35253 int band = 0;
35254 int flags = MSG_ANY;
35255 int ret;

35256 ctrl.buf = ctrlbuf;
35257 ctrl.maxlen = sizeof(ctrlbuf);

35258 data.buf = databuf;
35259 data.maxlen = sizeof(databuf);

35260 ret = getpmsg (fd, &ctrl, &data, &band, &flags);
```

35261 APPLICATION USAGE

35262 None.

35263 RATIONALE

35264 None.

35265 FUTURE DIRECTIONS

35266 The *getmsg()* and *getpmsg()* functions may be removed in a future version.

35267 SEE ALSO

35268 [Section 2.6](#) (on page 499), [poll\(\)](#), [putmsg\(\)](#), [read\(\)](#), [write\(\)](#)

35269 XBD [<stropts.h>](#)

35270 CHANGE HISTORY

35271 First released in Issue 4, Version 2.

35272 Issue 5

35273 Moved from X/OPEN UNIX extension to BASE.

35274 A paragraph regarding “high-priority control parts of messages” is added to the RETURN
35275 VALUE section.

35276 Issue 6

35277 This function is marked as part of the XSI STREAMS Option Group.

35278 The **restrict** keyword is added to the *getmsg()* and *getpmsg()* prototypes for alignment with the
35279 ISO/IEC 9899:1999 standard.

35280 Issue 7

35281 The *getmsg()* and *getpmsg()* functions are marked obsolescent.

NAME

getnameinfo — get name information

SYNOPSIS

```
#include <sys/socket.h>
#include <netdb.h>

int getnameinfo(const struct sockaddr *restrict sa, socklen_t salen,
                char *restrict node, socklen_t nodelen, char *restrict service,
                socklen_t servicelen, int flags);
```

DESCRIPTION

The *getnameinfo()* function shall translate a socket address to a node name and service location, all of which are defined as in *freeaddrinfo()*.

The *sa* argument points to a socket address structure to be translated. The *salen* argument contains the length of the address pointed to by *sa*.

IP6 If the socket address structure contains an IPv4-mapped IPv6 address or an IPv4-compatible IPv6 address, the implementation shall extract the embedded IPv4 address and lookup the node name for that IPv4 address.

If the address is the IPv6 unspecified address ("::"), a lookup shall not be performed and the behavior shall be the same as when the node's name cannot be located.

If the *node* argument is non-NULL and the *nodelen* argument is non-zero, then the *node* argument points to a buffer able to contain up to *nodelen* bytes that receives the node name as a null-terminated string. If the *node* argument is NULL or the *nodelen* argument is zero, the node name shall not be returned. If the node's name cannot be located, the numeric form of the address contained in the socket address structure pointed to by the *sa* argument is returned instead of its name.

If the *service* argument is non-NULL and the *servicelen* argument is non-zero, then the *service* argument points to a buffer able to contain up to *servicelen* bytes that receives the service name as a null-terminated string. If the *service* argument is NULL or the *servicelen* argument is zero, the service name shall not be returned. If the service's name cannot be located, the numeric form of the service address (for example, its port number) shall be returned instead of its name.

The *flags* argument is a flag that changes the default actions of the function. By default the fully-qualified domain name (FQDN) for the host shall be returned, but:

- If the flag bit NI_NOFQDN is set, only the node name portion of the FQDN shall be returned for local hosts.
- If the flag bit NI_NUMERICHOST is set, the numeric form of the address contained in the socket address structure pointed to by the *sa* argument shall be returned instead of its name.
- If the flag bit NI_NAMEREQD is set, an error shall be returned if the host's name cannot be located.
- If the flag bit NI_NUMERICSERV is set, the numeric form of the service address shall be returned (for example, its port number) instead of its name.
- If the flag bit NI_NUMERICSERVICE is set, the numeric form of the scope identifier shall be returned (for example, interface index) instead of its name. This flag shall be ignored if the *sa* argument is not an IPv6 address.

- If the flag bit NI_DGRAM is set, this indicates that the service is a datagram service (SOCK_DGRAM). The default behavior shall assume that the service is a stream service (SOCK_STREAM).

Notes:

1. The two NI_NUMERICxxx flags are required to support the `-n` flag that many commands provide.
2. The NI_DGRAM flag is required for the few AF_INET and AF_INET6 port numbers (for example, [512,514]) that represent different services for UDP and TCP.

The `getnameinfo()` function shall be thread-safe.

RETURN VALUE

A zero return value for `getnameinfo()` indicates successful completion; a non-zero return value indicates failure. The possible values for the failures are listed in the ERRORS section.

Upon successful completion, `getnameinfo()` shall return the *node* and *service* names, if requested, in the buffers provided. The returned names are always null-terminated strings.

ERRORS

The `getnameinfo()` function shall fail and return the corresponding value if:

[EAI_AGAIN]	The name could not be resolved at this time. Future attempts may succeed.
[EAI_BADFLAGS]	The <i>flags</i> had an invalid value.
[EAI_FAIL]	A non-recoverable error occurred.
[EAI_FAMILY]	The address family was not recognized or the address length was invalid for the specified family.
[EAI_MEMORY]	There was a memory allocation failure.
[EAI_NONAME]	The name does not resolve for the supplied parameters.
	NI_NAMEREQD is set and the host's name cannot be located, or both <i>nodename</i> and <i>servname</i> were null.
[EAI_OVERFLOW]	An argument buffer overflowed. The buffer pointed to by the <i>node</i> argument or the <i>service</i> argument was too small.
[EAI_SYSTEM]	A system error occurred. The error code can be found in <i>errno</i> .

EXAMPLES

None.

APPLICATION USAGE

If the returned values are to be used as part of any further name resolution (for example, passed to `getaddrinfo()`), applications should provide buffers large enough to store any result possible on the system.

Given the IPv4-mapped IPv6 address "`::ffff:1.2.3.4`", the implementation performs a lookup as if the socket address structure contains the IPv4 address "`1.2.3.4`".

The IPv6 unspecified address ("`:::`") and the IPv6 loopback address ("`:::1`") are not IPv4-compatible addresses.

35365 **RATIONALE**

35366 None.

35367 **FUTURE DIRECTIONS**

35368 None.

35369 **SEE ALSO**35370 *endservent()*, *freeaddrinfo()*, *gai_strerror()*, *inet_ntop()*, *socket()*35371 XBD **<netdb.h>**, **<sys/socket.h>**35372 **CHANGE HISTORY**

35373 First released in Issue 6. Derived from the XNS, Issue 5.2 specification.

35374 The **restrict** keyword is added to the *getnameinfo()* prototype for alignment with the
35375 ISO/IEC 9899:1999 standard.35376 IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/23 is applied, making various changes in
35377 the SYNOPSIS and DESCRIPTION for alignment with IPv6.35378 IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/24 is applied, adding the
35379 [EAI_OVERFLOW] error to the ERRORS section.35380 **Issue 7**35381 SD5-XSH-ERN-127 is applied, clarifying the behavior if the address is the IPv6 unspecified
35382 address.35383 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0246 [284] and XSH/TC1-2008/0247 +
35384 [285] are applied.

35385 NAME

35386 getnetbyaddr, getnetbyname, getnetent — network database functions

35387 SYNOPSIS

```
35388     #include <netdb.h>
35389     struct netent *getnetbyaddr(uint32_t net, int type);
35390     struct netent *getnetbyname(const char *name);
35391     struct netent *getnetent(void);
```

35392 DESCRIPTION

35393 Refer to *endnetent()*.

NAME

getopt, optarg, opterr, optind, optopt — command option parsing

SYNOPSIS

```
#include <unistd.h>

int getopt(int argc, char * const argv[], const char *optstring);
extern char *optarg;
extern int opterr, optind, optopt;
```

DESCRIPTION

The *getopt()* function is a command-line parser that shall follow Utility Syntax Guidelines 3, 4, 5, 6, 7, 9, and 10 in XBD [Section 12.2](#) (on page 215).

The parameters *argc* and *argv* are the argument count and argument array as passed to *main()* (see *exec()*). The argument *optstring* is a string of recognized option characters; if a character is followed by a <colon>, the option takes an argument. All option characters allowed by Utility Syntax Guideline 3 are allowed in *optstring*. The implementation may accept other characters as an extension.

The variable *optind* is the index of the next element of the *argv[]* vector to be processed. It shall be initialized to 1 by the system, and *getopt()* shall update it when it finishes with each element of *argv[]*. If the application sets *optind* to zero before calling *getopt()*, the behavior is unspecified. When an element of *argv[]* contains multiple option characters, it is unspecified how *getopt()* determines which options have already been processed.

The *getopt()* function shall return the next option character (if one is found) from *argv* that matches a character in *optstring*, if there is one that matches. If the option takes an argument, *getopt()* shall set the variable *optarg* to point to the option-argument as follows:

1. If the option was the last character in the string pointed to by an element of *argv*, then *optarg* shall contain the next element of *argv*, and *optind* shall be incremented by 2. If the resulting value of *optind* is greater than *argc*, this indicates a missing option-argument, and *getopt()* shall return an error indication.
2. Otherwise, *optarg* shall point to the string following the option character in that element of *argv*, and *optind* shall be incremented by 1.

If, when *getopt()* is called:

```
argv[optind]   is a null pointer
*argv[optind]  is not the character -
argv[optind]   points to the string "--"
```

getopt() shall return -1 without changing *optind*. If:

```
argv[optind]   points to the string "--"
```

getopt() shall return -1 after incrementing *optind*.

If *getopt()* encounters an option character that is not contained in *optstring*, it shall return the <question-mark> ('?') character. If it detects a missing option-argument, it shall return the <colon> character (':') if the first character of *optstring* was a <colon>, or a <question-mark> character ('?') otherwise. In either case, *getopt()* shall set the variable *optopt* to the option character that caused the error. If the application has not set the variable *opterr* to 0 and the first character of *optstring* is not a <colon>, *getopt()* shall also print a diagnostic message to *stderr* in the format specified for the *getopts* utility.

The *getopt()* function need not be thread-safe.

RETURN VALUE

The *getopt()* function shall return the next option character specified on the command line.

A <colon> (':') shall be returned if *getopt()* detects a missing argument and the first character of *optstring* was a <colon> (':').

A <question-mark> ('?') shall be returned if *getopt()* encounters an option character not in *optstring* or detects a missing argument and the first character of *optstring* was not a <colon> (':').

Otherwise, *getopt()* shall return -1 when all command line options are parsed.

ERRORS

If the application has not set the variable *opterr* to 0, the first character of *optstring* is not a <colon>, and a write error occurs while *getopt()* is printing a diagnostic message to *stderr*, then the error indicator for *stderr* shall be set; but *getopt()* shall still succeed and the value of *errno* after *getopt()* is unspecified.

EXAMPLES**Parsing Command Line Options**

The following code fragment shows how you might process the arguments for a utility that can take the mutually-exclusive options *a* and *b* and the options *f* and *o*, both of which require arguments:

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

int
main(int argc, char *argv[ ])
{
    int c;
    int bflg = 0, aflg = 0, errflg = 0;
    char *ifile;
    char *ofile;
    . . .
    while ((c = getopt(argc, argv, ":abf:o:")) != -1) {
        switch(c) {
            case 'a':
                if (bflg)
                    errflg++;
                else
                    aflg++;
                break;
            case 'b':
                if (aflg)
                    errflg++;
                else
                    bflg++;
                break;
            case 'f':
                ifile = optarg;
                break;
            case 'o':
```



```

35485         ofile = optarg;
35486         break;
35487     case ':':          /* -f or -o without operand */
35488         fprintf(stderr,
35489             "Option -%c requires an operand\n", optopt);
35490         errflg++;
35491         break;
35492     case '?':
35493         fprintf(stderr,
35494             "Unrecognized option: '-%c'\n", optopt);
35495         errflg++;
35496     }
35497 }
35498 if (errflg) {
35499     fprintf(stderr, "usage: . . . ");
35500     exit(2);
35501 }
35502 for ( ; optind < argc; optind++) {
35503     if (access(argv[optind], R_OK)) {
35504         . . .
35505     }

```

This code accepts any of the following as equivalent:

```

35507 cmd -ao arg path path
35508 cmd -a -o arg path path
35509 cmd -o arg -a path path
35510 cmd -a -o arg -- path path
35511 cmd -a -oarg path path
35512 cmd -aoarg path path

```

Selecting Options from the Command Line

The following example selects the type of database routines the user wants to use based on the *Options* argument.

```

35516 #include <unistd.h>
35517 #include <string.h>
35518 ...
35519 const char *Options = "hdbt1";
35520 ...
35521 int dbtype, c;
35522 char *st;
35523 ...
35524 dbtype = 0;
35525 while ((c = getopt(argc, argv, Options)) != -1) {
35526     if ((st = strchr(Options, c)) != NULL) {
35527         dbtype = st - Options;
35528         break;
35529     }
35530 }

```

APPLICATION USAGE

The *getopt()* function is only required to support option characters included in Utility Syntax Guideline 3. Many historical implementations of *getopt()* support other characters as options. This is an allowed extension, but applications that use extensions are not maximally portable. Note that support for multi-byte option characters is only possible when such characters can be represented as type **int**.

While *ferror(stderr)* may be used to detect failures to write a diagnostic to *stderr* when *getopt()* returns ' ?', the value of *errno* is unspecified in such a condition. Applications desiring more control over handling write failures should set *opterr* to 0 and independently perform output to *stderr*, rather than relying on *getopt()* to do the output.

RATIONALE

The *optopt* variable represents historical practice and allows the application to obtain the identity of the invalid option.

The description has been written to make it clear that *getopt()*, like the *getopts* utility, deals with option-arguments whether separated from the option by <blank> characters or not. Note that the requirements on *getopt()* and *getopts* are more stringent than the Utility Syntax Guidelines.

The *getopt()* function shall return -1, rather than EOF, so that <stdio.h> is not required.

The special significance of a <colon> as the first character of *optstring* makes *getopt()* consistent with the *getopts* utility. It allows an application to make a distinction between a missing argument and an incorrect option letter without having to examine the option letter. It is true that a missing argument can only be detected in one case, but that is a case that has to be considered.

FUTURE DIRECTIONS

None.

SEE ALSO

exec

XBD Section 12.2 (on page 215), <unistd.h>

XCU *getopts*

CHANGE HISTORY

First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 5

A note indicating that the *getopt()* function need not be reentrant is added to the DESCRIPTION.

Issue 6

IEEE PASC Interpretation 1003.2 #150 is applied.

Austin Group Interpretation 1003.1-2001 #156 is applied.

Issue 7

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0248 [318], XSH/TC1-2008/0249 [460], XSH/TC1-2008/0250 [189], XSH/TC1-2008/0251 [189], XSH/TC1-2008/0252 [189], and XSH/TC1-2008/0253 [460] are applied.

35570 **NAME**35571 `getpeername` — get the name of the peer socket35572 **SYNOPSIS**

```
35573 #include <sys/socket.h>
35574 int getpeername(int socket, struct sockaddr *restrict address,
35575                 socklen_t *restrict address_len);
```

35576 **DESCRIPTION**

35577 The `getpeername()` function shall retrieve the peer address of the specified socket, store this
 35578 address in the **sockaddr** structure pointed to by the *address* argument, and store the length of this
 35579 address in the object pointed to by the *address_len* argument.

35580 The *address_len* argument points to a **socklen_t** object which on input specifies the length of the +
 35581 supplied **sockaddr** structure, and on output specifies the length of the stored address. If the
 35582 actual length of the address is greater than the length of the supplied **sockaddr** structure, the
 35583 stored address shall be truncated.

35584 If the protocol permits connections by unbound clients, and the peer is not bound, then the
 35585 value stored in the object pointed to by *address* is unspecified.

35586 **RETURN VALUE**

35587 Upon successful completion, 0 shall be returned. Otherwise, -1 shall be returned and *errno* set to
 35588 indicate the error.

35589 **ERRORS**

35590 The `getpeername()` function shall fail if:

- 35591 [EBADF] The *socket* argument is not a valid file descriptor.
- 35592 [EINVAL] The socket has been shut down.
- 35593 [ENOTCONN] The socket is not connected or otherwise has not had the peer pre-specified.
- 35594 [ENOTSOCK] The *socket* argument does not refer to a socket.
- 35595 [EOPNOTSUPP] The operation is not supported for the socket protocol.

35596 The `getpeername()` function may fail if:

- 35597 [ENOBUFS] Insufficient resources were available in the system to complete the call.

35598 **EXAMPLES**

35599 None.

35600 **APPLICATION USAGE**

35601 None.

35602 **RATIONALE**

35603 None.

35604 **FUTURE DIRECTIONS**

35605 None.

35606 **SEE ALSO**

35607 [*accept\(\)*](#), [*bind\(\)*](#), [*getsockname\(\)*](#), [*socket\(\)*](#)

35608 XBD [*<sys/socket.h>*](#)

35609	CHANGE HISTORY	
35610	First released in Issue 6. Derived from the XNS, Issue 5.2 specification.	
35611	The restrict keyword is added to the <i>getpeername()</i> prototype for alignment with the	
35612	ISO/IEC 9899:1999 standard.	
35613	Issue 7	
35614	POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0254 [464] is applied.	+

35615 **NAME**

35616 getpgid — get the process group ID for a process

35617 **SYNOPSIS**

35618 #include <unistd.h>

35619 pid_t getpgid(pid_t pid);

35620 **DESCRIPTION**35621 The *getpgid()* function shall return the process group ID of the process whose process ID is equal
35622 to *pid*. If *pid* is equal to 0, *getpgid()* shall return the process group ID of the calling process.35623 **RETURN VALUE**35624 Upon successful completion, *getpgid()* shall return a process group ID. Otherwise, it shall return
35625 (**pid_t**)−1 and set *errno* to indicate the error.35626 **ERRORS**35627 The *getpgid()* function shall fail if:35628 [EPERM] The process whose process ID is equal to *pid* is not in the same session as the
35629 calling process, and the implementation does not allow access to the process
35630 group ID of that process from the calling process.35631 [ESRCH] There is no process with a process ID equal to *pid*.35632 The *getpgid()* function may fail if:35633 [EINVAL] The value of the *pid* argument is invalid.35634 **EXAMPLES**

35635 None.

35636 **APPLICATION USAGE**

35637 None.

35638 **RATIONALE**

35639 None.

35640 **FUTURE DIRECTIONS**

35641 None.

35642 **SEE ALSO**35643 *exec*, *fork()*, *getpgrp()*, *getpid()*, *getsid()*, *setpgid()*, *setsid()*

35644 XBD <unistd.h>

35645 **CHANGE HISTORY**

35646 First released in Issue 4, Version 2.

35647 **Issue 5**

35648 Moved from X/OPEN UNIX extension to BASE.

35649 **Issue 7**35650 The *getpgid()* function is moved from the XSI option to the Base.

NAME

getpgrp — get the process group ID of the calling process

SYNOPSIS

```
#include <unistd.h>

pid_t getpgrp(void);
```

DESCRIPTION

The *getpgrp()* function shall return the process group ID of the calling process.

RETURN VALUE

The *getpgrp()* function shall always be successful and no return value is reserved to indicate an error.

ERRORS

No errors are defined.

EXAMPLES

None.

APPLICATION USAGE

None.

RATIONALE

4.3 BSD provides a *getpgrp()* function that returns the process group ID for a specified process. Although this function supports job control, all known job control shells always specify the calling process with this function. Thus, the simpler System V *getpgrp()* suffices, and the added complexity of the 4.3 BSD *getpgrp()* is provided by the XSI extension *getpgid()*.

FUTURE DIRECTIONS

None.

SEE ALSO

exec, *fork()*, *getpgid()*, *getpid()*, *getppid()*, *kill()*, *setpgid()*, *setsid()*

XBD [**<sys/types.h>**](#), [**<unistd.h>**](#)

CHANGE HISTORY

First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6

In the SYNOPSIS, the optional include of the **<sys/types.h>** header is removed.

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The requirement to include **<sys/types.h>** has been removed. Although **<sys/types.h>** was required for conforming implementations of previous POSIX specifications, it was not required for UNIX applications.

35686 **NAME**35687 `getpid` — get the process ID35688 **SYNOPSIS**35689 `#include <unistd.h>`35690 `pid_t getpid(void);`35691 **DESCRIPTION**35692 The `getpid()` function shall return the process ID of the calling process.35693 **RETURN VALUE**35694 The `getpid()` function shall always be successful and no return value is reserved to indicate an
35695 error.35696 **ERRORS**

35697 No errors are defined.

35698 **EXAMPLES**

35699 None.

35700 **APPLICATION USAGE**

35701 None.

35702 **RATIONALE**

35703 None.

35704 **FUTURE DIRECTIONS**

35705 None.

35706 **SEE ALSO**35707 *`exec`, `fork()`, `getpgrp()`, `getppid()`, `kill()`, `mkdtemp()`, `setpgid()`, `setsid()`*35708 XBD **<sys/types.h>**, **<unistd.h>**35709 **CHANGE HISTORY**

35710 First released in Issue 1. Derived from Issue 1 of the SVID.

35711 **Issue 6**35712 In the SYNOPSIS, the optional include of the **<sys/types.h>** header is removed.35713 The following new requirements on POSIX implementations derive from alignment with the
35714 Single UNIX Specification:

- 35715 • The requirement to include **<sys/types.h>** has been removed. Although **<sys/types.h>** was
35716 required for conforming implementations of previous POSIX specifications, it was not
35717 required for UNIX applications.

35718 NAME

35719 getpmsg — receive next message from a STREAMS file

35720 SYNOPSIS

```
35721 OB XSI #include <stropts.h>
35722 int getpmsg(int fildes, struct strbuf *restrict ctlptr,
35723             struct strbuf *restrict dataptr, int *restrict bandp,
35724             int *restrict flagsp);
```

35725 DESCRIPTION

35726 Refer to *getmsg()*.

35727 NAME

35728 getppid — get the parent process ID

35729 SYNOPSIS

35730 #include <unistd.h>

35731 pid_t getppid(void);

35732 DESCRIPTION

35733 The *getppid()* function shall return the parent process ID of the calling process.

35734 RETURN VALUE

35735 The *getppid()* function shall always be successful and no return value is reserved to indicate an
35736 error.

35737 ERRORS

35738 No errors are defined.

35739 EXAMPLES

35740 None.

35741 APPLICATION USAGE

35742 None.

35743 RATIONALE

35744 None.

35745 FUTURE DIRECTIONS

35746 None.

35747 SEE ALSO

35748 *exec*, *fork()*, *getpgid()*, *getpgrp()*, *getpid()*, *kill()*, *setpgid()*, *setsid()*

35749 XBD <sys/types.h>, <unistd.h>

35750 CHANGE HISTORY

35751 First released in Issue 1. Derived from Issue 1 of the SVID.

35752 Issue 6

35753 In the SYNOPSIS, the optional include of the <sys/types.h> header is removed.

35754 The following new requirements on POSIX implementations derive from alignment with the
35755 Single UNIX Specification:

- 35756
 - The requirement to include <sys/types.h> has been removed. Although <sys/types.h> was
35757 required for conforming implementations of previous POSIX specifications, it was not
35758 required for UNIX applications.

NAME

getpriority, setpriority — get and set the nice value

SYNOPSIS

```

#include <sys/resource.h>

int getpriority(int which, id_t who);
int setpriority(int which, id_t who, int value);

```

DESCRIPTION

The *getpriority()* function shall obtain the nice value of a process, process group, or user. The *setpriority()* function shall set the nice value of a process, process group, or user to *value*+{NZERO}.

Target processes are specified by the values of the *which* and *who* arguments. The *which* argument may be one of the following values: PRIO_PROCESS, PRIO_PGRP, or PRIO_USER, indicating that the *who* argument is to be interpreted as a process ID, a process group ID, or an effective user ID, respectively. A 0 value for the *who* argument specifies the current process, process group, or user.

The nice value set with *setpriority()* shall be applied to the process. If the process is multi-threaded, the nice value shall affect all system scope threads in the process.

If more than one process is specified, *getpriority()* shall return value {NZERO} less than the lowest nice value pertaining to any of the specified processes, and *setpriority()* shall set the nice values of all of the specified processes to *value*+{NZERO}.

The default nice value is {NZERO}; lower nice values shall cause more favorable scheduling. While the range of valid nice values is [0,{NZERO}*2-1], implementations may enforce more restrictive limits. If *value*+{NZERO} is less than the system's lowest supported nice value, *setpriority()* shall set the nice value to the lowest supported value; if *value*+{NZERO} is greater than the system's highest supported nice value, *setpriority()* shall set the nice value to the highest supported value.

Only a process with appropriate privileges can lower its nice value.

PS|TPS

Any processes or threads using SCHED_FIFO or SCHED_RR shall be unaffected by a call to *setpriority()*. This is not considered an error. A process which subsequently reverts to SCHED_OTHER need not have its priority affected by such a *setpriority()* call.

The effect of changing the nice value may vary depending on the process-scheduling algorithm in effect.

Since *getpriority()* can return the value -1 upon successful completion, it is necessary to set *errno* to 0 prior to a call to *getpriority()*. If *getpriority()* returns the value -1, then *errno* can be checked to see if an error occurred or if the value is a legitimate nice value.

RETURN VALUE

Upon successful completion, *getpriority()* shall return an integer in the range -{NZERO} to {NZERO}-1. Otherwise, -1 shall be returned and *errno* set to indicate the error.

Upon successful completion, *setpriority()* shall return 0; otherwise, -1 shall be returned and *errno* set to indicate the error.

ERRORS

The *getpriority()* and *setpriority()* functions shall fail if:

[ESRCH] No process could be located using the *which* and *who* argument values specified.

[EINVAL] The value of the *which* argument was not recognized, or the value of the *who* argument is not a valid process ID, process group ID, or user ID.

In addition, *setpriority()* may fail if:

[EPERM] A process was located, but neither the real nor effective user ID of the executing process match the effective user ID of the process whose nice value is being changed.

[EACCES] A request was made to change the nice value to a lower numeric value and the current process does not have appropriate privileges.

EXAMPLES**Using getpriority()**

The following example returns the current scheduling priority for the process ID returned by the call to *getpid()*.

```
#include <sys/resource.h>
...
int which = PRIO_PROCESS;
id_t pid;
int ret;

pid = getpid();
ret = getpriority(which, pid);
```

Using setpriority()

The following example sets the priority for the current process ID to -20.

```
#include <sys/resource.h>
...
int which = PRIO_PROCESS;
id_t pid;
int priority = -20;
int ret;

pid = getpid();
ret = setpriority(which, pid, priority);
```

APPLICATION USAGE

The *getpriority()* and *setpriority()* functions work with an offset nice value (nice value -{NZERO}). The nice value is in the range [0,2*{NZERO} -1], while the return value for *getpriority()* and the third parameter for *setpriority()* are in the range [-{NZERO},{NZERO} -1].

RATIONALE

None.

35838 **FUTURE DIRECTIONS**

35839 None.

35840 **SEE ALSO**35841 *nice()*, *sched_get_priority_max()*, *sched_setscheduler()*

35842 XBD <sys/resource.h>

35843 **CHANGE HISTORY**

35844 First released in Issue 4, Version 2.

35845 **Issue 5**

35846 Moved from X/OPEN UNIX extension to BASE.

35847 The DESCRIPTION is reworded in terms of the nice value rather than *priority* to avoid confusion
35848 with functionality in the POSIX Realtime Extension.

35849 **NAME**

35850 getprotobyname, getprotobynumber, getprotoent — network protocol database functions

35851 **SYNOPSIS**

35852 #include <netdb.h>

35853 struct protoent *getprotobyname(const char *name);

35854 struct protoent *getprotobynumber(int proto);

35855 struct protoent *getprotoent(void);

35856 **DESCRIPTION**35857 Refer to *endprotoent()*.

35858 **NAME**

35859 getpwent — get user database entry

35860 **SYNOPSIS**

```
35861 XSI      #include <pwd.h>
35862          struct passwd *getpwent(void);
```

35863 **DESCRIPTION**

35864 Refer to *endpwent()*.

35865 **NAME**

35866 getpwnam, getpwnam_r — search user database for a name

35867 **SYNOPSIS**

```
35868     #include <pwd.h>
35869
35869     struct passwd *getpwnam(const char *name);
35870     int getpwnam_r(const char *name, struct passwd *pwd, char *buffer,
35871                   size_t bufsize, struct passwd **result);
```

35872 **DESCRIPTION**35873 The *getpwnam()* function shall search the user database for an entry with a matching *name*.35874 The *getpwnam()* function need not be thread-safe.

35875 Applications wishing to check for error situations should set *errno* to 0 before calling
 35876 *getpwnam()*. If *getpwnam()* returns a null pointer and *errno* is non-zero, an error occurred.

35877 The *getpwnam_r()* function shall update the **passwd** structure pointed to by *pwd* and store a
 35878 pointer to that structure at the location pointed to by *result*. The structure shall contain an entry
 35879 from the user database with a matching *name*. Storage referenced by the structure is allocated
 35880 from the memory provided with the *buffer* parameter, which is *bufsize* bytes in size. A call to
 35881 *sysconf(_SC_GETPW_R_SIZE_MAX)* returns either -1 without changing *errno* or an initial value
 35882 suggested for the size of this buffer. A null pointer shall be returned at the location pointed to
 35883 by *result* on error or if the requested entry is not found.

35884 **RETURN VALUE**

35885 The *getpwnam()* function shall return a pointer to a **struct passwd** with the structure as defined
 35886 in **<pwd.h>** with a matching entry if found. A null pointer shall be returned if the requested
 35887 entry is not found, or an error occurs. On error, *errno* shall be set to indicate the error.

35888 The application shall not modify the structure to which the return value points, nor any storage
 35889 areas pointed to by pointers within the structure. The returned pointer, and pointers within the
 35890 structure, might be invalidated or the structure or the storage areas might be overwritten by a
 35891 subsequent call to *getpwent()*, *getpwnam()*, or *getpwuid()*.

35892 The *getpwnam_r()* function shall return zero on success or if the requested entry was not found
 35893 and no error has occurred. If an error has occurred, an error number shall be returned to indicate
 35894 the error.

35895 **ERRORS**

35896 These functions may fail if:

35897	[EIO]	An I/O error has occurred.
35898	[EINTR]	A signal was caught during <i>getpwnam()</i> .
35899	[EMFILE]	All file descriptors available to the process are currently open.
35900	[ENFILE]	The maximum allowable number of files is currently open in the system.

35901 The *getpwnam_r()* function may fail if:

35902	[ERANGE]	Insufficient storage was supplied via <i>buffer</i> and <i>bufsize</i> to contain the data to be 35903 referenced by the resulting passwd structure.

EXAMPLES

Note that *sysconf*(*_SC_GETPW_R_SIZE_MAX*) may return *-1* if there is no hard limit on the size of the buffer needed to store all the groups returned. This example shows how an application can allocate a buffer of sufficient size to work with *getpwnam_r*().

```

long int initlen = sysconf(_SC_GETPW_R_SIZE_MAX);
size_t len;
if (initlen == -1)
    /* Default initial length. */
    len = 1024;
else
    len = (size_t) initlen;
struct passwd result;
struct passwd *resultp;
char *buffer = malloc(len);
if (buffer == NULL)
    ...handle error...
int e;
while ((e = getpwnam_r("someuser", &result, buffer, len, &resultp))
      == ERANGE)
{
    size_t newlen = 2 * len;
    if (newlen < len)
        ...handle error...
    len = newlen;
    char *newbuffer = realloc(buffer, len);
    if (newbuffer == NULL)
        ...handle error...
    buffer = newbuffer;
}
if (e != 0)
    ...handle error...
free (buffer);

```

Getting an Entry for the Login Name

The following example uses the *getlogin*() function to return the name of the user who logged in; this information is passed to the *getpwnam*() function to get the user database entry for that user.

```

#include <sys/types.h>
#include <pwd.h>
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
...
char *lgn;
struct passwd *pw;
...
if ((lgn = getlogin()) == NULL || (pw = getpwnam(lgn)) == NULL) {
    fprintf(stderr, "Get of user information failed.\n"); exit(1);
}
...

```


APPLICATION USAGE

Three names associated with the current process can be determined: *getpwuid(geteuid())* returns the name associated with the effective user ID of the process; *getlogin()* returns the name associated with the current login activity; and *getpwuid(getuid())* returns the name associated with the real user ID of the process.

The *getpwnam_r()* function is thread-safe and returns values in a user-supplied buffer instead of possibly using a static data area that may be overwritten by each call.

Portable applications should take into account that it is usual for an implementation to return `-1` from *sysconf()* indicating that there is no maximum for `_SC_GETPW_R_SIZE_MAX`.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

getpwuid(), *sysconf()*

XBD `<pwd.h>`, `<sys/types.h>`

CHANGE HISTORY

First released in Issue 1. Derived from System V Release 2.0.

Issue 5

Normative text previously in the APPLICATION USAGE section is moved to the RETURN VALUE section.

The *getpwnam_r()* function is included for alignment with the POSIX Threads Extension.

A note indicating that the *getpwnam()* function need not be reentrant is added to the DESCRIPTION.

Issue 6

The *getpwnam_r()* function is marked as part of the Thread-Safe Functions option.

The Open Group Corrigendum U028/3 is applied, correcting text in the DESCRIPTION describing matching the *name*.

In the SYNOPSIS, the optional include of the `<sys/types.h>` header is removed.

In the DESCRIPTION, the note about reentrancy is expanded to cover thread-safety.

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The requirement to include `<sys/types.h>` has been removed. Although `<sys/types.h>` was required for conforming implementations of previous POSIX specifications, it was not required for UNIX applications.
- In the RETURN VALUE section, the requirement to set *errno* on error is added.
- The [EMFILE], [ENFILE], and [ENXIO] optional error conditions are added.

The APPLICATION USAGE section is updated to include a note on the thread-safe function and its avoidance of possibly using a static data area.

IEEE PASC Interpretation 1003.1 #116 is applied, changing the description of the size of the buffer from *bufsize* characters to bytes.

35993 **Issue 7**

35994 Austin Group Interpretation 1003.1-2001 #156 is applied.

35995 SD5-XBD-ERN-4 is applied, changing the definition of the [EMFILE] error.

35996 SD5-XSH-ERN-166 is applied.

35997 The *getpwnam_r()* function is moved from the Thread-Safe Functions option to the Base.35998 A minor addition is made to the EXAMPLES section, reminding the application developer to
35999 free memory allocated as if by *malloc()*.

36000 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0255 [75,428] is applied.

+

36001 **NAME**36002 `getpwuid, getpwuid_r` — search user database for a user ID36003 **SYNOPSIS**

```
36004     #include <pwd.h>
36005     struct passwd *getpwuid(uid_t uid);
36006     int getpwuid_r(uid_t uid, struct passwd *pwd, char *buffer,
36007                   size_t bufsize, struct passwd **result);
```

36008 **DESCRIPTION**36009 The `getpwuid()` function shall search the user database for an entry with a matching *uid*.36010 The `getpwuid()` function need not be thread-safe.

36011 Applications wishing to check for error situations should set *errno* to 0 before calling `getpwuid()`.
 36012 If `getpwuid()` returns a null pointer and *errno* is set to non-zero, an error occurred.

36013 The `getpwuid_r()` function shall update the **passwd** structure pointed to by *pwd* and store a
 36014 pointer to that structure at the location pointed to by *result*. The structure shall contain an entry
 36015 from the user database with a matching *uid*. Storage referenced by the structure is allocated
 36016 from the memory provided with the *buffer* parameter, which is *bufsize* bytes in size. A call to
 36017 `sysconf(_SC_GETPW_R_SIZE_MAX)` returns either `-1` without changing *errno* or an initial value
 36018 suggested for the size of this buffer. A null pointer shall be returned at the location pointed to
 36019 by *result* on error or if the requested entry is not found.

36020 **RETURN VALUE**

36021 The `getpwuid()` function shall return a pointer to a **struct passwd** with the structure as defined in
 36022 `<pwd.h>` with a matching entry if found. A null pointer shall be returned if the requested entry
 36023 is not found, or an error occurs. On error, *errno* shall be set to indicate the error.

36024 The application shall not modify the structure to which the return value points, nor any storage |
 36025 areas pointed to by pointers within the structure. The returned pointer, and pointers within the |
 36026 structure, might be invalidated or the structure or the storage areas might be overwritten by a |
 36027 subsequent call to `getpwent()`, `getpwnam()`, or `getpwuid()`.

36028 If successful, the `getpwuid_r()` function shall return zero; otherwise, an error number shall be
 36029 returned to indicate the error.

36030 **ERRORS**

36031 These functions may fail if:

36032	[EIO]	An I/O error has occurred.
36033	[EINTR]	A signal was caught during <code>getpwuid()</code> .
36034	[EMFILE]	All file descriptors available to the process are currently open.
36035	[ENFILE]	The maximum allowable number of files is currently open in the system.

36036 The `getpwuid_r()` function may fail if:

36037	[ERANGE]	Insufficient storage was supplied via <i>buffer</i> and <i>bufsize</i> to contain the data to be 36038 referenced by the resulting passwd structure.

EXAMPLES

Note that `sysconf(_SC_GETPW_R_SIZE_MAX)` may return `-1` if there is no hard limit on the size of the buffer needed to store all the groups returned. This example shows how an application can allocate a buffer of sufficient size to work with `getpwuid_r()`.

```

36043 long int initlen = sysconf(_SC_GETPW_R_SIZE_MAX);
36044 size_t len;
36045 if (initlen == -1)
36046     /* Default initial length. */
36047     len = 1024;
36048 else
36049     len = (size_t) initlen;
36050 struct passwd result;
36051 struct passwd *resultp;
36052 char *buffer = malloc(len);
36053 if (buffer == NULL)
36054     ...handle error...
36055 int e;
36056 while ((e = getpwuid_r(42, &result, buffer, len, &resultp)) == ERANGE)
36057 {
36058     size_t newlen = 2 * len;
36059     if (newlen < len)
36060         ...handle error...
36061     len = newlen;
36062     char *newbuffer = realloc(buffer, len);
36063     if (newbuffer == NULL)
36064         ...handle error...
36065     buffer = newbuffer;
36066 }
36067 if (e != 0)
36068     ...handle error...
36069 free (buffer);

```

Getting an Entry for the Root User

The following example gets the user database entry for the user with user ID 0 (root).

```

36072 #include <sys/types.h>
36073 #include <pwd.h>
36074 ...
36075 uid_t id = 0;
36076 struct passwd *pwd;
36077 pwd = getpwuid(id);

```

Finding the Name for the Effective User ID

The following example defines *pws* as a pointer to a structure of type **passwd**, which is used to store the structure pointer returned by the call to the *getpwuid()* function. The *geteuid()* function shall return the effective user ID of the calling process; this is used as the search criteria for the *getpwuid()* function. The call to *getpwuid()* shall return a pointer to the structure containing that user ID value.

```
#include <unistd.h>
#include <sys/types.h>
#include <pwd.h>
...
struct passwd *pws;
pws = getpwuid(geteuid());
```

Finding an Entry in the User Database

The following example uses *getpwuid()* to search the user database for a user ID that was previously stored in a **stat** structure, then prints out the user name if it is found. If the user is not found, the program prints the numeric value of the user ID for the entry.

```
#include <sys/types.h>
#include <pwd.h>
#include <stdio.h>
...
struct stat statbuf;
struct passwd *pwd;
...
if ((pwd = getpwuid(statbuf.st_uid)) != NULL)
    printf(" %-8.8s", pwd->pw_name);
else
    printf(" %-8d", statbuf.st_uid);
```

APPLICATION USAGE

Three names associated with the current process can be determined: *getpwuid(geteuid())* returns the name associated with the effective user ID of the process; *getlogin()* returns the name associated with the current login activity; and *getpwuid(getuid())* returns the name associated with the real user ID of the process.

The *getpwuid_r()* function is thread-safe and returns values in a user-supplied buffer instead of possibly using a static data area that may be overwritten by each call.

Portable applications should take into account that it is usual for an implementation to return `-1` from *sysconf()* indicating that there is no maximum for `_SC_GETPW_R_SIZE_MAX`.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

getpwnam(), *geteuid()*, *getuid()*, *getlogin()*, *sysconf()*

XBD **<pwd.h>**, **<sys/types.h>**

CHANGE HISTORY

First released in Issue 1. Derived from System V Release 2.0.

Issue 5

Normative text previously in the APPLICATION USAGE section is moved to the RETURN VALUE section.

The *getpwuid_r()* function is included for alignment with the POSIX Threads Extension.

A note indicating that the *getpwuid()* function need not be reentrant is added to the DESCRIPTION.

Issue 6

The *getpwuid_r()* function is marked as part of the Thread-Safe Functions option.

The Open Group Corrigendum U028/3 is applied, correcting text in the DESCRIPTION describing matching the *uid*.

In the SYNOPSIS, the optional include of the **<sys/types.h>** header is removed.

In the DESCRIPTION, the note about reentrancy is expanded to cover thread-safety.

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The requirement to include **<sys/types.h>** has been removed. Although **<sys/types.h>** was required for conforming implementations of previous POSIX specifications, it was not required for UNIX applications.
- In the RETURN VALUE section, the requirement to set *errno* on error is added.
- The [EIO], [EINTR], [EMFILE], and [ENFILE] optional error conditions are added.

The APPLICATION USAGE section is updated to include a note on the thread-safe function and its avoidance of possibly using a static data area.

IEEE PASC Interpretation 1003.1 #116 is applied, changing the description of the size of the buffer from *bufsize* characters to bytes.

Issue 7

Austin Group Interpretation 1003.1-2001 #156 is applied.

SD5-XBD-ERN-4 is applied, changing the definition of the [EMFILE] error.

SD5-XSH-ERN-166 is applied.

The *getpwuid_r()* function is moved from the Thread-Safe Functions option to the Base.

A minor addition is made to the EXAMPLES section, reminding the application developer to free memory allocated as if by *malloc()*.

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0256 [75] is applied.

+

36154 **NAME**

36155 getrlimit, setrlimit — control maximum resource consumption

36156 **SYNOPSIS**

```

36157 XSI    #include <sys/resource.h>
36158
36158     int getrlimit(int resource, struct rlimit *rlp);
36159     int setrlimit(int resource, const struct rlimit *rlp);

```

36160 **DESCRIPTION**

36161 The *getrlimit()* function shall get, and the *setrlimit()* function shall set, limits on the consumption
 36162 of a variety of resources.

36163 Each call to either *getrlimit()* or *setrlimit()* identifies a specific resource to be operated upon as
 36164 well as a resource limit. A resource limit is represented by an **rlimit** structure. The *rlim_cur*
 36165 member specifies the current or soft limit and the *rlim_max* member specifies the maximum or
 36166 hard limit. Soft limits may be changed by a process to any value that is less than or equal to the
 36167 hard limit. A process may (irreversibly) lower its hard limit to any value that is greater than or
 36168 equal to the soft limit. Only a process with appropriate privileges can raise a hard limit. Both
 36169 hard and soft limits can be changed in a single call to *setrlimit()* subject to the constraints
 36170 described above.

36171 The value RLIM_INFINITY, defined in **<sys/resource.h>**, shall be considered to be larger than
 36172 any other limit value. If a call to *getrlimit()* returns RLIM_INFINITY for a resource, it means the
 36173 implementation shall not enforce limits on that resource. Specifying RLIM_INFINITY as any
 36174 resource limit value on a successful call to *setrlimit()* shall inhibit enforcement of that resource
 36175 limit.

36176 The following resources are defined:

36177	RLIMIT_CORE	This is the maximum size of a core file, in bytes, that may be created by a process. A limit of 0 shall prevent the creation of a core file. If this limit is exceeded, the writing of a core file shall terminate at this size.
36178		
36179		
36180	RLIMIT_CPU	This is the maximum amount of CPU time, in seconds, used by a process. If this limit is exceeded, SIGXCPU shall be generated for the process. If the process is catching or ignoring SIGXCPU, or all threads belonging to that process are blocking SIGXCPU, the behavior is unspecified.
36181		
36182		
36183		
36184	RLIMIT_DATA	This is the maximum size of a data segment of the process, in bytes. If this limit is exceeded, the <i>malloc()</i> function shall fail with <i>errno</i> set to [ENOMEM].
36185		
36186		
36187	RLIMIT_FSIZE	This is the maximum size of a file, in bytes, that may be created by a process. If a write or truncate operation would cause this limit to be exceeded, SIGXFSZ shall be generated for the thread. If the thread is blocking, or the process is catching or ignoring SIGXFSZ, continued attempts to increase the size of a file from end-of-file to beyond the limit shall fail with <i>errno</i> set to [EFBIG].
36188		
36189		
36190		
36191		
36192		
36193	RLIMIT_NOFILE	This is a number one greater than the maximum value that the system may assign to a newly-created descriptor. If this limit is exceeded, functions that allocate a file descriptor shall fail with <i>errno</i> set to [EMFILE]. This limit constrains the number of file descriptors that a process may allocate.
36194		
36195		
36196		
36197		

36198	RLIMIT_STACK	This is the maximum size of the initial thread's stack, in bytes. The implementation does not automatically grow the stack beyond this limit.
36199		If this limit is exceeded, SIGSEGV shall be generated for the thread. If the
36200		thread is blocking SIGSEGV, or the process is ignoring or catching
36201		SIGSEGV and has not made arrangements to use an alternate stack, the
36202		disposition of SIGSEGV shall be set to SIG_DFL before it is generated.
36203		
36204	RLIMIT_AS	This is the maximum size of total available memory of the process, in
36205		bytes. If this limit is exceeded, the <i>malloc()</i> and <i>mmap()</i> functions shall fail
36206		with <i>errno</i> set to [ENOMEM]. In addition, the automatic stack growth
36207		fails with the effects outlined above.
36208	When using the <i>getrlimit()</i> function, if a resource limit can be represented correctly in an object	
36209	of type rlim_t , then its representation is returned; otherwise, if the value of the resource limit is	
36210	equal to that of the corresponding saved hard limit, the value returned shall be	
36211	RLIM_SAVED_MAX; otherwise, the value returned shall be RLIM_SAVED_CUR.	
36212	When using the <i>setrlimit()</i> function, if the requested new limit is RLIM_INFINITY, the new limit	
36213	shall be "no limit"; otherwise, if the requested new limit is RLIM_SAVED_MAX, the new limit	
36214	shall be the corresponding saved hard limit; otherwise, if the requested new limit is	
36215	RLIM_SAVED_CUR, the new limit shall be the corresponding saved soft limit; otherwise, the	
36216	new limit shall be the requested value. In addition, if the corresponding saved limit can be	
36217	represented correctly in an object of type rlim_t then it shall be overwritten with the new limit.	
36218	The result of setting a limit to RLIM_SAVED_MAX or RLIM_SAVED_CUR is unspecified unless	
36219	a previous call to <i>getrlimit()</i> returned that value as the soft or hard limit for the corresponding	
36220	resource limit.	
36221	The determination of whether a limit can be correctly represented in an object of type rlim_t is	
36222	implementation-defined. For example, some implementations permit a limit whose value is	
36223	greater than RLIM_INFINITY and others do not.	
36224	The <i>exec</i> family of functions shall cause resource limits to be saved.	
36225	RETURN VALUE	
36226	Upon successful completion, <i>getrlimit()</i> and <i>setrlimit()</i> shall return 0. Otherwise, these functions	
36227	shall return -1 and set <i>errno</i> to indicate the error.	
36228	ERRORS	
36229	The <i>getrlimit()</i> and <i>setrlimit()</i> functions shall fail if:	
36230	[EINVAL]	An invalid <i>resource</i> was specified; or in a <i>setrlimit()</i> call, the new <i>rlim_cur</i>
36231		exceeds the new <i>rlim_max</i> .
36232	[EPERM]	The limit specified to <i>setrlimit()</i> would have raised the maximum limit value,
36233		and the calling process does not have appropriate privileges.
36234	The <i>setrlimit()</i> function may fail if:	
36235	[EINVAL]	The limit specified cannot be lowered because current usage is already higher
36236		than the limit.

EXAMPLES

None.

APPLICATION USAGE

If a process attempts to set the hard limit or soft limit for RLIMIT_NOFILE to less than the value of {_POSIX_OPEN_MAX} from <limits.h>, unexpected behavior may occur.

If a process attempts to set the hard limit or soft limit for RLIMIT_NOFILE to less than the highest currently open file descriptor +1, unexpected behavior may occur.

RATIONALE

It should be noted that RLIMIT_STACK applies “at least” to the stack of the initial thread in the process, and not to the sum of all the stacks in the process, as that would be very limiting unless the value is so big as to provide no value at all with a single thread.

FUTURE DIRECTIONS

None.

SEE ALSO

exec, fork(), malloc(), open(), sigaltstack(), sysconf(), ulimit()

XBD <stropts.h>, <sys/resource.h>

CHANGE HISTORY

First released in Issue 4, Version 2.

Issue 5

Moved from X/OPEN UNIX extension to BASE.

An APPLICATION USAGE section is added.

Large File Summit extensions are added.

Issue 6

IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/25 is applied, changing wording for RLIMIT_NOFILE in the DESCRIPTION related to functions that allocate a file descriptor failing with [EMFILE]. Text is added to the APPLICATION USAGE section noting the consequences of a process attempting to set the hard or soft limit for RLIMIT_NOFILE less than the highest currently open file descriptor +1.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/46 is applied, updating the definition of RLIMIT_STACK in the DESCRIPTION from “the maximum size of a process stack” to “the maximum size of the initial thread’s stack”. Text is added to the RATIONALE section.

NAME

getrusage — get information about resource utilization

SYNOPSIS

```
#include <sys/resource.h>

int getrusage(int who, struct rusage *r_usage);
```

DESCRIPTION

The *getrusage()* function shall provide measures of the resources used by the current process or its terminated and waited-for child processes. If the value of the *who* argument is `RUSAGE_SELF`, information shall be returned about resources used by the current process. If the value of the *who* argument is `RUSAGE_CHILDREN`, information shall be returned about resources used by the terminated and waited-for children of the current process. If the child is never waited for (for example, if the parent has `SA_NOCLDWAIT` set or sets `SIGCHLD` to `SIG_IGN`), the resource information for the child process is discarded and not included in the resource information provided by *getrusage()*.

The *r_usage* argument is a pointer to an object of type **struct rusage** in which the returned information is stored.

RETURN VALUE

Upon successful completion, *getrusage()* shall return 0; otherwise, `-1` shall be returned and *errno* set to indicate the error.

ERRORS

The *getrusage()* function shall fail if:

[EINVAL] The value of the *who* argument is not valid.

EXAMPLES**Using getrusage()**

The following example returns information about the resources used by the current process.

```
#include <sys/resource.h>
...
int who = RUSAGE_SELF;
struct rusage usage;
int ret;

ret = getrusage(who, &usage);
```

APPLICATION USAGE

None.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

exit(), *sigaction()*, *time()*, *times()*, *wait()*

XBD [<sys/resource.h>](#)

36308	CHANGE HISTORY
36309	First released in Issue 4, Version 2.
36310	Issue 5
36311	Moved from X/OPEN UNIX extension to BASE.

36312 **NAME**36313 gets — get a string from a *stdin* stream36314 **SYNOPSIS**

```
36315 OB      #include <stdio.h>
36316      char *gets(char *s);
```

36317 **DESCRIPTION**

36318 CX The functionality described on this reference page is aligned with the ISO C standard. Any
 36319 conflict between the requirements described here and the ISO C standard is unintentional. This
 36320 volume of POSIX.1-2008 defers to the ISO C standard.

36321 The *gets()* function shall read bytes from the standard input stream, *stdin*, into the array pointed
 36322 to by *s*, until a <newline> is read or an end-of-file condition is encountered. Any <newline> shall
 36323 be discarded and a null byte shall be placed immediately after the last byte read into the array.

36324 CX The *gets()* function may mark the last data access timestamp of the file associated with *stream* for
 36325 update. The last data access timestamp shall be marked for update by the first successful
 36326 execution of *fgetc()*, *fgets()*, *fread()*, *fscanf()*, *getc()*, *getchar()*, *getdelim()*, *getline()*, *gets()*, or
 36327 *scanf()* using *stream* that returns data not supplied by a prior call to *ungetc()*.

36328 **RETURN VALUE**

36329 Upon successful completion, *gets()* shall return *s*. If the end-of-file indicator for the stream is set,
 36330 or if the stream is at end-of-file, the end-of-file indicator for the stream shall be set and *gets()*
 36331 shall return a null pointer. If a read error occurs, the error indicator for the stream shall be set,
 36332 CX *gets()* shall return a null pointer, and set *errno* to indicate the error.

36333 **ERRORS**36334 Refer to *fgetc()*.36335 **EXAMPLES**

36336 None.

36337 **APPLICATION USAGE**

36338 Reading a line that overflows the array pointed to by *s* results in undefined behavior. The use of
 36339 *fgets()* is recommended.

36340 Since the user cannot specify the length of the buffer passed to *gets()*, use of this function is
 36341 discouraged. The length of the string read is unlimited. It is possible to overflow this buffer in
 36342 such a way as to cause applications to fail, or possible system security violations.

36343 Applications should use the *fgets()* function instead of the obsolescent *gets()* function.

36344 **RATIONALE**

36345 The standard developers decided to mark the *gets()* function as obsolescent even though it is in
 36346 the ISO C standard due to the possibility of buffer overflow.

36347 **FUTURE DIRECTIONS**

36348 The *gets()* function may be removed in a future version.

36349 **SEE ALSO**36350 [Section 2.5](#) (on page 495), [feof\(\)](#), [ferror\(\)](#), [fgets\(\)](#)36351 XBD [<stdio.h>](#)

CHANGE HISTORY

36352 First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6

36354 Extensions beyond the ISO C standard are marked.

Issue 7

36356 Austin Group Interpretation 1003.1-2001 #051 is applied, clarifying the RETURN VALUE section.

36357 The *gets()* function is marked obsolescent.

36358 Changes are made related to support for finegrained timestamps.

36359 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0257 [14] is applied.

+

36361 NAME

36362 getservbyname, getservbyport, getservent — network services database functions

36363 SYNOPSIS

36364 #include <netdb.h>

36365 struct servent *getservbyname(const char *name, const char *proto);

36366 struct servent *getservbyport(int port, const char *proto);

36367 struct servent *getservent(void);

36368 DESCRIPTION

36369 Refer to *endservent()*.

36370 **NAME**

36371 getsid — get the process group ID of a session leader

36372 **SYNOPSIS**

36373 #include <unistd.h>

36374 pid_t getsid(pid_t pid);

36375 **DESCRIPTION**

36376 The *getsid()* function shall obtain the process group ID of the process that is the session leader of
 36377 the process specified by *pid*. If *pid* is (**pid_t**)0, it specifies the calling process.

36378 **RETURN VALUE**

36379 Upon successful completion, *getsid()* shall return the process group ID of the session leader of
 36380 the specified process. Otherwise, it shall return **-1** and set *errno* to indicate the error. |

36381 **ERRORS**36382 The *getsid()* function shall fail if:

36383 [EPERM] The process specified by *pid* is not in the same session as the calling process,
 36384 and the implementation does not allow access to the process group ID of the
 36385 session leader of that process from the calling process.

36386 [ESRCH] There is no process with a process ID equal to *pid*.

36387 **EXAMPLES**

36388 None.

36389 **APPLICATION USAGE**

36390 None.

36391 **RATIONALE**

36392 None.

36393 **FUTURE DIRECTIONS**

36394 None.

36395 **SEE ALSO**36396 *exec*, *fork()*, *getpid()*, *getpgid()*, *setpgid()*, *setsid()*

36397 XBD <unistd.h>

36398 **CHANGE HISTORY**

36399 First released in Issue 4, Version 2.

36400 **Issue 5**

36401 Moved from X/OPEN UNIX extension to BASE.

36402 **Issue 7**36403 The *getsid()* function is moved from the XSI option to the Base.

36404 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0258 [421] is applied. +

NAME

getsockname — get the socket name

SYNOPSIS

```
#include <sys/socket.h>
```

```
int getsockname(int socket, struct sockaddr *restrict address,
                socklen_t *restrict address_len);
```

DESCRIPTION

The *getsockname()* function shall retrieve the locally-bound name of the specified socket, store this address in the **sockaddr** structure pointed to by the *address* argument, and store the length of this address in the object pointed to by the *address_len* argument.

The *address_len* argument points to a **socklen_t** object which on input specifies the length of the supplied **sockaddr** structure, and on output specifies the length of the stored address. If the actual length of the address is greater than the length of the supplied **sockaddr** structure, the stored address shall be truncated.

If the socket has not been bound to a local name, the value stored in the object pointed to by *address* is unspecified.

RETURN VALUE

Upon successful completion, 0 shall be returned, the *address* argument shall point to the address of the socket, and the *address_len* argument shall point to the length of the address. Otherwise, -1 shall be returned and *errno* set to indicate the error.

ERRORS

The *getsockname()* function shall fail if:

[EBADF] The *socket* argument is not a valid file descriptor.

[ENOTSOCK] The *socket* argument does not refer to a socket.

[EOPNOTSUPP] The operation is not supported for this socket's protocol.

The *getsockname()* function may fail if:

[EINVAL] The socket has been shut down.

[ENOBUFS] Insufficient resources were available in the system to complete the function.

EXAMPLES

None.

APPLICATION USAGE

None.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

accept(), *bind()*, *getpeername()*, *socket()*

XBD *<sys/socket.h>*

CHANGE HISTORY

36444 First released in Issue 6. Derived from the XNS, Issue 5.2 specification.

36446 The **restrict** keyword is added to the *getsockname()* prototype for alignment with the
36447 ISO/IEC 9899:1999 standard.

Issue 7

36448 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0259 [464] is applied.

+

NAME

getsockopt — get the socket options

SYNOPSIS

```
#include <sys/socket.h>

int getsockopt(int socket, int level, int option_name,
               void *restrict option_value, socklen_t *restrict option_len);
```

DESCRIPTION

The *getsockopt()* function manipulates options associated with a socket.

The *getsockopt()* function shall retrieve the value for the option specified by the *option_name* argument for the socket specified by the *socket* argument. If the size of the option value is greater than *option_len*, the value stored in the object pointed to by the *option_value* argument shall be silently truncated. Otherwise, the object pointed to by the *option_len* argument shall be modified to indicate the actual length of the value.

The *level* argument specifies the protocol level at which the option resides. To retrieve options at the socket level, specify the *level* argument as SOL_SOCKET. To retrieve options at other levels, supply the appropriate level identifier for the protocol controlling the option. For example, to indicate that an option is interpreted by the TCP (Transmission Control Protocol), set *level* to IPPROTO_TCP as defined in the **<netinet/in.h>** header.

The socket in use may require the process to have appropriate privileges to use the *getsockopt()* function.

The *option_name* argument specifies a single option to be retrieved. It can be one of the socket-level options defined in **<sys/socket.h>** and described in [Section 2.10.16](#) (on page 527).

RETURN VALUE

Upon successful completion, *getsockopt()* shall return 0; otherwise, -1 shall be returned and *errno* set to indicate the error.

ERRORS

The *getsockopt()* function shall fail if:

- [EBADF] The *socket* argument is not a valid file descriptor.
- [EINVAL] The specified option is invalid at the specified socket level.
- [ENOPROTOOPT] The option is not supported by the protocol.
- [ENOTSOCK] The *socket* argument does not refer to a socket.

The *getsockopt()* function may fail if:

- [EACCES] The calling process does not have appropriate privileges.
- [EINVAL] The socket has been shut down.
- [ENOBUFFS] Insufficient resources are available in the system to complete the function.

36486 EXAMPLES

36487 None.

36488 APPLICATION USAGE

36489 None.

36490 RATIONALE

36491 None.

36492 FUTURE DIRECTIONS

36493 None.

36494 SEE ALSO

36495 [Section 2.10.16](#) (on page 527), [bind\(\)](#), [close\(\)](#), [endprotoent\(\)](#), [setsockopt\(\)](#), [socket\(\)](#)

36496 XBD [<sys/socket.h>](#), [<netinet/in.h>](#)

36497 CHANGE HISTORY

36498 First released in Issue 6. Derived from the XNS, Issue 5.2 specification.

36499 The **restrict** keyword is added to the *getsockopt()* prototype for alignment with the
36500 ISO/IEC 9899:1999 standard.

36501 IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/47 is applied, updating the description of
36502 SO_LINGER in the DESCRIPTION so that it blocks the calling thread rather than the process.

36503 Issue 7

36504 Austin Group Interpretation 1003.1-2001 #158 is applied, removing text relating to socket options
36505 that is now in [Section 2.10.16](#) (on page 527).

NAME

getsubopt — parse suboption arguments from a string

SYNOPSIS

```
#include <stdlib.h>
```

```
int getsubopt(char **optionp, char * const *keylistp, char **valuep);
```

DESCRIPTION

The *getsubopt()* function shall parse suboption arguments in a flag argument. Such options often result from the use of *getopt()*.

The *getsubopt()* argument *optionp* is a pointer to a pointer to the option argument string. The suboption arguments shall be separated by <comma> characters and each may consist of either a single token, or a token-value pair separated by an <equals-sign>.

The *keylistp* argument shall be a pointer to a vector of strings. The end of the vector is identified by a null pointer. Each entry in the vector is one of the possible tokens that might be found in **optionp*. Since <comma> characters delimit suboption arguments in *optionp*, they should not appear in any of the strings pointed to by *keylistp*. Similarly, because an <equals-sign> separates a token from its value, the application should not include an <equals-sign> in any of the strings pointed to by *keylistp*. The *getsubopt()* function shall not modify the *keylistp* vector. +

The *valuep* argument is the address of a value string pointer.

If a <comma> appears in *optionp*, it shall be interpreted as a suboption separator. After <comma> characters have been processed, if there are one or more <equals-sign> characters in a suboption string, the first <equals-sign> in any suboption string shall be interpreted as a separator between a token and a value. Subsequent <equals-sign> characters in a suboption string shall be interpreted as part of the value.

If the string at **optionp* contains only one suboption argument (equivalently, no <comma> characters), *getsubopt()* shall update **optionp* to point to the null character at the end of the string. Otherwise, it shall isolate the suboption argument by replacing the <comma> separator with a null character, and shall update **optionp* to point to the start of the next suboption argument. If the suboption argument has an associated value (equivalently, contains an <equals-sign>), *getsubopt()* shall update **valuep* to point to the value's first character. Otherwise, it shall set **valuep* to a null pointer. The calling application may use this information to determine whether the presence or absence of a value for the suboption is an error.

Additionally, when *getsubopt()* fails to match the suboption argument with a token in the *keylistp* array, the calling application should decide if this is an error, or if the unrecognized option should be processed in another way.

RETURN VALUE

The *getsubopt()* function shall return the index of the matched token string, or -1 if no token strings were matched.

ERRORS

No errors are defined.

EXAMPLES

Parsing Suboptions

The following example uses the `getsubopt()` function to parse a *value* argument in the *optarg* external variable returned by a call to `getopt()`.

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

int do_all;
const char *type;
int read_size;
int write_size;
int read_only;

enum
{
    RO_OPTION = 0,
    RW_OPTION,
    READ_SIZE_OPTION,
    WRITE_SIZE_OPTION
};

const char *mount_opts[] =
{
    [RO_OPTION] = "ro",
    [RW_OPTION] = "rw",
    [READ_SIZE_OPTION] = "rsize",
    [WRITE_SIZE_OPTION] = "wsize",
    NULL
};

int
main(int argc, char *argv[])
{
    char *subopts, *value;
    int opt;

    while ((opt = getopt(argc, argv, "at:o:")) != -1)
        switch(opt)
        {
            case 'a':
                do_all = 1;
                break;
            case 't':
                type = optarg;
                break;
            case 'o':
                subopts = optarg;
                while (*subopts != ' ')
                {
                    char *saved = subopts;
                    switch(getsubopt(&subopts, (char **)mount_opts,
```

```

36592         &value))
36593     {
36594     case RO_OPTION:
36595         read_only = 1;
36596         break;
36597     case RW_OPTION:
36598         read_only = 0;
36599         break;
36600     case READ_SIZE_OPTION:
36601         if (value == NULL)
36602             abort();
36603         read_size = atoi(value);
36604         break;
36605     case WRITE_SIZE_OPTION:
36606         if (value == NULL)
36607             abort();
36608         write_size = atoi(value);
36609         break;
36610     default:
36611         /* Unknown suboption. */
36612         printf("Unknown suboption '%s'\n", saved);
36613         abort();
36614     }
36615 }
36616 break;
36617 default:
36618     abort();
36619 }
36620
36621 /* Do the real work. */
36622 return 0;
36623 }

```

If the above example is invoked with:

```
program -o ro,rsiz=512
```

then after option parsing, the variable *do_all* will be 0, *type* will be a null pointer, *read_size* will be 512, *write_size* will be 0, and *read_only* will be 1. If it is invoked with:

```
program -o oops
```

it will print:

```
"Unknown suboption 'oops' "
```

before aborting.

APPLICATION USAGE

The value of **valuep* when *getsubopt()* returns *-1* is unspecified. Historical implementations provide various incompatible extensions to allow an application to access the suboption text that was not found in the *keylistp* array.

RATIONALE

The *keylistp* argument of *getsubopt()* is typed as **char * const *** to match historical practice. However, the standard is clear that implementations will not modify either the array or the strings contained in the array, as if the argument had been typed **const char * const ***.

FUTURE DIRECTIONS

None.

SEE ALSO

[*getopt\(\)*](#)

XBD [`<stdlib.h>`](#)

CHANGE HISTORY

First released in Issue 4, Version 2.

Issue 5

Moved from X/OPEN UNIX extension to BASE.

Issue 6

IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/26 is applied, correcting an editorial error in the SYNOPSIS.

Issue 7

The *getsubopt()* function is moved from the XSI option to the Base.

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0260 [196], XSH/TC1-2008/0261 [196], XSH/TC1-2008/0262 [196], XSH/TC1-2008/0263 [196], XSH/TC1-2008/0264 [196], XSH/TC1-2008/0265 [196], XSH/TC1-2008/0266 [196], XSH/TC1-2008/0267 [196], XSH/TC1-2008/0268 [196], and XSH/TC1-2008/0269 [196] are applied.

NAME

gettimeofday — get the date and time

SYNOPSIS

OB XSI `#include <sys/time.h>`

```
int gettimeofday(struct timeval *restrict tp, void *restrict tzp);
```

DESCRIPTION

The *gettimeofday()* function shall obtain the current time, expressed as seconds and microseconds since the Epoch, and store it in the **timeval** structure pointed to by *tp*. The resolution of the system clock is unspecified.

If *tzp* is not a null pointer, the behavior is unspecified.

RETURN VALUE

The *gettimeofday()* function shall return 0 and no value shall be reserved to indicate an error.

ERRORS

No errors are defined.

EXAMPLES

None.

APPLICATION USAGE

Applications should use the *clock_gettime()* function instead of the obsolescent *gettimeofday()* function.

RATIONALE

None.

FUTURE DIRECTIONS

The *gettimeofday()* function may be removed in a future version.

SEE ALSO

clock_getres(), *ctime()*

XBD [<sys/time.h>](#)

CHANGE HISTORY

First released in Issue 4, Version 2.

Issue 5

Moved from X/OPEN UNIX extension to BASE.

Issue 6

The DESCRIPTION is updated to refer to “seconds since the Epoch” rather than “seconds since 00:00:00 UTC (Coordinated Universal Time), January 1 1970” for consistency with other *time* functions.

The **restrict** keyword is added to the *gettimeofday()* prototype for alignment with the ISO/IEC 9899:1999 standard.

Issue 7

The *gettimeofday()* function is marked obsolescent.

36695 **NAME**

36696 getuid — get a real user ID

36697 **SYNOPSIS**

36698 #include <unistd.h>

36699 uid_t getuid(void);

36700 **DESCRIPTION**36701 The *getuid()* function shall return the real user ID of the calling process.36702 **RETURN VALUE**36703 The *getuid()* function shall always be successful and no return value is reserved to indicate the
36704 error.36705 **ERRORS**

36706 No errors are defined.

36707 **EXAMPLES**36708 **Setting the Effective User ID to the Real User ID**36709 The following example sets the effective user ID and the real user ID of the current process to the
36710 real user ID of the caller.36711 #include <unistd.h>
36712 #include <sys/types.h>
36713 ...
36714 setreuid(getuid(), getuid());
36715 ...36716 **APPLICATION USAGE**

36717 None.

36718 **RATIONALE**

36719 None.

36720 **FUTURE DIRECTIONS**

36721 None.

36722 **SEE ALSO**36723 *getegid(), geteuid(), getgid(), setegid(), seteuid(), setgid(), setregid(), setreuid(), setuid()*36724 XBD **<sys/types.h>**, **<unistd.h>**36725 **CHANGE HISTORY**

36726 First released in Issue 1. Derived from Issue 1 of the SVID.

36727 **Issue 6**36728 In the SYNOPSIS, the optional include of the **<sys/types.h>** header is removed.36729 The following new requirements on POSIX implementations derive from alignment with the
36730 Single UNIX Specification:

- 36731
- The requirement to include **<sys/types.h>** has been removed. Although **<sys/types.h>** was
36732 required for conforming implementations of previous POSIX specifications, it was not
36733 required for UNIX applications.

36734 NAME

36735 getutxent, getutxid, getutxline — get user accounting database entries

36736 SYNOPSIS

```
36737 XSI      #include <utmpx.h>
36738          struct utmpx *getutxent(void);
36739          struct utmpx *getutxid(const struct utmpx *id);
36740          struct utmpx *getutxline(const struct utmpx *line);
```

36741 DESCRIPTION

36742 Refer to *endutxent()*.

36743 **NAME**36744 `getwc` — get a wide character from a stream36745 **SYNOPSIS**36746 `#include <stdio.h>`36747 `#include <wchar.h>`36748 `wint_t getwc(FILE *stream);`36749 **DESCRIPTION**

36750 **CX** The functionality described on this reference page is aligned with the ISO C standard. Any
 36751 conflict between the requirements described here and the ISO C standard is unintentional. This
 36752 volume of POSIX.1-2008 defers to the ISO C standard.

36753 The `getwc()` function shall be equivalent to `fgetwc()`, except that if it is implemented as a macro it
 36754 may evaluate *stream* more than once, so the argument should never be an expression with side-
 36755 effects.

36756 **RETURN VALUE**36757 Refer to `fgetwc()`.36758 **ERRORS**36759 Refer to `fgetwc()`.36760 **EXAMPLES**

36761 None.

36762 **APPLICATION USAGE**

36763 Since it may be implemented as a macro, `getwc()` may treat incorrectly a *stream* argument with
 36764 side-effects. In particular, `getwc(*f++)` does not necessarily work as expected. Therefore, use of
 36765 this function is not recommended; `fgetwc()` should be used instead.

36766 **RATIONALE**

36767 None.

36768 **FUTURE DIRECTIONS**

36769 None.

36770 **SEE ALSO**36771 [Section 2.5](#) (on page 495), `fgetwc()`36772 XBD `<stdio.h>`, `<wchar.h>`36773 **CHANGE HISTORY**

36774 First released as a World-wide Portability Interface in Issue 4. Derived from the MSE working
 36775 draft.

36776 **Issue 5**36777 The Optional Header (OH) marking is removed from `<stdio.h>`.36778 **Issue 7**

36779 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0270 [14] is applied.

+

NAME

getwchar — get a wide character from a *stdin* stream

SYNOPSIS

```
#include <wchar.h>
```

```
wint_t getwchar(void);
```

DESCRIPTION

CX The functionality described on this reference page is aligned with the ISO C standard. Any conflict between the requirements described here and the ISO C standard is unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

The *getwchar()* function shall be equivalent to *getwc(stdin)*.

RETURN VALUE

Refer to *fgetwc()*.

ERRORS

Refer to *fgetwc()*.

EXAMPLES

None.

APPLICATION USAGE

If the **wint_t** value returned by *getwchar()* is stored into a variable of type **wchar_t** and then compared against the **wint_t** macro WEOF, the result may be incorrect. Only the **wint_t** type is guaranteed to be able to represent any wide character and WEOF.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

Section 2.5 (on page 495), *fgetwc()*, *getwc()*

XBD <**wchar.h**>

CHANGE HISTORY

First released as a World-wide Portability Interface in Issue 4. Derived from the MSE working draft.

Issue 7

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0271 [14] is applied.

+

NAME

glob, globfree — generate pathnames matching a pattern

SYNOPSIS

```
#include <glob.h>

int glob(const char *restrict pattern, int flags,
         int (*errfunc)(const char *epath, int errno),
         glob_t *restrict pglob);
void globfree(glob_t *pglob);
```

DESCRIPTION

The *glob()* function is a pathname generator that shall implement the rules defined in XCU [Section 2.13](#) (on page 2354), with optional support for rule 3 in XCU [Section 2.13.3](#) (on page 2355).

The structure type **glob_t** is defined in **<glob.h>** and includes at least the following members:

Member Type	Member Name	Description
size_t	<i>gl_pathc</i>	Count of paths matched by <i>pattern</i> .
char **	<i>gl_pathv</i>	Pointer to a list of matched pathnames.
size_t	<i>gl_offs</i>	Slots to reserve at the beginning of <i>gl_pathv</i> .

The argument *pattern* is a pointer to a pathname pattern to be expanded. The *glob()* function shall match all accessible pathnames against this pattern and develop a list of all pathnames that match. In order to have access to a pathname, *glob()* requires search permission on every component of a path except the last, and read permission on each directory of any filename component of *pattern* that contains any of the following special characters: '*', '?', and '['.

The *glob()* function shall store the number of matched pathnames into *pglob->gl_pathc* and a pointer to a list of pointers to pathnames into *pglob->gl_pathv*. The pathnames shall be in sort order as defined by the current setting of the *LC_COLLATE* category; see XBD [Section 7.3.2](#) (on page 147). The first pointer after the last pathname shall be a null pointer. If the pattern does not match any pathnames, the returned number of matched paths is set to 0, and the contents of *pglob->gl_pathv* are implementation-defined.

It is the caller's responsibility to create the structure pointed to by *pglob*. The *glob()* function shall allocate other space as needed, including the memory pointed to by *gl_pathv*. The *globfree()* function shall free any space associated with *pglob* from a previous call to *glob()*.

The *flags* argument is used to control the behavior of *glob()*. The value of *flags* is a bitwise-inclusive OR of zero or more of the following constants, which are defined in **<glob.h>**:

GLOB_APPEND	Append pathnames generated to the ones from a previous call to <i>glob()</i> .
GLOB_DOOFFS	Make use of <i>pglob->gl_offs</i> . If this flag is set, <i>pglob->gl_offs</i> is used to specify how many null pointers to add to the beginning of <i>pglob->gl_pathv</i> . In other words, <i>pglob->gl_pathv</i> shall point to <i>pglob->gl_offs</i> null pointers, followed by <i>pglob->gl_pathc</i> pathname pointers, followed by a null pointer.
GLOB_ERR	Cause <i>glob()</i> to return when it encounters a directory that it cannot open or read. Ordinarily, <i>glob()</i> continues to find matches.
GLOB_MARK	Each pathname that is a directory that matches <i>pattern</i> shall have a <slash> appended.

- 36856 GLOB_NOCHECK Supports rule 3 in XCU [Section 2.13.3](#) (on page 2355). If *pattern* does not
 36857 match any pathname, then *glob()* shall return a list consisting of only
 36858 *pattern*, and the number of matched pathnames is 1.
- 36859 GLOB_NOESCAPE Disable backslash escaping.
- 36860 GLOB_NOSORT Ordinarily, *glob()* sorts the matching pathnames according to the current
 36861 setting of the *LC_COLLATE* category; see XBD [Section 7.3.2](#) (on page 147).
 36862 When this flag is used, the order of pathnames returned is unspecified.

36863 The GLOB_APPEND flag can be used to append a new set of pathnames to those found in a
 36864 previous call to *glob()*. The following rules apply to applications when two or more calls to
 36865 *glob()* are made with the same value of *pglob* and without intervening calls to *globfree()*:

- 36866 1. The first such call shall not set GLOB_APPEND. All subsequent calls shall set it.
- 36867 2. All the calls shall set GLOB_DOOFFS, or all shall not set it.
- 36868 3. After the second call, *pglob->gl_pathv* points to a list containing the following:
- 36869 a. Zero or more null pointers, as specified by GLOB_DOOFFS and *pglob->gl_offs*.
- 36870 b. Pointers to the pathnames that were in the *pglob->gl_pathv* list before the call, in
 36871 the same order as before.
- 36872 c. Pointers to the new pathnames generated by the second call, in the specified order.
- 36873 4. The count returned in *pglob->gl_pathc* shall be the total number of pathnames from the
 36874 two calls.
- 36875 5. The application can change any of the fields after a call to *glob()*. If it does, the
 36876 application shall reset them to the original value before a subsequent call, using the same
 36877 *pglob* value, to *globfree()* or *glob()* with the GLOB_APPEND flag.

36878 If, during the search, a directory is encountered that cannot be opened or read and *errfunc* is not
 36879 a null pointer, *glob()* calls (**errfunc()*) with two arguments:

- 36880 1. The *epath* argument is a pointer to the path that failed.
- 36881 2. The *errno* argument is the value of *errno* from the failure, as set by *opendir()*, *readdir()*, or
 36882 *stat()*. (Other values may be used to report other errors not explicitly documented for
 36883 those functions.)

36884 If (**errfunc()*) is called and returns non-zero, or if the GLOB_ERR flag is set in *flags*, *glob()* shall
 36885 stop the scan and return GLOB_ABORTED after setting *gl_pathc* and *gl_pathv* in *pglob* to reflect
 36886 the paths already scanned. If GLOB_ERR is not set and either *errfunc* is a null pointer or
 36887 (**errfunc()*) returns 0, the error shall be ignored.

36888 The *glob()* function shall not fail because of large files.

36889 RETURN VALUE

36890 Upon successful completion, *glob()* shall return 0. The argument *pglob->gl_pathc* shall return the
 36891 number of matched pathnames and the argument *pglob->gl_pathv* shall contain a pointer to a
 36892 null-terminated list of matched and sorted pathnames. However, if *pglob->gl_pathc* is 0, the
 36893 content of *pglob->gl_pathv* is undefined.

36894 The *globfree()* function shall not return a value.

36895 If *glob()* terminates due to an error, it shall return one of the non-zero constants defined in
 36896 **<glob.h>**. The arguments *pglob->gl_pathc* and *pglob->gl_pathv* are still set as defined above.

ERRORS

The *glob()* function shall fail and return the corresponding value if:

36899	GLOB_ABORTED	The scan was stopped because GLOB_ERR was set or (*errfunc()) returned non-zero.
36901	GLOB_NOMATCH	The pattern does not match any existing pathname, and GLOB_NOCHECK was not set in flags.
36903	GLOB_NOSPACE	An attempt to allocate memory failed.

EXAMPLES

One use of the GLOB_DOOFFS flag is by applications that build an argument list for use with *execv()*, *execve()*, or *execvp()*. Suppose, for example, that an application wants to do the equivalent of:

```
ls -l *.c
```

but for some reason:

```
system("ls -l *.c")
```

is not acceptable. The application could obtain approximately the same result using the sequence:

```
globbuf.gl_offs = 2;
glob("*.c", GLOB_DOOFFS, NULL, &globbuf);
globbuf.gl_pathv[0] = "ls";
globbuf.gl_pathv[1] = "-l";
execvp("ls", &globbuf.gl_pathv[0]);
```

Using the same example:

```
ls -l *.c *.h
```

could be approximately simulated using GLOB_APPEND as follows:

```
globbuf.gl_offs = 2;
glob("*.c", GLOB_DOOFFS, NULL, &globbuf);
glob("*.h", GLOB_DOOFFS|GLOB_APPEND, NULL, &globbuf);
...
```

APPLICATION USAGE

This function is not provided for the purpose of enabling utilities to perform pathname expansion on their arguments, as this operation is performed by the shell, and utilities are explicitly not expected to redo this. Instead, it is provided for applications that need to do pathname expansion on strings obtained from other sources, such as a pattern typed by a user or read from a file.

If a utility needs to see if a pathname matches a given pattern, it can use *fnmatch()*.

Note that *gl_pathc* and *gl_pathv* have meaning even if *glob()* fails. This allows *glob()* to report partial results in the event of an error. However, if *gl_pathc* is 0, *gl_pathv* is unspecified even if *glob()* did not return an error.

The GLOB_NOCHECK option could be used when an application wants to expand a pathname if wildcards are specified, but wants to treat the pattern as just a string otherwise. The *sh* utility might use this for option-arguments, for example.

The new pathnames generated by a subsequent call with GLOB_APPEND are not sorted together with the previous pathnames. This mirrors the way that the shell handles pathname

expansion when multiple expansions are done on a command line.

Applications that need tilde and parameter expansion should use *wordexp()*.

RATIONALE

It was claimed that the GLOB_DOOFFS flag is unnecessary because it could be simulated using:

```
new = (char **)malloc((n + pglob->gl_pathc + 1)
    * sizeof(char *));
(void) memcpy(new+n, pglob->gl_pathv,
    pglob->gl_pathc * sizeof(char *));
(void) memset(new, 0, n * sizeof(char *));
free(pglob->gl_pathv);
pglob->gl_pathv = new;
```

However, this assumes that the memory pointed to by *gl_pathv* is a block that was separately created using *malloc()*. This is not necessarily the case. An application should make no assumptions about how the memory referenced by fields in *pglob* was allocated. It might have been obtained from *malloc()* in a large chunk and then carved up within *glob()*, or it might have been created using a different memory allocator. It is not the intent of the standard developers to specify or imply how the memory used by *glob()* is managed.

The GLOB_APPEND flag would be used when an application wants to expand several different patterns into a single list.

FUTURE DIRECTIONS

None.

SEE ALSO

exec, *fdopendir()*, *fnmatch()*, *fstatat()*, *readdir()*, [Section 2.6](#)
XBD [Section 7.3.2](#) (on page 147), [<glob.h>](#)

CHANGE HISTORY

First released in Issue 4. Derived from the ISO POSIX-2 standard.

Issue 5

Moved from POSIX2 C-language Binding to BASE.

Issue 6

The normative text is updated to avoid use of the term “must” for application requirements.

The **restrict** keyword is added to the *glob()* prototype for alignment with the ISO/IEC 9899:1999 standard.

36972 **NAME**

36973 gmtime, gmtime_r — convert a time value to a broken-down UTC time

36974 **SYNOPSIS**

```
36975 #include <time.h>
36976 struct tm *gmtime(const time_t *timer);
36977 CX struct tm *gmtime_r(const time_t *restrict timer,
36978 struct tm *restrict result);
```

36979 **DESCRIPTION**

36980 CX For *gmtime()*: The functionality described on this reference page is aligned with the ISO C
 36981 standard. Any conflict between the requirements described here and the ISO C standard is
 36982 unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

36983 The *gmtime()* function shall convert the time in seconds since the Epoch pointed to by *timer* into
 36984 a broken-down time, expressed as Coordinated Universal Time (UTC).

36985 CX The relationship between a time in seconds since the Epoch used as an argument to *gmtime()*
 36986 and the **tm** structure (defined in the **<time.h>** header) is that the result shall be as specified in
 36987 the expression given in the definition of seconds since the Epoch (see XBD [Section 4.15](#), on page
 36988 113), where the names in the structure and in the expression correspond.

36989 The same relationship shall apply for *gmtime_r()*.

36990 The *gmtime()* function need not be thread-safe.

36991 The *asctime()*, *ctime()*, *gmtime()*, and *localtime()* functions shall return values in one of two static
 36992 objects: a broken-down time structure and an array of type **char**. Execution of any of the
 36993 functions may overwrite the information returned in either of these objects by any of the other
 36994 functions.

36995 The *gmtime_r()* function shall convert the time in seconds since the Epoch pointed to by *timer*
 36996 into a broken-down time expressed as Coordinated Universal Time (UTC). The broken-down
 36997 time is stored in the structure referred to by *result*. The *gmtime_r()* function shall also return the
 36998 address of the same structure.

36999 **RETURN VALUE**

37000 Upon successful completion, the *gmtime()* function shall return a pointer to a **struct tm**. If an
 37001 CX error is detected, *gmtime()* shall return a null pointer and set *errno* to indicate the error.

37002 Upon successful completion, *gmtime_r()* shall return the address of the structure pointed to by
 37003 the argument *result*. If an error is detected, *gmtime_r()* shall return a null pointer and set *errno* to
 37004 indicate the error.

37005 **ERRORS**

37006 CX The *gmtime()* and *gmtime_r()* functions shall fail if:

37007 CX **[EOVERFLOW]** The result cannot be represented.

EXAMPLES

None.

APPLICATION USAGE

The *gmtime_r()* function is thread-safe and returns values in a user-supplied buffer instead of possibly using a static data area that may be overwritten by each call.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

asctime(), *clock()*, *ctime()*, *difftime()*, *localtime()*, *mktime()*, *strftime()*, *strptime()*, *time()*, *utime()*

XBD Section 4.15 (on page 113), [<time.h>](#)

CHANGE HISTORY

First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 5

A note indicating that the *gmtime()* function need not be reentrant is added to the DESCRIPTION.

The *gmtime_r()* function is included for alignment with the POSIX Threads Extension.

Issue 6

The *gmtime_r()* function is marked as part of the Thread-Safe Functions option.

Extensions beyond the ISO C standard are marked.

The APPLICATION USAGE section is updated to include a note on the thread-safe function and its avoidance of possibly using a static data area.

The **restrict** keyword is added to the *gmtime_r()* prototype for alignment with the ISO/IEC 9899:1999 standard.

IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/27 is applied, adding the [EOVERFLOW] error.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/48 is applied, updating the error handling for *gmtime_r()*.

Issue 7

Austin Group Interpretation 1003.1-2001 #156 is applied.

The *gmtime_r()* function is moved from the Thread-Safe Functions option to the Base.

NAME

grantpt — grant access to the slave pseudo-terminal device

SYNOPSIS

```
#include <stdlib.h>

int grantpt(int fildes);
```

DESCRIPTION

The *grantpt()* function shall change the mode and ownership of the slave pseudo-terminal device associated with its master pseudo-terminal counterpart. The *fildes* argument is a file descriptor that refers to a master pseudo-terminal device. The user ID of the slave shall be set to the real UID of the calling process and the group ID shall be set to an unspecified group ID. The permission mode of the slave pseudo-terminal shall be set to readable and writable by the owner, and writable by the group.

The behavior of the *grantpt()* function is unspecified if the application has installed a signal handler to catch SIGCHLD signals.

RETURN VALUE

Upon successful completion, *grantpt()* shall return 0; otherwise, it shall return -1 and set *errno* to indicate the error.

ERRORS

The *grantpt()* function may fail if:

[EBADF]	The <i>fildes</i> argument is not a valid open file descriptor.
[EINVAL]	The <i>fildes</i> argument is not associated with a master pseudo-terminal device.
[EACCES]	The corresponding slave pseudo-terminal device could not be accessed.

EXAMPLES

None.

APPLICATION USAGE

None.

RATIONALE

See the RATIONALE section for *posix_openpt()*.

FUTURE DIRECTIONS

None.

SEE ALSO

open(), *posix_openpt()*, *ptsname()*, *unlockpt()*

XBD *<stdlib.h>*

CHANGE HISTORY

First released in Issue 4, Version 2.

Issue 5

Moved from X/OPEN UNIX extension to BASE.

The last paragraph of the DESCRIPTION is moved from the APPLICATION USAGE section.

37078 **Issue 7**
37079

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0272 [96] is applied.

+

37080 **NAME**37081 `hcreate, hdestroy, hsearch` — manage hash search table37082 **SYNOPSIS**

```

37083 XSI      #include <search.h>
37084
37084      int hcreate(size_t nel);
37085      void hdestroy(void);
37086      ENTRY *hsearch(ENTRY item, ACTION action);

```

37087 **DESCRIPTION**37088 The `hcreate()`, `hdestroy()`, and `hsearch()` functions shall manage hash search tables.

37089 The `hcreate()` function shall allocate sufficient space for the table, and the application shall ensure it is called before `hsearch()` is used. The `nel` argument is an estimate of the maximum number of entries that the table shall contain. This number may be adjusted upward by the algorithm in order to obtain certain mathematically favorable circumstances.

37093 The `hdestroy()` function shall dispose of the search table, and may be followed by another call to `hcreate()`. After the call to `hdestroy()`, the data can no longer be considered accessible.

37095 The `hsearch()` function is a hash-table search routine. It shall return a pointer into a hash table indicating the location at which an entry can be found. The `item` argument is a structure of type **ENTRY** (defined in the `<search.h>` header) containing two pointers: `item.key` points to the comparison key (a `char *`), and `item.data` (a `void *`) points to any other data to be associated with that key. The comparison function used by `hsearch()` is `strcmp()`. The `action` argument is a member of an enumeration type **ACTION** indicating the disposition of the entry if it cannot be found in the table. **ENTER** indicates that the item should be inserted in the table at an appropriate point. **FIND** indicates that no entry should be made. Unsuccessful resolution is indicated by the return of a null pointer.

37104 These functions need not be thread-safe.

37105 **RETURN VALUE**37106 The `hcreate()` function shall return 0 if it cannot allocate sufficient space for the table; otherwise, it shall return non-zero.37108 The `hdestroy()` function shall not return a value.

37109 The `hsearch()` function shall return a null pointer if either the action is **FIND** and the item could not be found or the action is **ENTER** and the table is full.

37111 **ERRORS**37112 The `hcreate()` and `hsearch()` functions may fail if:

37113 [ENOMEM] Insufficient storage space is available.

37114 **EXAMPLES**

37115 The following example reads in strings followed by two numbers and stores them in a hash table, discarding duplicates. It then reads in strings and finds the matching entry in the hash table and prints it out.

```

37118 #include <stdio.h>
37119 #include <search.h>
37120 #include <string.h>
37121
37121 struct info {          /* This is the info stored in the table */
37122     int age, room;     /* other than the key. */
37123 };

```

```

37124     #define NUM_EMPL      5000      /* # of elements in search table. */
37125     int main(void)
37126     {
37127         char string_space[NUM_EMPL*20]; /* Space to store strings. */
37128         struct info info_space[NUM_EMPL]; /* Space to store employee info. */
37129         char *str_ptr = string_space; /* Next space in string_space. */
37130         struct info *info_ptr = info_space;
37131                                     /* Next space in info_space. */
37132         ENTRY item;
37133         ENTRY *found_item; /* Name to look for in table. */
37134         char name_to_find[30];
37135
37136         int i = 0;
37137
37138         /* Create table; no error checking is performed. */
37139         (void) hcreate(NUM_EMPL);
37140         while (scanf("%s%d%d", str_ptr, &info_ptr->age,
37141                     &info_ptr->room) != EOF && i++ < NUM_EMPL) {
37142             /* Put information in structure, and structure in item. */
37143             item.key = str_ptr;
37144             item.data = info_ptr;
37145             str_ptr += strlen(str_ptr) + 1;
37146             info_ptr++;
37147
37148             /* Put item into table. */
37149             (void) hsearch(item, ENTER);
37150         }
37151
37152         /* Access table. */
37153         item.key = name_to_find;
37154         while (scanf("%s", item.key) != EOF) {
37155             if ((found_item = hsearch(item, FIND)) != NULL) {
37156                 /* If item is in the table. */
37157                 (void)printf("found %s, age = %d, room = %d\n",
37158                             found_item->key,
37159                             ((struct info *)found_item->data)->age,
37160                             ((struct info *)found_item->data)->room);
37161             } else
37162                 (void)printf("no such employee %s\n", name_to_find);
37163         }
37164         return 0;
37165     }

```

APPLICATION USAGE

The *hcreate()* and *hsearch()* functions may use *malloc()* to allocate space.

RATIONALE

None.

FUTURE DIRECTIONS

None.

37168 **SEE ALSO**37169 *bsearch()*, *lsearch()*, *malloc()*, *strcmp()*, *tdelete()*37170 XBD <*search.h*>37171 **CHANGE HISTORY**

37172 First released in Issue 1. Derived from Issue 1 of the SVID.

37173 **Issue 6**

37174 The normative text is updated to avoid use of the term “must” for application requirements.

37175 A note indicating that this function need not be reentrant is added to the DESCRIPTION.

37176 **Issue 7**

37177 Austin Group Interpretation 1003.1-2001 #156 is applied.

NAME

htonl, htons, ntohl, ntohs — convert values between host and network byte order

SYNOPSIS

```
#include <arpa/inet.h>

uint32_t htonl(uint32_t hostlong);
uint16_t htons(uint16_t hostshort);
uint32_t ntohl(uint32_t netlong);
uint16_t ntohs(uint16_t netshort);
```

DESCRIPTION

These functions shall convert 16-bit and 32-bit quantities between network byte order and host byte order.

On some implementations, these functions are defined as macros.

The `uint32_t` and `uint16_t` types are defined in `<inttypes.h>`.

RETURN VALUE

The `htonl()` and `htons()` functions shall return the argument value converted from host to network byte order.

The `ntohl()` and `ntohs()` functions shall return the argument value converted from network to host byte order.

ERRORS

No errors are defined.

EXAMPLES

None.

APPLICATION USAGE

These functions are most often used in conjunction with IPv4 addresses and ports as returned by `gethostent()` and `getservent()`.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

[*endhostent\(\)*](#), [*endservent\(\)*](#)

XBD [*<arpa/inet.h>*](#), [*<inttypes.h>*](#)

CHANGE HISTORY

First released in Issue 6. Derived from the XNS, Issue 5.2 specification.

37212 **NAME**

37213 hypot, hypotf, hypotl — Euclidean distance function

37214 **SYNOPSIS**

```
37215 #include <math.h>
37216 double hypot(double x, double y);
37217 float hypotf(float x, float y);
37218 long double hypotl(long double x, long double y);
```

37219 **DESCRIPTION**

37220 CX The functionality described on this reference page is aligned with the ISO C standard. Any
 37221 conflict between the requirements described here and the ISO C standard is unintentional. This
 37222 volume of POSIX.1-2008 defers to the ISO C standard.

37223 These functions shall compute the value of the square root of x^2+y^2 without undue overflow or
 37224 underflow.

37225 An application wishing to check for error situations should set *errno* to zero and call
 37226 *feclearexcept*(FE_ALL_EXCEPT) before calling these functions. On return, if *errno* is non-zero or
 37227 *fetestexcept*(FE_INVALID | FE_DIVBYZERO | FE_OVERFLOW | FE_UNDERFLOW) is non-
 37228 zero, an error has occurred.

37229 **RETURN VALUE**

37230 Upon successful completion, these functions shall return the length of the hypotenuse of a right-
 37231 angled triangle with sides of length *x* and *y*.

37232 If the correct value would cause overflow, a range error shall occur and *hypot()*, *hypotf()*, and
 37233 *hypotl()* shall return the value of the macro HUGE_VAL, HUGE_VALF, and HUGE_VALL,
 37234 respectively.

37235 MX If *x* or *y* is $\pm\text{Inf}$, $\pm\text{Inf}$ shall be returned (even if one of *x* or *y* is NaN).

37236 If *x* or *y* is NaN, and the other is not $\pm\text{Inf}$, a NaN shall be returned.

37237 MXX If both arguments are subnormal and the correct result is subnormal, a range error may occur
 37238 and the correct result shall be returned.

37239 **ERRORS**

37240 These functions shall fail if:

37241 Range Error The result overflows.

37242 If the integer expression (*math_errhandling* & MATH_ERRNO) is non-zero,
 37243 then *errno* shall be set to [ERANGE]. If the integer expression
 37244 (*math_errhandling* & MATH_ERREXCEPT) is non-zero, then the overflow
 37245 floating-point exception shall be raised.

37246 These functions may fail if:

37247 MX Range Error The result underflows.

37248 If the integer expression (*math_errhandling* & MATH_ERRNO) is non-zero,
 37249 then *errno* shall be set to [ERANGE]. If the integer expression
 37250 (*math_errhandling* & MATH_ERREXCEPT) is non-zero, then the underflow
 37251 floating-point exception shall be raised.

EXAMPLES

See the EXAMPLES section in `atan2()`.

APPLICATION USAGE

$\text{hypot}(x,y)$, $\text{hypot}(y,x)$, and $\text{hypot}(x,-y)$ are equivalent.

$\text{hypot}(x, \pm 0)$ is equivalent to $\text{fabs}(x)$.

Underflow only happens when both x and y are subnormal and the (inexact) result is also subnormal.

These functions take precautions against overflow during intermediate steps of the computation.

On error, the expressions `(math_errhandling & MATH_ERRNO)` and `(math_errhandling & MATH_ERREXCEPT)` are independent of each other, but at least one of them must be non-zero.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

[`atan2\(\)`](#), [`feclearexcept\(\)`](#), [`fetestexcept\(\)`](#), [`isnan\(\)`](#), [`sqrt\(\)`](#)

XBD [Section 4.19](#) (on page 116), [`<math.h>`](#)

CHANGE HISTORY

First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 5

The DESCRIPTION is updated to indicate how an application should check for an error. This text was previously published in the APPLICATION USAGE section.

Issue 6

The `hypot()` function is no longer marked as an extension.

The `hypotf()` and `hypotl()` functions are added for alignment with the ISO/IEC 9899:1999 standard.

The DESCRIPTION, RETURN VALUE, ERRORS, and APPLICATION USAGE sections are revised to align with the ISO/IEC 9899:1999 standard.

IEC 60559:1989 standard floating-point extensions over the ISO/IEC 9899:1999 standard are marked.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/49 is applied, updating the EXAMPLES section.

Issue 7

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0273 [68] is applied.

+

NAME

iconv — codeset conversion function

SYNOPSIS

```
#include <iconv.h>

size_t iconv(iconv_t cd, char **restrict inbuf,
             size_t *restrict inbytesleft, char **restrict outbuf,
             size_t *restrict outbytesleft);
```

DESCRIPTION

The *iconv()* function shall convert the sequence of characters from one codeset, in the array specified by *inbuf*, into a sequence of corresponding characters in another codeset, in the array specified by *outbuf*. The codesets are those specified in the *iconv_open()* call that returned the conversion descriptor, *cd*. The *inbuf* argument points to a variable that points to the first character in the input buffer and *inbytesleft* indicates the number of bytes to the end of the buffer to be converted. The *outbuf* argument points to a variable that points to the first available byte in the output buffer and *outbytesleft* indicates the number of the available bytes to the end of the buffer.

For state-dependent encodings, the conversion descriptor *cd* is placed into its initial shift state by a call for which *inbuf* is a null pointer, or for which *inbuf* points to a null pointer. When *iconv()* is called in this way, and if *outbuf* is not a null pointer or a pointer to a null pointer, and *outbytesleft* points to a positive value, *iconv()* shall place, into the output buffer, the byte sequence to change the output buffer to its initial shift state. If the output buffer is not large enough to hold the entire reset sequence, *iconv()* shall fail and set *errno* to [E2BIG]. Subsequent calls with *inbuf* as other than a null pointer or a pointer to a null pointer cause the conversion to take place from the current state of the conversion descriptor.

If a sequence of input bytes does not form a valid character in the specified codeset, conversion shall stop after the previous successfully converted character. If the input buffer ends with an incomplete character or shift sequence, conversion shall stop after the previous successfully converted bytes. If the output buffer is not large enough to hold the entire converted input, conversion shall stop just prior to the input bytes that would cause the output buffer to overflow. The variable pointed to by *inbuf* shall be updated to point to the byte following the last byte successfully used in the conversion. The value pointed to by *inbytesleft* shall be decremented to reflect the number of bytes still not converted in the input buffer. The variable pointed to by *outbuf* shall be updated to point to the byte following the last byte of converted output data. The value pointed to by *outbytesleft* shall be decremented to reflect the number of bytes still available in the output buffer. For state-dependent encodings, the conversion descriptor shall be updated to reflect the shift state in effect at the end of the last successfully converted byte sequence.

If *iconv()* encounters a character in the input buffer that is valid, but for which an identical character does not exist in the target codeset, *iconv()* shall perform an implementation-defined conversion on this character.

RETURN VALUE

The *iconv()* function shall update the variables pointed to by the arguments to reflect the extent of the conversion and return the number of non-identical conversions performed. If the entire string in the input buffer is converted, the value pointed to by *inbytesleft* shall be 0. If the input conversion is stopped due to any conditions mentioned above, the value pointed to by *inbytesleft* shall be non-zero and *errno* shall be set to indicate the condition. If an error occurs, *iconv()* shall return (**size_t**)−1 and set *errno* to indicate the error.

ERRORS

The *iconv()* function shall fail if:

[EILSEQ] Input conversion stopped due to an input byte that does not belong to the input codeset.

[E2BIG] Input conversion stopped due to lack of space in the output buffer.

[EINVAL] Input conversion stopped due to an incomplete character or shift sequence at the end of the input buffer.

The *iconv()* function may fail if:

[EBADF] The *cd* argument is not a valid open conversion descriptor.

EXAMPLES

None.

APPLICATION USAGE

The *inbuf* argument indirectly points to the memory area which contains the conversion input data. The *outbuf* argument indirectly points to the memory area which is to contain the result of the conversion. The objects indirectly pointed to by *inbuf* and *outbuf* are not restricted to containing data that is directly representable in the ISO C standard language **char** data type. The type of *inbuf* and *outbuf*, **char ****, does not imply that the objects pointed to are interpreted as null-terminated C strings or arrays of characters. Any interpretation of a byte sequence that represents a character in a given character set encoding scheme is done internally within the codeset converters. For example, the area pointed to indirectly by *inbuf* and/or *outbuf* can contain all zero octets that are not interpreted as string terminators but as coded character data according to the respective codeset encoding scheme. The type of the data (**char**, **short**, **long**, and so on) read or stored in the objects is not specified, but may be inferred for both the input and output data by the converters determined by the *fromcode* and *tocode* arguments of *iconv_open()*.

Regardless of the data type inferred by the converter, the size of the remaining space in both input and output objects (the *intbytesleft* and *outbytesleft* arguments) is always measured in bytes.

For implementations that support the conversion of state-dependent encodings, the conversion descriptor must be able to accurately reflect the shift-state in effect at the end of the last successful conversion. It is not required that the conversion descriptor itself be updated, which would require it to be a pointer type. Thus, implementations are free to implement the descriptor as a handle (other than a pointer type) by which the conversion information can be accessed and updated.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

[*iconv_open\(\)*](#), [*iconv_close\(\)*](#), [*mbsrtowcs\(\)*](#)

XBD [**<iconv.h>**](#)

CHANGE HISTORY

First released in Issue 4. Derived from the HP-UX Manual.

37374 **Issue 6**

37375 The SYNOPSIS has been corrected to align with the **<iconv.h>** reference page.

37376 The **restrict** keyword is added to the *iconv()* prototype for alignment with the
37377 ISO/IEC 9899:1999 standard.

37378 **Issue 7**

37379 The *iconv()* function is moved from the XSI option to the Base.

37380 NAME

37381 iconv_close — codeset conversion deallocation function

37382 SYNOPSIS

37383 #include <iconv.h>

37384 int iconv_close(iconv_t *cd*);

37385 DESCRIPTION

37386 The *iconv_close()* function shall deallocate the conversion descriptor *cd* and all other associated
37387 resources allocated by *iconv_open()*.

37388 If a file descriptor is used to implement the type **iconv_t**, that file descriptor shall be closed.

37389 RETURN VALUE

37390 Upon successful completion, 0 shall be returned; otherwise, -1 shall be returned and *errno* set to
37391 indicate the error.

37392 ERRORS

37393 The *iconv_close()* function may fail if:

37394 [EBADF] The conversion descriptor is invalid.

37395 EXAMPLES

37396 None.

37397 APPLICATION USAGE

37398 None.

37399 RATIONALE

37400 None.

37401 FUTURE DIRECTIONS

37402 None.

37403 SEE ALSO

37404 *iconv()*, *iconv_open()*

37405 XBD <**iconv.h**>

37406 CHANGE HISTORY

37407 First released in Issue 4. Derived from the HP-UX Manual.

37408 Issue 7

37409 The *iconv_close()* function is moved from the XSI option to the Base.

37410 **NAME**

37411 iconv_open — codeset conversion allocation function

37412 **SYNOPSIS**

37413 #include <iconv.h>

37414 iconv_t iconv_open(const char **tocode*, const char **fromcode*);37415 **DESCRIPTION**

37416 The *iconv_open()* function shall return a conversion descriptor that describes a conversion from
 37417 the codeset specified by the string pointed to by the *fromcode* argument to the codeset specified
 37418 by the string pointed to by the *tocode* argument. For state-dependent encodings, the conversion
 37419 descriptor shall be in a codeset-dependent initial shift state, ready for immediate use with
 37420 *iconv()*.

37421 Settings of *fromcode* and *tocode* and their permitted combinations are implementation-defined.

37422 A conversion descriptor shall remain valid until it is closed by *iconv_close()* or an implicit close.

37423 If a file descriptor is used to implement conversion descriptors, the FD_CLOEXEC flag shall be
 37424 set; see <fcntl.h>.

37425 **RETURN VALUE**

37426 Upon successful completion, *iconv_open()* shall return a conversion descriptor for use on
 37427 subsequent calls to *iconv()*. Otherwise, *iconv_open()* shall return (**iconv_t**)−1 and set *errno* to
 37428 indicate the error.

37429 **ERRORS**

37430 The *iconv_open()* function may fail if:

37431 [EMFILE] All file descriptors available to the process are currently open.

37432 [ENFILE] Too many files are currently open in the system.

37433 [ENOMEM] Insufficient storage space is available.

37434 [EINVAL] The conversion specified by *fromcode* and *tocode* is not supported by the
 37435 implementation.

37436 **EXAMPLES**

37437 None.

37438 **APPLICATION USAGE**

37439 Some implementations of *iconv_open()* use *malloc()* to allocate space for internal buffer areas.
 37440 The *iconv_open()* function may fail if there is insufficient storage space to accommodate these
 37441 buffers.

37442 Conforming applications must assume that conversion descriptors are not valid after a call to
 37443 one of the *exec* functions.

37444 Application developers should consult the system documentation to determine the supported
 37445 codesets and their naming schemes.

37446 **RATIONALE**

37447 None.

37448 **FUTURE DIRECTIONS**

37449 None.

37450 **SEE ALSO**37451 *iconv()*, *iconv_close()*

37452 XBD <fcntl.h>, <iconv.h>

37453 **CHANGE HISTORY**

37454 First released in Issue 4. Derived from the HP-UX Manual.

37455 **Issue 7**

37456 SD5-XBD-ERN-4 is applied, changing the definition of the [EMFILE] error.

37457 The *iconv_open()* function is moved from the XSI option to the Base.

37458 **NAME**

37459 if_freenameindex — free memory allocated by if_nameindex

37460 **SYNOPSIS**

37461 #include <net/if.h>

37462 void if_freenameindex(struct if_nameindex *ptr);

37463 **DESCRIPTION**

37464 The *if_freenameindex()* function shall free the memory allocated by *if_nameindex()*. The *ptr*
37465 argument shall be a pointer that was returned by *if_nameindex()*. After *if_freenameindex()* has
37466 been called, the application shall not use the array of which *ptr* is the address.

37467 **RETURN VALUE**

37468 None.

37469 **ERRORS**

37470 No errors are defined.

37471 **EXAMPLES**

37472 None.

37473 **APPLICATION USAGE**

37474 None.

37475 **RATIONALE**

37476 None.

37477 **FUTURE DIRECTIONS**

37478 None.

37479 **SEE ALSO**37480 *getsockopt()*, *if_indextoname()*, *if_nameindex()*, *if_nametoindex()*, *setsockopt()*

37481 XBD <net/if.h>

37482 **CHANGE HISTORY**

37483 First released in Issue 6. Derived from the XNS, Issue 5.2 specification.

37484 **NAME**

37485 if_indextoname — map a network interface index to its corresponding name

37486 **SYNOPSIS**

37487 #include <net/if.h>

37488 char *if_indextoname(unsigned *ifindex*, char **ifname*);37489 **DESCRIPTION**37490 The *if_indextoname()* function shall map an interface index to its corresponding name.37491 When this function is called, *ifname* shall point to a buffer of at least {IF_NAMESIZE} bytes. The
37492 function shall place in this buffer the name of the interface with index *ifindex*.37493 **RETURN VALUE**37494 If *ifindex* is an interface index, then the function shall return the value supplied in *ifname*, which
37495 points to a buffer now containing the interface name. Otherwise, the function shall return a null
37496 pointer and set *errno* to indicate the error.37497 **ERRORS**37498 The *if_indextoname()* function shall fail if:

37499 [ENXIO] The interface does not exist.

37500 **EXAMPLES**

37501 None.

37502 **APPLICATION USAGE**

37503 None.

37504 **RATIONALE**

37505 None.

37506 **FUTURE DIRECTIONS**

37507 None.

37508 **SEE ALSO**37509 *getsockopt()*, *if_freenameindex()*, *if_nameindex()*, *if_nametoindex()*, *setsockopt()*

37510 XBD <net/if.h>

37511 **CHANGE HISTORY**

37512 First released in Issue 6. Derived from the XNS, Issue 5.2 specification.

37513 IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/28 is applied, changing {IFNAMSIZ} to
37514 {IF_NAMESIZ} in the DESCRIPTION.

37515 **NAME**

37516 if_nameindex — return all network interface names and indexes

37517 **SYNOPSIS**

37518 #include <net/if.h>

37519 struct if_nameindex *if_nameindex(void);

37520 **DESCRIPTION**

37521 The *if_nameindex()* function shall return an array of *if_nameindex* structures, one structure per
 37522 interface. The end of the array is indicated by a structure with an *if_index* field of zero and an
 37523 *if_name* field of NULL.

37524 Applications should call *if_freenameindex()* to release the memory that may be dynamically
 37525 allocated by this function, after they have finished using it.

37526 **RETURN VALUE**

37527 An array of structures identifying local interfaces. A null pointer is returned upon an error, with
 37528 *errno* set to indicate the error.

37529 **ERRORS**37530 The *if_nameindex()* function may fail if:

37531 [ENOBUFS] Insufficient resources are available to complete the function.

37532 **EXAMPLES**

37533 None.

37534 **APPLICATION USAGE**

37535 None.

37536 **RATIONALE**

37537 None.

37538 **FUTURE DIRECTIONS**

37539 None.

37540 **SEE ALSO**37541 *getsockopt()*, *if_freenameindex()*, *if_indextoname()*, *if_nametoindex()*, *setsockopt()*

37542 XBD <net/if.h>

37543 **CHANGE HISTORY**

37544 First released in Issue 6. Derived from the XNS, Issue 5.2 specification.

37545 NAME

37546 if_nametoindex — map a network interface name to its corresponding index

37547 SYNOPSIS

37548 #include <net/if.h>

37549 unsigned if_nametoindex(const char *ifname);

37550 DESCRIPTION

37551 The *if_nametoindex()* function shall return the interface index corresponding to name *ifname*.

37552 RETURN VALUE

37553 The corresponding index if *ifname* is the name of an interface; otherwise, zero.

37554 ERRORS

37555 No errors are defined.

37556 EXAMPLES

37557 None.

37558 APPLICATION USAGE

37559 None.

37560 RATIONALE

37561 None.

37562 FUTURE DIRECTIONS

37563 None.

37564 SEE ALSO

37565 [*getsockopt\(\)*](#), [*if_freenameindex\(\)*](#), [*if_indextoname\(\)*](#), [*if_nameindex\(\)*](#), [*setsockopt\(\)*](#)

37566 XBD [**<net/if.h>**](#)

37567 CHANGE HISTORY

37568 First released in Issue 6. Derived from the XNS, Issue 5.2 specification.

37569 **NAME**

37570 ilogb, ilogbf, ilogbl — return an unbiased exponent

37571 **SYNOPSIS**

```
37572     #include <math.h>
37573
37574     int ilogb(double x);
37575     int ilogbf(float x);
37576     int ilogbl(long double x);
```

37576 **DESCRIPTION**

37577 CX The functionality described on this reference page is aligned with the ISO C standard. Any
 37578 conflict between the requirements described here and the ISO C standard is unintentional. This
 37579 volume of POSIX.1-2008 defers to the ISO C standard.

37580 These functions shall return the exponent part of their argument x . Formally, the return value is
 37581 the integral part of $\log_r |x|$ as a signed integral value, for non-zero x , where r is the radix of the
 37582 machine's floating-point arithmetic, which is the value of `FLT_RADIX` defined in `<float.h>`.

37583 An application wishing to check for error situations should set `errno` to zero and call
 37584 *feclearexcept*(`FE_ALL_EXCEPT`) before calling these functions. On return, if `errno` is non-zero or
 37585 *fetestexcept*(`FE_INVALID` | `FE_DIVBYZERO` | `FE_OVERFLOW` | `FE_UNDERFLOW`) is non-
 37586 zero, an error has occurred.

37587 **RETURN VALUE**

37588 Upon successful completion, these functions shall return the exponent part of x as a signed
 37589 integer value. They are equivalent to calling the corresponding `logb()` function and casting the
 37590 returned value to type `int`.

37591 XSI If x is 0, the value `FP_ILOGB0` shall be returned. On XSI-conformant systems, a domain error
 37592 shall occur;

37593 CX otherwise, a domain error may occur.

37594 XSI If x is $\pm\text{Inf}$, the value `{INT_MAX}` shall be returned. On XSI-conformant systems, a domain
 37595 error shall occur;

37596 CX otherwise, a domain error may occur.

37597 XSI If x is a NaN, the value `FP_ILOGBNAN` shall be returned. On XSI-conformant systems, a
 37598 domain error shall occur;

37599 CX otherwise, a domain error may occur.

37600 MX If the correct value is greater than `{INT_MAX}`, a domain error shall occur and an unspecified
 37601 XSI value shall be returned. On XSI-conformant systems, a domain error shall occur and
 37602 `{INT_MAX}` shall be returned.

37603 MX If the correct value is less than `{INT_MIN}`, a domain error shall occur and an unspecified value
 37604 XSI shall be returned. On XSI-conformant systems, a domain error shall occur and `{INT_MIN}` shall
 37605 be returned.

37606 **ERRORS**

37607 These functions shall fail if:

37608 XSI|MX **Domain Error** The correct value is not representable as an integer.

37609 XSI The x argument is zero, NaN, or $\pm\text{Inf}$.

37610 XSI|MX If the integer expression (*math_errhandling* & `MATH_ERRNO`) is non-zero,
 37611 then `errno` shall be set to `[EDOM]`. If the integer expression (*math_errhandling*
 37612 & `MATH_ERREXCEPT`) is non-zero, then the invalid floating-point exception
 37613 shall be raised.

37614 These functions may fail if:

37615 Domain Error The *x* argument is zero, NaN, or $\pm\text{Inf}$.

37616 If the integer expression (*math_errhandling* & MATH_ERRNO) is non-zero,
 37617 then *errno* shall be set to [EDOM]. If the integer expression (*math_errhandling*
 37618 & MATH_ERREXCEPT) is non-zero, then the invalid floating-point exception
 37619 shall be raised.

37620 EXAMPLES

37621 None.

37622 APPLICATION USAGE

37623 On error, the expressions (*math_errhandling* & MATH_ERRNO) and (*math_errhandling* &
 37624 MATH_ERREXCEPT) are independent of each other, but at least one of them must be non-zero.

37625 RATIONALE

37626 The errors come from taking the expected floating-point value and converting it to **int**, which is
 37627 an invalid operation in IEEE Std 754-1985 (since overflow, infinity, and NaN are not
 37628 representable in a type **int**), so should be a domain error.

37629 There are no known implementations that overflow. For overflow to happen, {INT_MAX} must
 37630 be less than $\text{LDBL\_MAX\_EXP} \times \log_2(\text{FLT\_RADIX})$ or {INT_MIN} must be greater than
 37631 $\text{LDBL\_MIN\_EXP} \times \log_2(\text{FLT\_RADIX})$ if subnormals are not supported, or {INT_MIN} must be
 37632 greater than $(\text{LDBL\_MIN\_EXP} - \text{LDBL\_MANT\_DIG}) \times \log_2(\text{FLT\_RADIX})$ if subnormals are
 37633 supported.

37634 FUTURE DIRECTIONS

37635 None.

37636 SEE ALSO

37637 [*feclearexcept\(\)*](#), [*fetestexcept\(\)*](#), [*logb\(\)*](#), [*scalbln\(\)*](#)

37638 XBD Section 4.19 (on page 116), [**<float.h>**](#), [**<math.h>**](#)

37639 CHANGE HISTORY

37640 First released in Issue 4, Version 2.

37641 Issue 5

37642 Moved from X/OPEN UNIX extension to BASE.

37643 Issue 6

37644 The *ilogb()* function is no longer marked as an extension.

37645 The *ilogbf()* and *ilogbl()* functions are added for alignment with the ISO/IEC 9899:1999
 37646 standard.

37647 The RETURN VALUE section is revised for alignment with the ISO/IEC 9899:1999 standard.

37648 Functionality relating to the XSI option is marked.

37649 Issue 7

37650 ISO/IEC 9899:1999 standard, Technical Corrigendum 2 #48 (SD5-XSH-ERN-71), #49, and #79
 37651 (SD5-XSH-ERN-72) are applied.

37652 **NAME**

37653 imaxabs — return absolute value

37654 **SYNOPSIS**

37655 #include <inttypes.h>

37656 intmax_t imaxabs(intmax_t j);

37657 **DESCRIPTION**

37658 CX The functionality described on this reference page is aligned with the ISO C standard. Any
37659 conflict between the requirements described here and the ISO C standard is unintentional. This
37660 volume of POSIX.1-2008 defers to the ISO C standard.

37661 The *imaxabs()* function shall compute the absolute value of an integer *j*. If the result cannot be
37662 represented, the behavior is undefined.

37663 **RETURN VALUE**37664 The *imaxabs()* function shall return the absolute value.37665 **ERRORS**

37666 No errors are defined.

37667 **EXAMPLES**

37668 None.

37669 **APPLICATION USAGE**

37670 The absolute value of the most negative number cannot be represented in two's complement.

37671 **RATIONALE**

37672 None.

37673 **FUTURE DIRECTIONS**

37674 None.

37675 **SEE ALSO**37676 *imaxdiv()*

37677 XBD <inttypes.h>

37678 **CHANGE HISTORY**

37679 First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

NAME

imaxdiv — return quotient and remainder

SYNOPSIS

```
#include <inttypes.h>
```

```
imaxdiv_t imaxdiv(intmax_t numer, intmax_t denom);
```

DESCRIPTION

CX The functionality described on this reference page is aligned with the ISO C standard. Any conflict between the requirements described here and the ISO C standard is unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

The *imaxdiv()* function shall compute *numer* / *denom* and *numer* % *denom* in a single operation.

RETURN VALUE

The *imaxdiv()* function shall return a structure of type **imaxdiv_t**, comprising both the quotient and the remainder. The structure shall contain (in either order) the members *quot* (the quotient) and *rem* (the remainder), each of which has type **intmax_t**.

If either part of the result cannot be represented, the behavior is undefined.

ERRORS

No errors are defined.

EXAMPLES

None.

APPLICATION USAGE

None.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

[*imaxabs\(\)*](#)

XBD [**<inttypes.h>**](#)

CHANGE HISTORY

First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

NAME

inet_addr, inet_ntoa — IPv4 address manipulation

SYNOPSIS

```
#include <arpa/inet.h>

in_addr_t inet_addr(const char *cp);
char *inet_ntoa(struct in_addr in);
```

DESCRIPTION

The *inet_addr()* function shall convert the string pointed to by *cp*, in the standard IPv4 dotted decimal notation, to an integer value suitable for use as an Internet address.

The *inet_ntoa()* function shall convert the Internet host address specified by *in* to a string in the Internet standard dot notation.

The *inet_ntoa()* function need not be thread-safe.

All Internet addresses shall be returned in network order (bytes ordered from left to right).

Values specified using IPv4 dotted decimal notation take one of the following forms:

a.b.c.d When four parts are specified, each shall be interpreted as a byte of data and assigned, from left to right, to the four bytes of an Internet address.

a.b.c When a three-part address is specified, the last part shall be interpreted as a 16-bit quantity and placed in the rightmost two bytes of the network address. This makes the three-part address format convenient for specifying Class B network addresses as "128.net.host".

a.b When a two-part address is supplied, the last part shall be interpreted as a 24-bit quantity and placed in the rightmost three bytes of the network address. This makes the two-part address format convenient for specifying Class A network addresses as "net.host".

a When only one part is given, the value shall be stored directly in the network address without any byte rearrangement.

All numbers supplied as parts in IPv4 dotted decimal notation may be decimal, octal, or hexadecimal, as specified in the ISO C standard (that is, a leading 0x or 0X implies hexadecimal; otherwise, a leading '0' implies octal; otherwise, the number is interpreted as decimal).

RETURN VALUE

Upon successful completion, *inet_addr()* shall return the Internet address. Otherwise, it shall return (*in_addr_t*)(-1).

The *inet_ntoa()* function shall return a pointer to the network address in Internet standard dot notation.

ERRORS

No errors are defined.

37746 EXAMPLES

37747 None.

37748 APPLICATION USAGE

37749 The return value of *inet_ntoa()* may point to static data that may be overwritten by subsequent
37750 calls to *inet_ntoa()*.

37751 RATIONALE

37752 None.

37753 FUTURE DIRECTIONS

37754 None.

37755 SEE ALSO

37756 *endhostent()*, *endnetent()*

37757 XBD <arpa/inet.h>

37758 CHANGE HISTORY

37759 First released in Issue 6. Derived from the XNS, Issue 5.2 specification.

37760 Issue 7

37761 Austin Group Interpretation 1003.1-2001 #156 is applied.

NAME

inet_ntop, inet_pton — convert IPv4 and IPv6 addresses between binary and text form

SYNOPSIS

```
#include <arpa/inet.h>

const char *inet_ntop(int af, const void *restrict src,
    char *restrict dst, socklen_t size);
int inet_pton(int af, const char *restrict src, void *restrict dst);
```

DESCRIPTION

The *inet_ntop()* function shall convert a numeric address into a text string suitable for presentation. The *af* argument shall specify the family of the address. This can be AF_INET or AF_INET6. The *src* argument points to a buffer holding an IPv4 address if the *af* argument is AF_INET, or an IPv6 address if the *af* argument is AF_INET6; the address must be in network byte order. The *dst* argument points to a buffer where the function stores the resulting text string; it shall not be NULL. The *size* argument specifies the size of this buffer, which shall be large enough to hold the text string (INET_ADDRSTRLEN characters for IPv4, INET6_ADDRSTRLEN characters for IPv6).

The *inet_pton()* function shall convert an address in its standard text presentation form into its numeric binary form. The *af* argument shall specify the family of the address. The AF_INET and AF_INET6 address families shall be supported. The *src* argument points to the string being passed in. The *dst* argument points to a buffer into which the function stores the numeric address; this shall be large enough to hold the numeric address (32 bits for AF_INET, 128 bits for AF_INET6).

If the *af* argument of *inet_pton()* is AF_INET, the *src* string shall be in the standard IPv4 dotted-decimal form:

```
ddd.ddd.ddd.ddd
```

where "ddd" is a one to three digit decimal number between 0 and 255 (see *inet_addr()*). The *inet_pton()* function does not accept other formats (such as the octal numbers, hexadecimal numbers, and fewer than four numbers that *inet_addr()* accepts).

If the *af* argument of *inet_pton()* is AF_INET6, the *src* string shall be in one of the following standard IPv6 text forms:

1. The preferred form is "x:x:x:x:x:x:x:x", where the 'x's are the hexadecimal values of the eight 16-bit pieces of the address. Leading zeros in individual fields can be omitted, but there shall be at least one numeral in every field.
2. A string of contiguous zero fields in the preferred form can be shown as "::". The "::" can only appear once in an address. Unspecified addresses ("0:0:0:0:0:0:0:0") may be represented simply as "::".
3. A third form that is sometimes more convenient when dealing with a mixed environment of IPv4 and IPv6 nodes is "x:x:x:x:x:x.d.d.d.d", where the 'x's are the hexadecimal values of the six high-order 16-bit pieces of the address, and the 'd's are the decimal values of the four low-order 8-bit pieces of the address (standard IPv4 representation).

Note: A more extensive description of the standard representations of IPv6 addresses can be found in RFC 2373.

RETURN VALUE

The *inet_ntop()* function shall return a pointer to the buffer containing the text string if the conversion succeeds, and NULL otherwise, and set *errno* to indicate the error.

The *inet_pton()* function shall return 1 if the conversion succeeds, with the address pointed to by *dst* in network byte order. It shall return 0 if the input is not a valid IPv4 dotted-decimal string or a valid IPv6 address string, or -1 with *errno* set to [EAFNOSUPPORT] if the *af* argument is unknown.

ERRORS

The *inet_ntop()* and *inet_pton()* functions shall fail if:

[EAFNOSUPPORT]

The *af* argument is invalid.

[ENOSPC]

The size of the *inet_ntop()* result buffer is inadequate.

EXAMPLES

None.

APPLICATION USAGE

None.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

XBD [<arpa/inet.h>](#)

CHANGE HISTORY

First released in Issue 6. Derived from the XNS, Issue 5.2 specification.

IPv6 extensions are marked.

The **restrict** keyword is added to the *inet_ntop()* and *inet_pton()* prototypes for alignment with the ISO/IEC 9899:1999 standard.

IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/29 is applied, adding “the address must be in network byte order” to the end of the fourth sentence of the first paragraph in the DESCRIPTION.

NAME

initstate, random, setstate, srandom — pseudo-random number functions

SYNOPSIS

```
XSI #include <stdlib.h>

char *initstate(unsigned seed, char *state, size_t size);
long random(void);
char *setstate(char *state);
void srandom(unsigned seed);
```

DESCRIPTION

The *random()* function shall use a non-linear additive feedback random-number generator employing a default state array size of 31 **long** integers to return successive pseudo-random numbers in the range from 0 to $2^{31}-1$. The period of this random-number generator is approximately $16 \times (2^{31}-1)$. The size of the state array determines the period of the random-number generator. Increasing the state array size shall increase the period.

With 256 bytes of state information, the period of the random-number generator shall be greater than 2^{69} .

Like *rand()*, *random()* shall produce by default a sequence of numbers that can be duplicated by calling *srandom()* with 1 as the seed.

The *srandom()* function shall initialize the current state array using the value of *seed*.

The *initstate()* and *setstate()* functions handle restarting and changing random-number generators. The *initstate()* function allows a state array, pointed to by the *state* argument, to be initialized for future use. The *size* argument, which specifies the size in bytes of the state array, shall be used by *initstate()* to decide what type of random-number generator to use; the larger the state array, the more random the numbers. Values for the amount of state information are 8, 32, 64, 128, and 256 bytes. Other values greater than 8 bytes are rounded down to the nearest one of these values. If *initstate()* is called with $8 \leq \text{size} < 32$, then *random()* shall use a simple linear congruential random number generator. The *seed* argument specifies a starting point for the random-number sequence and provides for restarting at the same point. The *initstate()* function shall return a pointer to the previous state information array.

If *initstate()* has not been called, then *random()* shall behave as though *initstate()* had been called with *seed*=1 and *size*=128.

Once a state has been initialized, *setstate()* allows switching between state arrays. The array defined by the *state* argument shall be used for further random-number generation until *initstate()* is called or *setstate()* is called again. The *setstate()* function shall return a pointer to the previous state array.

RETURN VALUE

If *initstate()* is called with *size* less than 8, it shall return NULL.

The *random()* function shall return the generated pseudo-random number.

The *srandom()* function shall not return a value.

Upon successful completion, *initstate()* and *setstate()* shall return a pointer to the previous state array; otherwise, a null pointer shall be returned.

ERRORS

No errors are defined.

EXAMPLES

None.

APPLICATION USAGE

After initialization, a state array can be restarted at a different point in one of two ways:

1. The *initstate()* function can be used, with the desired seed, state array, and size of the array.
2. The *setstate()* function, with the desired state, can be used, followed by *srandom()* with the desired seed. The advantage of using both of these functions is that the size of the state array does not have to be saved once it is initialized.

Although some implementations of *random()* have written messages to standard error, such implementations do not conform to POSIX.1-2008.

Issue 5 restored the historical behavior of this function.

Threaded applications should use *erand48()*, *nrand48()*, or *jrand48()* instead of *random()* when an independent random number sequence in multiple threads is required.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

drand48(), *rand()*

XBD [<stdlib.h>](#)

CHANGE HISTORY

First released in Issue 4, Version 2.

Issue 5

Moved from X/OPEN UNIX extension to BASE.

In the DESCRIPTION, the phrase “values smaller than 8” is replaced with “values greater than or equal to 8, or less than 32”, “*size*<8” is replaced with “*8*≤*size* <32”, and a new first paragraph is added to the RETURN VALUE section. A note is added to the APPLICATION USAGE indicating that these changes restore the historical behavior of the function.

Issue 6

In the DESCRIPTION, duplicate text “For values greater than or equal to 8...” is removed.

IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/30 is applied, removing *rand_r()* from the list of suggested functions in the APPLICATION USAGE section.

Issue 7

The type of the first argument to *setstate()* is changed from **const char *** to **char ***.

NAME

insque, remque — insert or remove an element in a queue

SYNOPSIS

```
XSI    #include <search.h>

void insque(void *element, void *pred);
void remque(void *element);
```

DESCRIPTION

The *insque()* and *remque()* functions shall manipulate queues built from doubly-linked lists. The queue can be either circular or linear. An application using *insque()* or *remque()* shall ensure it defines a structure in which the first two members of the structure are pointers to the same type of structure, and any further members are application-specific. The first member of the structure is a forward pointer to the next entry in the queue. The second member is a backward pointer to the previous entry in the queue. If the queue is linear, the queue is terminated with null pointers. The names of the structure and of the pointer members are not subject to any special restriction.

The *insque()* function shall insert the element pointed to by *element* into a queue immediately after the element pointed to by *pred*.

The *remque()* function shall remove the element pointed to by *element* from a queue.

If the queue is to be used as a linear list, invoking *insque(&element, NULL)*, where *element* is the initial element of the queue, shall initialize the forward and backward pointers of *element* to null pointers.

If the queue is to be used as a circular list, the application shall ensure it initializes the forward pointer and the backward pointer of the initial element of the queue to the element's own address.

RETURN VALUE

The *insque()* and *remque()* functions do not return a value.

ERRORS

No errors are defined.

EXAMPLES**Creating a Linear Linked List**

The following example creates a linear linked list.

```
#include <search.h>
...
struct myque element1;
struct myque element2;

char *data1 = "DATA1";
char *data2 = "DATA2";
...
element1.data = data1;
element2.data = data2;

insque (&element1, NULL);
insque (&element2, &element1);
```

Creating a Circular Linked List

The following example creates a circular linked list.

```
#include <search.h>
...
struct myque element1;
struct myque element2;

char *data1 = "DATA1";
char *data2 = "DATA2";
...
element1.data = data1;
element2.data = data2;

element1.fwd = &element1;
element1.bck = &element1;

insque (&element2, &element1);
```

Removing an Element

The following example removes the element pointed to by *element1*.

```
#include <search.h>
...
struct myque element1;
...
remque (&element1);
```

APPLICATION USAGE

The historical implementations of these functions described the arguments as being of type **struct qelem *** rather than as being of type **void *** as defined here. In those implementations, **struct qelem** was commonly defined in **<search.h>** as:

```
struct qelem {
    struct qelem *q_forw;
    struct qelem *q_back;
};
```

Applications using these functions, however, were never able to use this structure directly since it provided no room for the actual data contained in the elements. Most applications defined structures that contained the two pointers as the initial elements and also provided space for, or pointers to, the object's data. Applications that used these functions to update more than one type of table also had the problem of specifying two or more different structures with the same name, if they literally used **struct qelem** as specified.

As described here, the implementations were actually expecting a structure type where the first two members were forward and backward pointers to structures. With C compilers that didn't provide function prototypes, applications used structures as specified in the DESCRIPTION above and the compiler did what the application expected.

If this method had been carried forward with an ISO C standard compiler and the historical function prototype, most applications would have to be modified to cast pointers to the structures actually used to be pointers to **struct qelem** to avoid compilation warnings. By specifying **void *** as the argument type, applications do not need to change (unless they specifically referenced **struct qelem** and depended on it being defined in **<search.h>**).

37999 **RATIONALE**

38000 None.

38001 **FUTURE DIRECTIONS**

38002 None.

38003 **SEE ALSO**38004 XBD [<search.h>](#)38005 **CHANGE HISTORY**

38006 First released in Issue 4, Version 2.

38007 **Issue 5**

38008 Moved from X/OPEN UNIX extension to BASE.

38009 **Issue 6**

38010 The normative text is updated to avoid use of the term “must” for application requirements.

38011 **NAME**38012 **ioctl** — control a STREAMS device (**STREAMS**)38013 **SYNOPSIS**

```
38014 OB XSR  #include <stropts.h>
38015         int ioctl(int fildes, int request, ... /* arg */);
```

38016 **DESCRIPTION**

38017 The *ioctl()* function shall perform a variety of control functions on STREAMS devices. For non-
 38018 STREAMS devices, the functions performed by this call are unspecified. The *request* argument
 38019 and an optional third argument (with varying type) shall be passed to and interpreted by the
 38020 appropriate part of the STREAM associated with *fildes*.

38021 The *fildes* argument is an open file descriptor that refers to a device.

38022 The *request* argument selects the control function to be performed and shall depend on the
 38023 STREAMS device being addressed.

38024 The *arg* argument represents additional information that is needed by this specific STREAMS
 38025 device to perform the requested function. The type of *arg* depends upon the particular control
 38026 request, but it shall be either an integer or a pointer to a device-specific data structure.

38027 The *ioctl()* commands applicable to STREAMS, their arguments, and error conditions that apply
 38028 to each individual command are described below.

38029 The following *ioctl()* commands, with error values indicated, are applicable to all STREAMS
 38030 files:

38031 **I_PUSH** Pushes the module whose name is pointed to by *arg* onto the top of the current
 38032 STREAM, just below the STREAM head. It then calls the *open()* function of the
 38033 newly-pushed module.

38034 The *ioctl()* function with the **I_PUSH** command shall fail if:

38035 [EINVAL] Invalid module name.

38036 [ENXIO] Open function of new module failed.

38037 [ENXIO] Hangup received on *fildes*.

38038 **I_POP** Removes the module just below the STREAM head of the STREAM pointed to
 38039 by *fildes*. The *arg* argument should be 0 in an **I_POP** request.

38040 The *ioctl()* function with the **I_POP** command shall fail if:

38041 [EINVAL] No module present in the STREAM.

38042 [ENXIO] Hangup received on *fildes*.

38043 **I_LOOK** Retrieves the name of the module just below the STREAM head of the
 38044 STREAM pointed to by *fildes*, and places it in a character string pointed to by
 38045 *arg*. The buffer pointed to by *arg* should be at least FMNAMESZ+1 bytes long,
 38046 where FMNAMESZ is defined in **<stropts.h>**.

38047 The *ioctl()* function with the **I_LOOK** command shall fail if:

38048 [EINVAL] No module present in the STREAM.

38049 **I_FLUSH** Flushes read and/or write queues, depending on the value of *arg*. Valid *arg*
 38050 values are:

38051	FLUSHR	Flush all read queues.
38052	FLUSHW	Flush all write queues.
38053	FLUSHRW	Flush all read and all write queues.
38054	The <i>ioctl()</i> function with the I_FLUSH command shall fail if:	
38055	[EINVAL]	Invalid <i>arg</i> value.
38056	[EAGAIN] or [ENOSR]	Unable to allocate buffers for flush message.
38057		
38058	[ENXIO]	Hangup received on <i>fildev</i> .
38059	I_FLUSHBAND	Flushes a particular band of messages. The <i>arg</i> argument points to a bandinfo structure. The <i>bi_flag</i> member may be one of FLUSHR, FLUSHW, or FLUSHRW as described above. The <i>bi_pri</i> member determines the priority band to be flushed.
38060		
38061		
38062		
38063	I_SETSIG	Requests that the STREAMS implementation send the SIGPOLL signal to the calling process when a particular event has occurred on the STREAM associated with <i>fildev</i> . I_SETSIG supports an asynchronous processing capability in STREAMS. The value of <i>arg</i> is a bitmask that specifies the events for which the process should be signaled. It is the bitwise-inclusive OR of any combination of the following constants:
38064		
38065		
38066		
38067		A normal (priority band set to 0) message has arrived at the head of a STREAM head read queue. A signal shall be generated even if the message is of zero length.
38068	S_RDNORM	
38069		
38070		
38071		A message with a non-zero priority band has arrived at the head of a STREAM head read queue. A signal shall be generated even if the message is of zero length.
38072	S_RDBAND	
38073		
38074		
38075	S_INPUT	A message, other than a high-priority message, has arrived at the head of a STREAM head read queue. A signal shall be generated even if the message is of zero length.
38076		
38077		
38078		
38079	S_HIPRI	A high-priority message is present on a STREAM head read queue. A signal shall be generated even if the message is of zero length.
38080		
38081		
38082		
38083	S_OUTPUT	The write queue for normal data (priority band 0) just below the STREAM head is no longer full. This notifies the process that there is room on the queue for sending (or writing) normal data downstream.
38084		
38085		
38086		
38087	S_WRNORM	Equivalent to S_OUTPUT.
38088		
38089		
38090		
38091	S_WRBAND	The write queue for a non-zero priority band just below the STREAM head is no longer full. This notifies the process that there is room on the queue for sending (or writing) priority data downstream.
38092		
38090	S_MSG	A STREAMS signal message that contains the SIGPOLL signal has reached the front of the STREAM head read queue.
38091		
38092		

38093		S_ERROR	Notification of an error condition has reached the STREAM head.
38094			
38095		S_HANGUP	Notification of a hangup has reached the STREAM head.
38096		S_BANDURG	When used in conjunction with S_RDBAND, SIGURG is generated instead of SIGPOLL when a priority message reaches the front of the STREAM head read queue.
38097			
38098			
38099			If <i>arg</i> is 0, the calling process shall be unregistered and shall not receive further SIGPOLL signals for the stream associated with <i>fildev</i> .
38100			
38101			Processes that wish to receive SIGPOLL signals shall ensure that they explicitly register to receive them using I_SETSIG. If several processes register to receive this signal for the same event on the same STREAM, each process shall be signaled when the event occurs.
38102			
38103			
38104			
38105			The <i>ioctl()</i> function with the I_SETSIG command shall fail if:
38106		[EINVAL]	The value of <i>arg</i> is invalid.
38107		[EINVAL]	The value of <i>arg</i> is 0 and the calling process is not registered to receive the SIGPOLL signal.
38108			
38109		[EAGAIN]	There were insufficient resources to store the signal request.
38110	I_GETSIG		Returns the events for which the calling process is currently registered to be sent a SIGPOLL signal. The events are returned as a bitmask in an int pointed to by <i>arg</i> , where the events are those specified in the description of I_SETSIG above.
38111			
38112			
38113			
38114			The <i>ioctl()</i> function with the I_GETSIG command shall fail if:
38115		[EINVAL]	Process is not registered to receive the SIGPOLL signal.
38116	I_FIND		Compares the names of all modules currently present in the STREAM to the name pointed to by <i>arg</i> , and returns 1 if the named module is present in the STREAM, or returns 0 if the named module is not present.
38117			
38118			
38119			The <i>ioctl()</i> function with the I_FIND command shall fail if:
38120		[EINVAL]	<i>arg</i> does not contain a valid module name.
38121	I_PEEK		Retrieves the information in the first message on the STREAM head read queue without taking the message off the queue. It is analogous to <i>getmsg()</i> except that this command does not remove the message from the queue. The <i>arg</i> argument points to a strpeek structure.
38122			
38123			
38124			
38125			The application shall ensure that the <i>maxlen</i> member in the ctlbuf and databuf structures is set to the number of bytes of control information and/or data information, respectively, to retrieve. The <i>flags</i> member may be marked RS_HIPRI or 0, as described by <i>getmsg()</i> . If the process sets <i>flags</i> to RS_HIPRI, for example, I_PEEK shall only look for a high-priority message on the STREAM head read queue.
38126			
38127			
38128			
38129			
38130			
38131			I_PEEK returns 1 if a message was retrieved, and returns 0 if no message was found on the STREAM head read queue, or if the RS_HIPRI flag was set in <i>flags</i> and a high-priority message was not present on the STREAM head read queue. It does not wait for a message to arrive. On return, ctlbuf specifies information in the control buffer, databuf specifies information in the data
38132			
38133			
38134			
38135			

38136		buffer, and <i>flags</i> contains the value RS_HIPRI or 0.
38137	I_SRDOPT	Sets the read mode using the value of the argument <i>arg</i> . Read modes are
38138		described in <i>read()</i> . Valid <i>arg</i> flags are:
38139		RNORM Byte-stream mode, the default.
38140		RMSGD Message-discard mode.
38141		RMSGN Message-nondiscard mode.
38142		The bitwise-inclusive OR of RMSGD and RMSGN shall return [EINVAL]. The
38143		bitwise-inclusive OR of RNORM and either RMSGD or RMSGN shall result in
38144		the other flag overriding RNORM which is the default.
38145		In addition, treatment of control messages by the STREAM head may be
38146		changed by setting any of the following flags in <i>arg</i> :
38147		RPROTNORM Fail <i>read()</i> with [EBADMSG] if a message containing a
38148		control part is at the front of the STREAM head read queue.
38149		RPROTDAT Deliver the control part of a message as data when a process
38150		issues a <i>read()</i> .
38151		RPROTDIS Discard the control part of a message, delivering any data
38152		portion, when a process issues a <i>read()</i> .
38153		The <i>ioctl()</i> function with the I_SRDOPT command shall fail if:
38154		[EINVAL] The <i>arg</i> argument is not valid.
38155	I_GRDOPT	Returns the current read mode setting, as described above, in an int pointed to
38156		by the argument <i>arg</i> . Read modes are described in <i>read()</i> .
38157	I_NREAD	Counts the number of data bytes in the data part of the first message on the
38158		STREAM head read queue and places this value in the int pointed to by <i>arg</i> .
38159		The return value for the command shall be the number of messages on the
38160		STREAM head read queue. For example, if 0 is returned in <i>arg</i> , but the <i>ioctl()</i>
38161		return value is greater than 0, this indicates that a zero-length message is next
38162		on the queue.
38163	I_FDINSERT	Creates a message from specified buffer(s), adds information about another
38164		STREAM, and sends the message downstream. The message contains a
38165		control part and an optional data part. The data and control parts to be sent
38166		are distinguished by placement in separate buffers, as described below. The
38167		<i>arg</i> argument points to a strfdinsert structure.
38168		The application shall ensure that the <i>len</i> member in the ctlbuf strbuf structure
38169		is set to the size of a t_uscalar_t plus the number of bytes of control
38170		information to be sent with the message. The <i>fildev</i> member specifies the file
38171		descriptor of the other STREAM, and the <i>offset</i> member, which must be
38172		suitably aligned for use as a t_uscalar_t , specifies the offset from the start of
38173		the control buffer where I_FDINSERT shall store a t_uscalar_t whose
38174		interpretation is specific to the STREAM end. The application shall ensure that
38175		the <i>len</i> member in the databuf strbuf structure is set to the number of bytes of
38176		data information to be sent with the message, or to 0 if no data part is to be
38177		sent.
38178		The <i>flags</i> member specifies the type of message to be created. A normal
38179		message is created if <i>flags</i> is set to 0, and a high-priority message is created if

38180		<i>flags</i> is set to RS_HIPRI. For non-priority messages, I_FDINSERT shall block if
38181		the STREAM write queue is full due to internal flow control conditions. For
38182		priority messages, I_FDINSERT does not block on this condition. For non-
38183		priority messages, I_FDINSERT does not block when the write queue is full
38184		and O_NONBLOCK is set. Instead, it fails and sets <i>errno</i> to [EAGAIN].
38185		I_FDINSERT also blocks, unless prevented by lack of internal resources,
38186		waiting for the availability of message blocks in the STREAM, regardless of
38187		priority or whether O_NONBLOCK has been specified. No partial message is
38188		sent.
38189		The <i>ioctl()</i> function with the I_FDINSERT command shall fail if:
38190	[EAGAIN]	A non-priority message is specified, the O_NONBLOCK
38191		flag is set, and the STREAM write queue is full due to
38192		internal flow control conditions.
38193	[EAGAIN] or [ENOSR]	
38194		Buffers cannot be allocated for the message that is to be
38195		created.
38196	[EINVAL]	One of the following:
38197		— The <i>fildes</i> member of the strfdinsert structure is not a
38198		valid, open STREAM file descriptor.
38199		— The size of a t_uscalar_t plus <i>offset</i> is greater than the
38200		<i>len</i> member for the buffer specified through ctlbuf .
38201		— The <i>offset</i> member does not specify a properly-aligned
38202		location in the data buffer.
38203		— An undefined value is stored in <i>flags</i> .
38204	[ENXIO]	Hangup received on the STREAM identified by either the
38205		<i>fildes</i> argument or the <i>fildes</i> member of the strfdinsert
38206		structure.
38207	[ERANGE]	The <i>len</i> member for the buffer specified through databuf
38208		does not fall within the range specified by the maximum
38209		and minimum packet sizes of the topmost STREAM
38210		module; or the <i>len</i> member for the buffer specified through
38211		databuf is larger than the maximum configured size of the
38212		data part of a message; or the <i>len</i> member for the buffer
38213		specified through ctlbuf is larger than the maximum
38214		configured size of the control part of a message.
38215	I_STR	Constructs an internal STREAMS <i>ioctl()</i> message from the data pointed to by
38216		<i>arg</i> , and sends that message downstream.
38217		This mechanism is provided to send <i>ioctl()</i> requests to downstream modules
38218		and drivers. It allows information to be sent with <i>ioctl()</i> , and returns to the
38219		process any information sent upstream by the downstream recipient. I_STR
38220		shall block until the system responds with either a positive or negative
38221		acknowledgement message, or until the request times out after some period of
38222		time. If the request times out, it shall fail with <i>errno</i> set to [ETIME].
38223		At most, one I_STR can be active on a STREAM. Further I_STR calls shall
38224		block until the active I_STR completes at the STREAM head. The default

38225		timeout interval for these requests is 15 seconds. The O_NONBLOCK flag has
38226		no effect on this call.
38227		To send requests downstream, the application shall ensure that <i>arg</i> points to a
38228		striocctl structure.
38229		The <i>ic_cmd</i> member is the internal <i>ioctl()</i> command intended for a
38230		downstream module or driver and <i>ic_timeout</i> is the number of seconds
38231		(−1=infinite, 0=use implementation-defined timeout interval, >0=as specified)
38232		an I_STR request shall wait for acknowledgement before timing out. <i>ic_len</i> is
38233		the number of bytes in the data argument, and <i>ic_dp</i> is a pointer to the data
38234		argument. The <i>ic_len</i> member has two uses: on input, it contains the length of
38235		the data argument passed in, and on return from the command, it contains the
38236		number of bytes being returned to the process (the buffer pointed to by <i>ic_dp</i>
38237		should be large enough to contain the maximum amount of data that any
38238		module or the driver in the STREAM can return).
38239		The STREAM head shall convert the information pointed to by the striocctl
38240		structure to an internal <i>ioctl()</i> command message and send it downstream.
38241		The <i>ioctl()</i> function with the I_STR command shall fail if:
38242		[EAGAIN] or [ENOSR]
38243		Unable to allocate buffers for the <i>ioctl()</i> message.
38244		[EINVAL] The <i>ic_len</i> member is less than 0 or larger than the
38245		maximum configured size of the data part of a message, or
38246		<i>ic_timeout</i> is less than −1.
38247		[ENXIO] Hangup received on <i>fildev</i> .
38248		[ETIME] A downstream <i>ioctl()</i> timed out before acknowledgement
38249		was received.
38250		An I_STR can also fail while waiting for an acknowledgement if a message
38251		indicating an error or a hangup is received at the STREAM head. In addition,
38252		an error code can be returned in the positive or negative acknowledgement
38253		message, in the event the <i>ioctl()</i> command sent downstream fails. For these
38254		cases, I_STR shall fail with <i>errno</i> set to the value in the message.
38255	I_SWROPT	Sets the write mode using the value of the argument <i>arg</i> . Valid bit settings for
38256		<i>arg</i> are:
38257		SNDZERO Send a zero-length message downstream when a <i>write()</i> of 0
38258		bytes occurs. To not send a zero-length message when a
38259		<i>write()</i> of 0 bytes occurs, the application shall ensure that
38260		this bit is not set in <i>arg</i> (for example, <i>arg</i> would be set to 0).
38261		The <i>ioctl()</i> function with the I_SWROPT command shall fail if:
38262		[EINVAL] <i>arg</i> is not the above value.
38263	I_GWROPT	Returns the current write mode setting, as described above, in the int that is
38264		pointed to by the argument <i>arg</i> .
38265	I_SENDFD	Creates a new reference to the open file description associated with the file
38266		descriptor <i>arg</i> , and writes a message on the STREAMS-based pipe <i>fildev</i>
38267		containing this reference, together with the user ID and group ID of the calling
38268		process.

38269		The <i>ioctl()</i> function with the I_SENDFD command shall fail if:
38270		[EAGAIN] The sending STREAM is unable to allocate a message block
38271		to contain the file pointer; or the read queue of the receiving
38272		STREAM head is full and cannot accept the message sent by
38273		I_SENDFD.
38274		[EBADF] The <i>arg</i> argument is not a valid, open file descriptor.
38275		[EINVAL] The <i>fildev</i> argument is not connected to a STREAM pipe.
38276		[ENXIO] Hangup received on <i>fildev</i> .
38277		The <i>ioctl()</i> function with the I_SENDFD command may fail if:
38278		[EINVAL] The <i>arg</i> argument is equal to the <i>fildev</i> argument.
38279	I_RECVFD	Retrieves the reference to an open file description from a message written to a
38280		STREAMS-based pipe using the I_SENDFD command, and allocates a new file
38281		descriptor in the calling process that refers to this open file description. The
38282		<i>arg</i> argument is a pointer to a strrecvfd data structure as defined in
38283		<stropts.h>.
38284		The <i>fd</i> member is a file descriptor. The <i>uid</i> and <i>gid</i> members are the effective
38285		user ID and effective group ID, respectively, of the sending process.
38286		If O_NONBLOCK is not set, I_RECVFD shall block until a message is present
38287		at the STREAM head. If O_NONBLOCK is set, I_RECVFD shall fail with <i>errno</i>
38288		set to [EAGAIN] if no message is present at the STREAM head.
38289		If the message at the STREAM head is a message sent by an I_SENDFD, a new
38290		file descriptor shall be allocated for the open file descriptor referenced in
38291		the message. The new file descriptor is placed in the <i>fd</i> member of the strrecvfd
38292		structure pointed to by <i>arg</i> .
38293		The <i>ioctl()</i> function with the I_RECVFD command shall fail if:
38294		[EAGAIN] A message is not present at the STREAM head read queue
38295		and the O_NONBLOCK flag is set.
38296		[EBADMSG] The message at the STREAM head read queue is not a
38297		message containing a passed file descriptor.
38298		[EMFILE] All file descriptors available to the process are currently
38299		open.
38300		[ENXIO] Hangup received on <i>fildev</i> .
38301	I_LIST	Allows the process to list all the module names on the STREAM, up to and
38302		including the topmost driver name. If <i>arg</i> is a null pointer, the return value
38303		shall be the number of modules, including the driver, that are on the STREAM
38304		pointed to by <i>fildev</i> . This lets the process allocate enough space for the module
38305		names. Otherwise, it should point to a str_list structure.
38306		The <i>sl_nmods</i> member indicates the number of entries the process has
38307		allocated in the array. Upon return, the <i>sl_modlist</i> member of the str_list
38308		structure shall contain the list of module names, and the number of entries
38309		that have been filled into the <i>sl_modlist</i> array is found in the <i>sl_nmods</i> member
38310		(the number includes the number of modules including the driver). The return
38311		value from <i>ioctl()</i> shall be 0. The entries are filled in starting at the top of the

38312		STREAM and continuing downstream until either the end of the STREAM is
38313		reached, or the number of requested modules (<i>sl_nmods</i>) is satisfied.
38314		The <i>ioctl()</i> function with the <i>I_LIST</i> command shall fail if:
38315		[EINVAL] The <i>sl_nmods</i> member is less than 1.
38316		[EAGAIN] or [ENOSR]
38317		Unable to allocate buffers.
38318	<i>I_ATMARK</i>	Allows the process to see if the message at the head of the STREAM head read
38319		queue is marked by some module downstream. The <i>arg</i> argument determines
38320		how the checking is done when there may be multiple marked messages on
38321		the STREAM head read queue. It may take on the following values:
38322		ANYMARK Check if the message is marked.
38323		LASTMARK Check if the message is the last one marked on the queue.
38324		The bitwise-inclusive OR of the flags ANYMARK and LASTMARK is
38325		permitted.
38326		The return value shall be 1 if the mark condition is satisfied; otherwise, the
38327		value shall be 0.
38328		The <i>ioctl()</i> function with the <i>I_ATMARK</i> command shall fail if:
38329		[EINVAL] Invalid <i>arg</i> value.
38330	<i>I_CKBAND</i>	Checks if the message of a given priority band exists on the STREAM head
38331		read queue. This shall return 1 if a message of the given priority exists, 0 if no
38332		such message exists, or -1 on error. <i>arg</i> should be of type int .
38333		The <i>ioctl()</i> function with the <i>I_CKBAND</i> command shall fail if:
38334		[EINVAL] Invalid <i>arg</i> value.
38335	<i>I_GETBAND</i>	Returns the priority band of the first message on the STREAM head read
38336		queue in the integer referenced by <i>arg</i> .
38337		The <i>ioctl()</i> function with the <i>I_GETBAND</i> command shall fail if:
38338		[ENODATA] No message on the STREAM head read queue.
38339	<i>I_CANPUT</i>	Checks if a certain band is writable. <i>arg</i> is set to the priority band in question.
38340		The return value shall be 0 if the band is flow-controlled, 1 if the band is
38341		writable, or -1 on error.
38342		The <i>ioctl()</i> function with the <i>I_CANPUT</i> command shall fail if:
38343		[EINVAL] Invalid <i>arg</i> value.
38344	<i>I_SETCLTIME</i>	This request allows the process to set the time the STREAM head shall delay
38345		when a STREAM is closing and there is data on the write queues. Before
38346		closing each module or driver, if there is data on its write queue, the STREAM
38347		head shall delay for the specified amount of time to allow the data to drain. If,
38348		after the delay, data is still present, it shall be flushed. The <i>arg</i> argument is a
38349		pointer to an integer specifying the number of milliseconds to delay, rounded
38350		up to the nearest valid value. If <i>I_SETCLTIME</i> is not performed on a STREAM,
38351		an implementation-defined default timeout interval is used.

38352		The <i>ioctl()</i> function with the I_SETCLTIME command shall fail if:
38353	[EINVAL]	Invalid <i>arg</i> value.
38354	I_GETCLTIME	Returns the close time delay in the integer pointed to by <i>arg</i> .
38355	Multiplexed STREAMS Configurations	
38356	The following commands are used for connecting and disconnecting multiplexed STREAMS configurations. These commands use an implementation-defined default timeout interval.	
38357		
38358	I_LINK	Connects two STREAMs, where <i>fildev</i> is the file descriptor of the STREAM connected to the multiplexing driver, and <i>arg</i> is the file descriptor of the STREAM connected to another driver. The STREAM designated by <i>arg</i> is connected below the multiplexing driver. I_LINK requires the multiplexing driver to send an acknowledgement message to the STREAM head regarding the connection. This call shall return a multiplexer ID number (an identifier used to disconnect the multiplexer; see I_UNLINK) on success, and -1 on failure.
38359		
38360		
38361		
38362		
38363		
38364		
38365		
38366		The <i>ioctl()</i> function with the I_LINK command shall fail if:
38367	[ENXIO]	Hangup received on <i>fildev</i> .
38368	[ETIME]	Timeout before acknowledgement message was received at STREAM head.
38369		
38370	[EAGAIN] or [ENOSR]	
38371		Unable to allocate STREAMS storage to perform the I_LINK.
38372		
38373	[EBADF]	The <i>arg</i> argument is not a valid, open file descriptor.
38374	[EINVAL]	The <i>fildev</i> argument does not support multiplexing; or <i>arg</i> is not a STREAM or is already connected downstream from a multiplexer; or the specified I_LINK operation would connect the STREAM head in more than one place in the multiplexed STREAM.
38375		
38376		
38377		
38378		
38379		An I_LINK can also fail while waiting for the multiplexing driver to acknowledge the request, if a message indicating an error or a hangup is received at the STREAM head of <i>fildev</i> . In addition, an error code can be returned in the positive or negative acknowledgement message. For these cases, I_LINK fails with <i>errno</i> set to the value in the message.
38380		
38381		
38382		
38383		
38384	I_UNLINK	Disconnects the two STREAMs specified by <i>fildev</i> and <i>arg</i> . <i>fildev</i> is the file descriptor of the STREAM connected to the multiplexing driver. The <i>arg</i> argument is the multiplexer ID number that was returned by the I_LINK <i>ioctl()</i> command when a STREAM was connected downstream from the multiplexing driver. If <i>arg</i> is MUXID_ALL, then all STREAMs that were connected to <i>fildev</i> shall be disconnected. As in I_LINK, this command requires acknowledgement.
38385		
38386		
38387		
38388		
38389		
38390		
38391		The <i>ioctl()</i> function with the I_UNLINK command shall fail if:
38392	[ENXIO]	Hangup received on <i>fildev</i> .

38393		[ETIME]	Timeout before acknowledgement message was received at
38394			STREAM head.
38395		[EAGAIN] or [ENOSR]	
38396			Unable to allocate buffers for the acknowledgement
38397			message.
38398		[EINVAL]	Invalid multiplexer ID number.
38399			An I_UNLINK can also fail while waiting for the multiplexing driver to
38400			acknowledge the request if a message indicating an error or a hangup is
38401			received at the STREAM head of <i>fildev</i> . In addition, an error code can be
38402			returned in the positive or negative acknowledgement message. For these
38403			cases, I_UNLINK shall fail with <i>errno</i> set to the value in the message.
38404	I_PLINK		Creates a <i>persistent connection</i> between two STREAMs, where <i>fildev</i> is the file
38405			descriptor of the STREAM connected to the multiplexing driver, and <i>arg</i> is the
38406			file descriptor of the STREAM connected to another driver. This call shall
38407			create a persistent connection which can exist even if the file descriptor <i>fildev</i>
38408			associated with the upper STREAM to the multiplexing driver is closed. The
38409			STREAM designated by <i>arg</i> gets connected via a persistent connection below
38410			the multiplexing driver. I_PLINK requires the multiplexing driver to send an
38411			acknowledgement message to the STREAM head. This call shall return a
38412			multiplexer ID number (an identifier that may be used to disconnect the
38413			multiplexer; see I_PUNLINK) on success, and -1 on failure.
38414			The <i>ioctl()</i> function with the I_PLINK command shall fail if:
38415		[ENXIO]	Hangup received on <i>fildev</i> .
38416		[ETIME]	Timeout before acknowledgement message was received at
38417			STREAM head.
38418		[EAGAIN] or [ENOSR]	
38419			Unable to allocate STREAMS storage to perform the
38420			I_PLINK.
38421		[EBADF]	The <i>arg</i> argument is not a valid, open file descriptor.
38422		[EINVAL]	The <i>fildev</i> argument does not support multiplexing; or <i>arg</i> is
38423			not a STREAM or is already connected downstream from a
38424			multiplexer; or the specified I_PLINK operation would
38425			connect the STREAM head in more than one place in the
38426			multiplexed STREAM.
38427			An I_PLINK can also fail while waiting for the multiplexing driver to
38428			acknowledge the request, if a message indicating an error or a hangup is
38429			received at the STREAM head of <i>fildev</i> . In addition, an error code can be
38430			returned in the positive or negative acknowledgement message. For these
38431			cases, I_PLINK shall fail with <i>errno</i> set to the value in the message.
38432	I_PUNLINK		Disconnects the two STREAMs specified by <i>fildev</i> and <i>arg</i> from a persistent
38433			connection. The <i>fildev</i> argument is the file descriptor of the STREAM
38434			connected to the multiplexing driver. The <i>arg</i> argument is the multiplexer ID
38435			number that was returned by the I_PLINK <i>ioctl()</i> command when a STREAM
38436			was connected downstream from the multiplexing driver. If <i>arg</i> is
38437			MUXID_ALL, then all STREAMs which are persistent connections to <i>fildev</i>
38438			shall be disconnected. As in I_PLINK, this command requires the multiplexing

38439	driver to acknowledge the request.
38440	The <i>ioctl()</i> function with the I_PUNLINK command shall fail if:
38441	[ENXIO] Hangup received on <i>fildev</i> .
38442	[ETIME] Timeout before acknowledgement message was received at
38443	STREAM head.
38444	[EAGAIN] or [ENOSR]
38445	Unable to allocate buffers for the acknowledgement
38446	message.
38447	[EINVAL] Invalid multiplexer ID number.
38448	An I_PUNLINK can also fail while waiting for the multiplexing driver to
38449	acknowledge the request if a message indicating an error or a hangup is
38450	received at the STREAM head of <i>fildev</i> . In addition, an error code can be
38451	returned in the positive or negative acknowledgement message. For these
38452	cases, I_PUNLINK shall fail with <i>errno</i> set to the value in the message.
38453	RETURN VALUE
38454	Upon successful completion, <i>ioctl()</i> shall return a value other than -1 that depends upon the
38455	STREAMS device control function. Otherwise, it shall return -1 and set <i>errno</i> to indicate the
38456	error.
38457	ERRORS
38458	Under the following general conditions, <i>ioctl()</i> shall fail if:
38459	[EBADF] The <i>fildev</i> argument is not a valid open file descriptor.
38460	[EINTR] A signal was caught during the <i>ioctl()</i> operation.
38461	[EINVAL] The STREAM or multiplexer referenced by <i>fildev</i> is linked (directly or
38462	indirectly) downstream from a multiplexer.
38463	If an underlying device driver detects an error, then <i>ioctl()</i> shall fail if:
38464	[EINVAL] The <i>request</i> or <i>arg</i> argument is not valid for this device.
38465	[EIO] Some physical I/O error has occurred.
38466	[ENOTTY] The file associated with the <i>fildev</i> argument is not a STREAMS device that
38467	accepts control functions.
38468	[ENXIO] The <i>request</i> and <i>arg</i> arguments are valid for this device driver, but the service
38469	requested cannot be performed on this particular sub-device.
38470	[ENODEV] The <i>fildev</i> argument refers to a valid STREAMS device, but the corresponding
38471	device driver does not support the <i>ioctl()</i> function.
38472	If a STREAM is connected downstream from a multiplexer, any <i>ioctl()</i> command except
38473	I_UNLINK and I_PUNLINK shall set <i>errno</i> to [EINVAL].

EXAMPLES

None.

APPLICATION USAGE

The implementation-defined timeout interval for STREAMS has historically been 15 seconds.

RATIONALE

None.

FUTURE DIRECTIONS

The *ioctl()* function may be removed in a future version.

SEE ALSO

Section 2.6 (on page 499), *close()*, *fcntl()*, *getmsg()*, *open()*, *pipe()*, *poll()*, *putmsg()*, *read()*, *sigaction()*, *write()*

XBD <stropts.h>

CHANGE HISTORY

First released in Issue 4, Version 2.

Issue 5

Moved from X/OPEN UNIX extension to BASE.

Issue 6

The Open Group Corrigendum U028/4 is applied, correcting text in the *I_FDINSERT* [EINVAL] case to refer to *ctlbuf*.

This function is marked as part of the XSI STREAMS Option Group.

The normative text is updated to avoid use of the term “must” for application requirements.

Issue 7

Austin Group Interpretation 1003.1-2001 #155 is applied, adding a “may fail” [EINVAL] error condition for the *I_SENDFD* command.

SD5-XSH-ERN-100 is applied, correcting the definition of the [ENOTTY] error condition.

The *ioctl()* function is marked obsolescent.

38500 **NAME**

38501 isalnum, isalnum_l — test for an alphanumeric character

38502 **SYNOPSIS**

38503 #include <ctype.h>

38504 int isalnum(int c);

38505 CX int isalnum_l(int c, locale_t locale);

38506 **DESCRIPTION**38507 CX For *isalnum()*: The functionality described on this reference page is aligned with the ISO C
38508 standard. Any conflict between the requirements described here and the ISO C standard is
38509 unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.38510 CX The *isalnum()* and *isalnum_l()* functions shall test whether *c* is a character of class **alpha** or
38511 CX **digit** in the current locale, or in the locale represented by *locale*, respectively; see XBD Chapter 7
38512 (on page 135).38513 The *c* argument is an **int**, the value of which the application shall ensure is representable as an
38514 **unsigned char** or equal to the value of the macro EOF. If the argument has any other value, the
38515 behavior is undefined.38516 CX The behavior is undefined if the *locale* argument to *isalnum_l()* is the special locale object +
38517 LC_GLOBAL_LOCALE or is not a valid locale object handle.38518 **RETURN VALUE**38519 CX The *isalnum()* and *isalnum_l()* functions shall return non-zero if *c* is an alphanumeric character;
38520 otherwise, they shall return 0.38521 **ERRORS**

38522 No errors are defined.

38523 **EXAMPLES**

38524 None.

38525 **APPLICATION USAGE**38526 To ensure applications portability, especially across natural languages, only these functions and
38527 the functions in the reference pages listed in the SEE ALSO section should be used for character
38528 classification.38529 **RATIONALE**

38530 None.

38531 **FUTURE DIRECTIONS**

38532 None.

38533 **SEE ALSO**38534 *isalpha()*, *isblank()*, *iscntrl()*, *isdigit()*, *isgraph()*, *islower()*, *isprint()*, *ispunct()*, *isspace()*, *isupper()*,
38535 *isxdigit()*, *setlocale()*, *uselocale()*

38536 XBD Chapter 7 (on page 135), <ctype.h>, <stdio.h>

38537 **CHANGE HISTORY**

38538 First released in Issue 1. Derived from Issue 1 of the SVID.

38539 **Issue 6**

38540 The normative text is updated to avoid use of the term “must” for application requirements.

38541 **Issue 7**

38542 The *isalnum_l()* function is added from The Open Group Technical Standard, 2006, Extended
38543 API Set Part 4.

38544 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0274 [302], XSH/TC1-2008/0275 [283], +
38545 and XSH/TC1-2008/0276 [283] are applied.

38546 **NAME**

38547 isalpha, isalpha_l — test for an alphabetic character

38548 **SYNOPSIS**

38549 #include <ctype.h>

38550 int isalpha(int c);

38551 CX int isalpha_l(int c, locale_t locale);

38552 **DESCRIPTION**

38553 CX For *isalpha()*: The functionality described on this reference page is aligned with the ISO C
 38554 standard. Any conflict between the requirements described here and the ISO C standard is
 38555 unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

38556 CX The *isalpha()* and *isalpha_l()* functions shall test whether *c* is a character of class **alpha** in the
 38557 CX current locale, or in the locale represented by *locale*, respectively; see XBD Chapter 7 (on page
 38558 135).

38559 The *c* argument is an **int**, the value of which the application shall ensure is representable as an
 38560 **unsigned char** or equal to the value of the macro EOF. If the argument has any other value, the
 38561 behavior is undefined.

38562 CX The behavior is undefined if the *locale* argument to *isalpha_l()* is the special locale object +
 38563 LC_GLOBAL_LOCALE or is not a valid locale object handle.

38564 **RETURN VALUE**

38565 CX The *isalpha()* and *isalpha_l()* functions shall return non-zero if *c* is an alphabetic character;
 38566 otherwise, they shall return 0.

38567 **ERRORS**

38568 No errors are defined.

38569 **EXAMPLES**

38570 None.

38571 **APPLICATION USAGE**

38572 To ensure applications portability, especially across natural languages, only these functions and
 38573 the functions in the reference pages listed in the SEE ALSO section should be used for character
 38574 classification.

38575 **RATIONALE**

38576 None.

38577 **FUTURE DIRECTIONS**

38578 None.

38579 **SEE ALSO**

38580 *isalnum()*, *isblank()*, *iscntrl()*, *isdigit()*, *isgraph()*, *islower()*, *isprint()*, *ispunct()*, *isspace()*, *isupper()*,
 38581 *isxdigit()*, *setlocale()*, *uselocale()*

38582 XBD Chapter 7 (on page 135), <ctype.h>, <locale.h>, <stdio.h>

38583 **CHANGE HISTORY**

38584 First released in Issue 1. Derived from Issue 1 of the SVID.

38585 **Issue 6**

38586 The normative text is updated to avoid use of the term “must” for application requirements.

38587 **Issue 7**38588
38589

The *isalpha_l()* function is added from The Open Group Technical Standard, 2006, Extended API Set Part 4.

38590
38591

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0277 [302], XSH/TC1-2008/0278 [283], + and XSH/TC1-2008/0279 [283] are applied.

38592 **NAME**

38593 isascii — test for a 7-bit US-ASCII character

38594 **SYNOPSIS**

```
38595 OB XSI  #include <ctype.h>
38596         int isascii(int c);
```

38597 **DESCRIPTION**38598 The *isascii()* function shall test whether *c* is a 7-bit US-ASCII character code.38599 The *isascii()* function is defined on all integer values.38600 **RETURN VALUE**38601 The *isascii()* function shall return non-zero if *c* is a 7-bit US-ASCII character code between 0 and
38602 octal 0177 inclusive; otherwise, it shall return 0.38603 **ERRORS**

38604 No errors are defined.

38605 **EXAMPLES**

38606 None.

38607 **APPLICATION USAGE**38608 The *isascii()* function cannot be used portably in a localized application.38609 **RATIONALE**

38610 None.

38611 **FUTURE DIRECTIONS**38612 The *isascii()* function may be removed in a future version.38613 **SEE ALSO**38614 XBD [<ctype.h>](#)38615 **CHANGE HISTORY**

38616 First released in Issue 1. Derived from Issue 1 of the SVID.

38617 **Issue 7**38618 The *isascii()* function is marked obsolescent.

38619 **NAME**38620 isastream — test a file descriptor (**STREAMS**)38621 **SYNOPSIS**

```
38622 OB XSR #include <stropts.h>  
38623 int isastream(int fildes);
```

38624 **DESCRIPTION**

38625 The *isastream()* function shall test whether *fildes*, an open file descriptor, is associated with a
38626 STREAMS-based file.

38627 **RETURN VALUE**

38628 Upon successful completion, *isastream()* shall return 1 if *fildes* refers to a STREAMS-based file
38629 and 0 if not. Otherwise, *isastream()* shall return -1 and set *errno* to indicate the error.

38630 **ERRORS**

38631 The *isastream()* function shall fail if:

38632 [EBADF] The *fildes* argument is not a valid open file descriptor.

38633 **EXAMPLES**

38634 None.

38635 **APPLICATION USAGE**

38636 None.

38637 **RATIONALE**

38638 None.

38639 **FUTURE DIRECTIONS**

38640 The *isastream()* function may be removed in a future version.

38641 **SEE ALSO**

38642 XBD [<stropts.h>](#)

38643 **CHANGE HISTORY**

38644 First released in Issue 4, Version 2.

38645 **Issue 5**

38646 Moved from X/OPEN UNIX extension to BASE.

38647 **Issue 7**

38648 The *isastream()* function is marked obsolescent.

NAME

isatty — test for a terminal device

SYNOPSIS

```
#include <unistd.h>
```

```
int isatty(int fildes);
```

DESCRIPTION

The *isatty()* function shall test whether *fildes*, an open file descriptor, is associated with a terminal device.

RETURN VALUE

The *isatty()* function shall return 1 if *fildes* is associated with a terminal; otherwise, it shall return 0 and may set *errno* to indicate the error.

ERRORS

The *isatty()* function may fail if:

[EBADF] The *fildes* argument is not a valid open file descriptor.

[ENOTTY] The file associated with the *fildes* argument is not a terminal.

EXAMPLES

None.

APPLICATION USAGE

The *isatty()* function does not necessarily indicate that a human being is available for interaction via *fildes*. It is quite possible that non-terminal devices are connected to the communications line.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

XBD [**<unistd.h>**](#)

CHANGE HISTORY

First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The optional setting of *errno* to indicate an error is added.
- The [EBADF] and [ENOTTY] optional error conditions are added.

Issue 7

SD5-XSH-ERN-100 is applied, correcting the definition of the [ENOTTY] error condition.

38685 **NAME**

38686 isblank, isblank_l — test for a blank character

38687 **SYNOPSIS**

```
38688     #include <ctype.h>
38689
38689     int isblank(int c);
38690 CX     int isblank_l(int c, locale_t locale);
```

38691 **DESCRIPTION**

38692 CX For *isblank()*: The functionality described on this reference page is aligned with the ISO C
 38693 standard. Any conflict between the requirements described here and the ISO C standard is
 38694 unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

38695 CX The *isblank()* and *isblank_l()* functions shall test whether *c* is a character of class **blank** in the
 38696 CX current locale, or in the locale represented by *locale*, respectively; see XBD Chapter 7 (on page
 38697 135).

38698 The *c* argument is a type **int**, the value of which the application shall ensure is a character
 38699 representable as an **unsigned char** or equal to the value of the macro EOF. If the argument has
 38700 any other value, the behavior is undefined.

38701 CX The behavior is undefined if the *locale* argument to *isblank_l()* is the special locale object +
 38702 LC_GLOBAL_LOCALE or is not a valid locale object handle.

38703 **RETURN VALUE**

38704 CX The *isblank()* and *isblank_l()* functions shall return non-zero if *c* is a <blank>; otherwise, they
 38705 shall return 0.

38706 **ERRORS**

38707 No errors are defined.

38708 **EXAMPLES**

38709 None.

38710 **APPLICATION USAGE**

38711 To ensure applications portability, especially across natural languages, only these functions and
 38712 the functions in the reference pages listed in the SEE ALSO section should be used for character
 38713 classification.

38714 **RATIONALE**

38715 None.

38716 **FUTURE DIRECTIONS**

38717 None.

38718 **SEE ALSO**

38719 *isalnum()*, *isalpha()*, *iscntrl()*, *isdigit()*, *isgraph()*, *islower()*, *isprint()*, *ispunct()*, *isspace()*, *isupper()*,
 38720 *isxdigit()*, *setlocale()*, *uselocale()*

38721 XBD Chapter 7 (on page 135), <ctype.h>, <locale.h>

38722 **CHANGE HISTORY**

38723 First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

38724 **Issue 7**

38725 The *isblank_l()* function is added from The Open Group Technical Standard, 2006, Extended API
 38726 Set Part 4.


38727
38728

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0280 [302], XSH/TC1-2008/0281 [283], +
and XSH/TC1-2008/0282 [283] are applied.



38729 **NAME**

38730 isctrl, isctrl_l — test for a control character

38731 **SYNOPSIS**

38732 #include <ctype.h>

38733 int isctrl(int c);

38734 CX int isctrl_l(int c, locale_t locale);

38735 **DESCRIPTION**

38736 CX For *isctrl()*: The functionality described on this reference page is aligned with the ISO C
 38737 standard. Any conflict between the requirements described here and the ISO C standard is
 38738 unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

38739 CX The *isctrl()* and *isctrl_l()* functions shall test whether *c* is a character of class **cntrl** in the
 38740 CX current locale, or in the locale represented by *locale*, respectively; see XBD Chapter 7 (on page
 38741 135).

38742 The *c* argument is a type **int**, the value of which the application shall ensure is a character
 38743 representable as an **unsigned char** or equal to the value of the macro EOF. If the argument has
 38744 any other value, the behavior is undefined.

38745 CX The behavior is undefined if the *locale* argument to *isctrl_l()* is the special locale object +
 38746 LC_GLOBAL_LOCALE or is not a valid locale object handle.

38747 **RETURN VALUE**

38748 CX The *isctrl()* and *isctrl_l()* functions shall return non-zero if *c* is a control character; otherwise,
 38749 they shall return 0.

38750 **ERRORS**

38751 No errors are defined.

38752 **EXAMPLES**

38753 None.

38754 **APPLICATION USAGE**

38755 To ensure applications portability, especially across natural languages, only these functions and
 38756 the functions in the reference pages listed in the SEE ALSO section should be used for character
 38757 classification.

38758 **RATIONALE**

38759 None.

38760 **FUTURE DIRECTIONS**

38761 None.

38762 **SEE ALSO**

38763 *isalnum()*, *isalpha()*, *isblank()*, *isdigit()*, *isgraph()*, *islower()*, *isprint()*, *ispunct()*, *isspace()*,
 38764 *isupper()*, *isxdigit()*, *setlocale()*, *uselocale()*

38765 XBD Chapter 7 (on page 135), <ctype.h>, <locale.h>

38766 **CHANGE HISTORY**

38767 First released in Issue 1. Derived from Issue 1 of the SVID.

38768 **Issue 6**

38769 The normative text is updated to avoid use of the term “must” for application requirements.

38770 **Issue 7**

38771 The *iscntrl_l()* function is added from The Open Group Technical Standard, 2006, Extended API
38772 Set Part 4.

38773 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0283 [302], XSH/TC1-2008/0284 [283], +
38774 and XSH/TC1-2008/0285 [283] are applied.

38775 **NAME**38776 `isdigit, isdigit_l` — test for a decimal digit38777 **SYNOPSIS**

```
38778        #include <ctype.h>
38779        int isdigit(int c);
38780 CX     int isdigit_l(int c, locale_t locale);
```

38781 **DESCRIPTION**

38782 CX For `isdigit()`: The functionality described on this reference page is aligned with the ISO C
 38783 standard. Any conflict between the requirements described here and the ISO C standard is
 38784 unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

38785 CX The `isdigit()` and `isdigit_l()` functions shall test whether *c* is a character of class **digit** in the
 38786 CX current locale, or in the locale represented by *locale*, respectively; see XBD Chapter 7 (on page
 38787 135).

38788 The *c* argument is an **int**, the value of which the application shall ensure is a character
 38789 representable as an **unsigned char** or equal to the value of the macro EOF. If the argument has
 38790 any other value, the behavior is undefined.

38791 CX The behavior is undefined if the *locale* argument to `isdigit_l()` is the special locale object +
 38792 LC_GLOBAL_LOCALE or is not a valid locale object handle.

38793 **RETURN VALUE**

38794 CX The `isdigit()` and `isdigit_l()` functions shall return non-zero if *c* is a decimal digit; otherwise,
 38795 they shall return 0.

38796 **ERRORS**

38797 No errors are defined.

38798 **EXAMPLES**

38799 None.

38800 **APPLICATION USAGE**

38801 To ensure applications portability, especially across natural languages, only these functions and
 38802 the functions in the reference pages listed in the SEE ALSO section should be used for character
 38803 classification.

38804 **RATIONALE**

38805 None.

38806 **FUTURE DIRECTIONS**

38807 None.

38808 **SEE ALSO**

38809 [*isalnum\(\)*](#), [*isalpha\(\)*](#), [*isblank\(\)*](#), [*iscntrl\(\)*](#), [*isgraph\(\)*](#), [*islower\(\)*](#), [*isprint\(\)*](#), [*ispunct\(\)*](#), [*isspace\(\)*](#),
 38810 [*isupper\(\)*](#), [*isxdigit\(\)*](#)

38811 XBD Chapter 7 (on page 135), [*<ctype.h>*](#), [*<locale.h>*](#)

38812 **CHANGE HISTORY**

38813 First released in Issue 1. Derived from Issue 1 of the SVID.

38814 **Issue 6**

38815 The normative text is updated to avoid use of the term “must” for application requirements.

38816 **Issue 7**

38817 The *isdigit_l()* function is added from The Open Group Technical Standard, 2006, Extended API
38818 Set Part 4.

38819 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0286 [302], XSH/TC1-2008/0287 [283], +
38820 and XSH/TC1-2008/0288 [283] are applied.

38821 **NAME**

38822 isfinite — test for finite value

38823 **SYNOPSIS**

38824 #include <math.h>

38825 int isfinite(real-floating x);

38826 **DESCRIPTION**

38827 CX The functionality described on this reference page is aligned with the ISO C standard. Any
38828 conflict between the requirements described here and the ISO C standard is unintentional. This
38829 volume of POSIX.1-2008 defers to the ISO C standard.

38830 The *isfinite()* macro shall determine whether its argument has a finite value (zero, subnormal, or
38831 normal, and not infinite or NaN). First, an argument represented in a format wider than its
38832 semantic type is converted to its semantic type. Then determination is based on the type of the
38833 argument.

38834 **RETURN VALUE**38835 The *isfinite()* macro shall return a non-zero value if and only if its argument has a finite value.38836 **ERRORS**

38837 No errors are defined.

38838 **EXAMPLES**

38839 None.

38840 **APPLICATION USAGE**

38841 None.

38842 **RATIONALE**

38843 None.

38844 **FUTURE DIRECTIONS**

38845 None.

38846 **SEE ALSO**38847 *fpclassify()*, *isinf()*, *isnan()*, *isnormal()*, *signbit()*

38848 XBD <math.h>

38849 **CHANGE HISTORY**

38850 First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

38851 **NAME**

38852 isgraph, isgraph_l — test for a visible character

38853 **SYNOPSIS**

38854 #include <ctype.h>

38855 int isgraph(int c);

38856 CX int isgraph_l(int c, locale_t locale);

38857 **DESCRIPTION**

38858 CX For *isgraph()*: The functionality described on this reference page is aligned with the ISO C
 38859 standard. Any conflict between the requirements described here and the ISO C standard is
 38860 unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

38861 CX The *isgraph()* and *isgraph_l()* functions shall test whether *c* is a character of class **graph** in the
 38862 CX current locale, or in the locale represented by *locale*, respectively; see XBD Chapter 7 (on page
 38863 135).

38864 The *c* argument is an **int**, the value of which the application shall ensure is a character
 38865 representable as an **unsigned char** or equal to the value of the macro EOF. If the argument has
 38866 any other value, the behavior is undefined.

38867 CX The behavior is undefined if the *locale* argument to *isgraph_l()* is the special locale object +
 38868 LC_GLOBAL_LOCALE or is not a valid locale object handle.

38869 **RETURN VALUE**

38870 CX The *isgraph()* and *isgraph_l()* functions shall return non-zero if *c* is a character with a visible
 38871 representation; otherwise, they shall return 0.

38872 **ERRORS**

38873 No errors are defined.

38874 **EXAMPLES**

38875 None.

38876 **APPLICATION USAGE**

38877 To ensure applications portability, especially across natural languages, only these functions and
 38878 the functions in the reference pages listed in the SEE ALSO section should be used for character
 38879 classification.

38880 **RATIONALE**

38881 None.

38882 **FUTURE DIRECTIONS**

38883 None.

38884 **SEE ALSO**

38885 *isalnum()*, *isalpha()*, *isblank()*, *iscntrl()*, *isdigit()*, *islower()*, *isprint()*, *ispunct()*, *isspace()*, *isupper()*,
 38886 *isxdigit()*, *setlocale()*, *uselocale()*

38887 XBD Chapter 7 (on page 135), <ctype.h>, <locale.h>

38888 **CHANGE HISTORY**

38889 First released in Issue 1. Derived from Issue 1 of the SVID.

38890 **Issue 6**

38891 The normative text is updated to avoid use of the term “must” for application requirements.

38892 **Issue 7**38893
38894

The *isgraph_l()* function is added from The Open Group Technical Standard, 2006, Extended API Set Part 4.

38895
38896

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0289 [302], XSH/TC1-2008/0290 [283], + and XSH/TC1-2008/0291 [283] are applied.

38897 **NAME**38898 isgreater — test if *x* greater than *y*38899 **SYNOPSIS**

38900 #include <math.h>

38901 int isgreater(real-floating *x*, real-floating *y*);38902 **DESCRIPTION**

38903 **CX** The functionality described on this reference page is aligned with the ISO C standard. Any
38904 conflict between the requirements described here and the ISO C standard is unintentional. This
38905 volume of POSIX.1-2008 defers to the ISO C standard.

38906 The *isgreater()* macro shall determine whether its first argument is greater than its second
38907 argument. The value of *isgreater(x, y)* shall be equal to $(x) > (y)$; however, unlike $(x) > (y)$,
38908 *isgreater(x, y)* shall not raise the invalid floating-point exception when *x* and *y* are unordered.

38909 **RETURN VALUE**38910 Upon successful completion, the *isgreater()* macro shall return the value of $(x) > (y)$.38911 If *x* or *y* is NaN, 0 shall be returned.38912 **ERRORS**

38913 No errors are defined.

38914 **EXAMPLES**

38915 None.

38916 **APPLICATION USAGE**

38917 The relational and equality operators support the usual mathematical relationships between
38918 numeric values. For any ordered pair of numeric values, exactly one of the relationships (less,
38919 greater, and equal) is true. Relational operators may raise the invalid floating-point exception
38920 when argument values are NaNs. For a NaN and a numeric value, or for two NaNs, just the
38921 unordered relationship is true. This macro is a quiet (non-floating-point exception raising)
38922 version of a relational operator. It facilitates writing efficient code that accounts for NaNs
38923 without suffering the invalid floating-point exception. In the SYNOPSIS section, **real-floating**
38924 indicates that the argument shall be an expression of **real-floating** type.

38925 **RATIONALE**

38926 None.

38927 **FUTURE DIRECTIONS**

38928 None.

38929 **SEE ALSO**38930 *isgreaterequal()*, *isless()*, *islessequal()*, *islessgreater()*, *isunordered()*

38931 XBD <math.h>

38932 **CHANGE HISTORY**

38933 First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

38934 **NAME**38935 isgreaterequal — test if *x* is greater than or equal to *y*38936 **SYNOPSIS**

38937 #include <math.h>

38938 int isgreaterequal(real-floating *x*, real-floating *y*);38939 **DESCRIPTION**

38940 **CX** The functionality described on this reference page is aligned with the ISO C standard. Any
 38941 conflict between the requirements described here and the ISO C standard is unintentional. This
 38942 volume of POSIX.1-2008 defers to the ISO C standard.

38943 The *isgreaterequal()* macro shall determine whether its first argument is greater than or equal to
 38944 its second argument. The value of *isgreaterequal(x, y)* shall be equal to $(x) \geq (y)$; however, unlike
 38945 $(x) \geq (y)$, *isgreaterequal(x, y)* shall not raise the invalid floating-point exception when *x* and *y* are
 38946 unordered.

38947 **RETURN VALUE**38948 Upon successful completion, the *isgreaterequal()* macro shall return the value of $(x) \geq (y)$.38949 If *x* or *y* is NaN, 0 shall be returned.38950 **ERRORS**

38951 No errors are defined.

38952 **EXAMPLES**

38953 None.

38954 **APPLICATION USAGE**

38955 The relational and equality operators support the usual mathematical relationships between
 38956 numeric values. For any ordered pair of numeric values, exactly one of the relationships (less,
 38957 greater, and equal) is true. Relational operators may raise the invalid floating-point exception
 38958 when argument values are NaNs. For a NaN and a numeric value, or for two NaNs, just the
 38959 unordered relationship is true. This macro is a quiet (non-floating-point exception raising)
 38960 version of a relational operator. It facilitates writing efficient code that accounts for NaNs
 38961 without suffering the invalid floating-point exception. In the SYNOPSIS section, **real-floating**
 38962 indicates that the argument shall be an expression of **real-floating** type.

38963 **RATIONALE**

38964 None.

38965 **FUTURE DIRECTIONS**

38966 None.

38967 **SEE ALSO**38968 *isgreater()*, *isless()*, *islessequal()*, *islessgreater()*, *isunordered()*

38969 XBD <math.h>

38970 **CHANGE HISTORY**

38971 First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

38972 **NAME**

38973 isinf — test for infinity

38974 **SYNOPSIS**

38975 #include <math.h>

38976 int isinf(real-floating x);

38977 **DESCRIPTION**

38978 CX The functionality described on this reference page is aligned with the ISO C standard. Any
38979 conflict between the requirements described here and the ISO C standard is unintentional. This
38980 volume of POSIX.1-2008 defers to the ISO C standard.

38981 The *isinf()* macro shall determine whether its argument value is an infinity (positive or
38982 negative). First, an argument represented in a format wider than its semantic type is converted
38983 to its semantic type. Then determination is based on the type of the argument.

38984 **RETURN VALUE**38985 The *isinf()* macro shall return a non-zero value if and only if its argument has an infinite value.38986 **ERRORS**

38987 No errors are defined.

38988 **EXAMPLES**

38989 None.

38990 **APPLICATION USAGE**

38991 None.

38992 **RATIONALE**

38993 None.

38994 **FUTURE DIRECTIONS**

38995 None.

38996 **SEE ALSO**38997 *fpclassify()*, *isfinite()*, *isnan()*, *isnormal()*, *signbit()*

38998 XBD <math.h>

38999 **CHANGE HISTORY**

39000 First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

39001 **NAME**39002 isless — test if *x* is less than *y*39003 **SYNOPSIS**

39004 #include <math.h>

39005 int isless(real-floating *x*, real-floating *y*);39006 **DESCRIPTION**

39007 **CX** The functionality described on this reference page is aligned with the ISO C standard. Any
 39008 conflict between the requirements described here and the ISO C standard is unintentional. This
 39009 volume of POSIX.1-2008 defers to the ISO C standard.

39010 The *isless()* macro shall determine whether its first argument is less than its second argument.
 39011 The value of *isless(x, y)* shall be equal to $(x) < (y)$; however, unlike $(x) < (y)$, *isless(x, y)* shall not
 39012 raise the invalid floating-point exception when *x* and *y* are unordered.

39013 **RETURN VALUE**39014 Upon successful completion, the *isless()* macro shall return the value of $(x) < (y)$.39015 If *x* or *y* is NaN, 0 shall be returned.39016 **ERRORS**

39017 No errors are defined.

39018 **EXAMPLES**

39019 None.

39020 **APPLICATION USAGE**

39021 The relational and equality operators support the usual mathematical relationships between
 39022 numeric values. For any ordered pair of numeric values, exactly one of the relationships (less,
 39023 greater, and equal) is true. Relational operators may raise the invalid floating-point exception
 39024 when argument values are NaNs. For a NaN and a numeric value, or for two NaNs, just the
 39025 unordered relationship is true. This macro is a quiet (non-floating-point exception raising)
 39026 version of a relational operator. It facilitates writing efficient code that accounts for NaNs
 39027 without suffering the invalid floating-point exception. In the SYNOPSIS section, **real-floating**
 39028 indicates that the argument shall be an expression of **real-floating** type.

39029 **RATIONALE**

39030 None.

39031 **FUTURE DIRECTIONS**

39032 None.

39033 **SEE ALSO**39034 *isgreater()*, *isgreaterequal()*, *islessequal()*, *islessgreater()*, *isunordered()*

39035 XBD <math.h>

39036 **CHANGE HISTORY**

39037 First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

NAME

islessequal — test if *x* is less than or equal to *y*

SYNOPSIS

```
#include <math.h>
```

```
int islessequal(real-floating x, real-floating y);
```

DESCRIPTION

CX The functionality described on this reference page is aligned with the ISO C standard. Any conflict between the requirements described here and the ISO C standard is unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

The *islessequal()* macro shall determine whether its first argument is less than or equal to its second argument. The value of *islessequal*(*x*, *y*) shall be equal to (*x*) <= (*y*); however, unlike (*x*) <= (*y*), *islessequal*(*x*, *y*) shall not raise the invalid floating-point exception when *x* and *y* are unordered.

RETURN VALUE

Upon successful completion, the *islessequal()* macro shall return the value of (*x*) <= (*y*).

If *x* or *y* is NaN, 0 shall be returned.

ERRORS

No errors are defined.

EXAMPLES

None.

APPLICATION USAGE

The relational and equality operators support the usual mathematical relationships between numeric values. For any ordered pair of numeric values, exactly one of the relationships (less, greater, and equal) is true. Relational operators may raise the invalid floating-point exception when argument values are NaNs. For a NaN and a numeric value, or for two NaNs, just the unordered relationship is true. This macro is a quiet (non-floating-point exception raising) version of a relational operator. It facilitates writing efficient code that accounts for NaNs without suffering the invalid floating-point exception. In the SYNOPSIS section, **real-floating** indicates that the argument shall be an expression of **real-floating** type.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

isgreater(), *isgreaterequal()*, *isless()*, *islessgreater()*, *isunordered()*

XBD **<math.h>**

CHANGE HISTORY

First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

39076 **NAME**39077 `islessgreater` — test if *x* is less than or greater than *y*39078 **SYNOPSIS**39079 `#include <math.h>`39080 `int islessgreater(real-floating x, real-floating y);`39081 **DESCRIPTION**

39082 CX The functionality described on this reference page is aligned with the ISO C standard. Any
 39083 conflict between the requirements described here and the ISO C standard is unintentional. This
 39084 volume of POSIX.1-2008 defers to the ISO C standard.

39085 The `islessgreater()` macro shall determine whether its first argument is less than or greater than
 39086 its second argument. The `islessgreater(x, y)` macro is similar to `(x < (y) || (x) > (y))`; however,
 39087 `islessgreater(x, y)` shall not raise the invalid floating-point exception when *x* and *y* are unordered
 39088 (nor shall it evaluate *x* and *y* twice).

39089 **RETURN VALUE**

39090 Upon successful completion, the `islessgreater()` macro shall return the value of
 39091 `(x < (y) || (x) > (y))`.

39092 If *x* or *y* is NaN, 0 shall be returned.

39093 **ERRORS**

39094 No errors are defined.

39095 **EXAMPLES**

39096 None.

39097 **APPLICATION USAGE**

39098 The relational and equality operators support the usual mathematical relationships between
 39099 numeric values. For any ordered pair of numeric values, exactly one of the relationships (less,
 39100 greater, and equal) is true. Relational operators may raise the invalid floating-point exception
 39101 when argument values are NaNs. For a NaN and a numeric value, or for two NaNs, just the
 39102 unordered relationship is true. This macro is a quiet (non-floating-point exception raising)
 39103 version of a relational operator. It facilitates writing efficient code that accounts for NaNs
 39104 without suffering the invalid floating-point exception. In the SYNOPSIS section, **real-floating**
 39105 indicates that the argument shall be an expression of **real-floating** type.

39106 **RATIONALE**

39107 None.

39108 **FUTURE DIRECTIONS**

39109 None.

39110 **SEE ALSO**

39111 [*isgreater\(\)*](#), [*isgreaterequal\(\)*](#), [*isless\(\)*](#), [*islessequal\(\)*](#), [*isunordered\(\)*](#)

39112 XBD [`<math.h>`](#)

39113 **CHANGE HISTORY**

39114 First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

39115 **NAME**

39116 islower, islower_l — test for a lowercase letter

39117 **SYNOPSIS**

```
39118     #include <ctype.h>
39119
39119     int islower(int c);
39120 CX    int islower_l(int c, locale_t locale);
```

39121 **DESCRIPTION**

39122 CX For *islower()*: The functionality described on this reference page is aligned with the ISO C
 39123 standard. Any conflict between the requirements described here and the ISO C standard is
 39124 unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

39125 CX The *islower()* and *islower_l()* functions shall test whether *c* is a character of class **lower** in the
 39126 CX current locale, or in the locale represented by *locale*, respectively; see XBD Chapter 7 (on page
 39127 135).

39128 The *c* argument is an **int**, the value of which the application shall ensure is a character
 39129 representable as an **unsigned char** or equal to the value of the macro EOF. If the argument has
 39130 any other value, the behavior is undefined.

39131 CX The behavior is undefined if the *locale* argument to *islower_l()* is the special locale object
 39132 LC_GLOBAL_LOCALE or is not a valid locale object handle.

39133 **RETURN VALUE**

39134 CX The *islower()* and *islower_l()* functions shall return non-zero if *c* is a lowercase letter; otherwise,
 39135 they shall return 0.

39136 **ERRORS**

39137 No errors are defined.

39138 **EXAMPLES**39139 **Testing for a Lowercase Letter**

39140 Two examples follow, the first using *islower()*, the second using multiple concurrent locales and
 39141 *islower_l()*.

39142 The examples test whether the value is a lowercase letter, based on the current locale, then use it
 39143 as part of a key value.

```
39144     /* Example 1 -- using islower() */
39145     #include <ctype.h>
39146     #include <stdlib.h>
39147     #include <locale.h>
39148     ...
39149     char *keystr;
39150     int elementlen, len;
39151     unsigned char c;
39152     ...
39153     setlocale(LC_ALL, "");
39154     ...
39155     len = 0;
39156     while (len < elementlen) {
39157         c = (unsigned char) (rand() % 256);
39158         ...
```

```

39159         if (islower(c))
39160             keystr[len++] = c;
39161     }
39162     ...
39163     /* Example 2 -- using islower_l() */
39164     #include <ctype.h>
39165     #include <stdlib.h>
39166     #include <locale.h>
39167     ...
39168     char *keystr;
39169     int elementlen, len;
39170     unsigned char c;
39171     ...
39172     locale_t loc = newlocale (LC_ALL_MASK, "", (locale_t) 0);
39173     ...
39174     len = 0;
39175     while (len < elementlen) {
39176         c = (unsigned char) (rand() % 256);
39177         ...
39178         if (islower_l(c, loc))
39179             keystr[len++] = c;
39180     }
39181     ...

```

APPLICATION USAGE

To ensure applications portability, especially across natural languages, only these functions and the functions in the reference pages listed in the SEE ALSO section should be used for character classification.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

[*isalnum\(\)*](#), [*isalpha\(\)*](#), [*isblank\(\)*](#), [*iscntrl\(\)*](#), [*isdigit\(\)*](#), [*isgraph\(\)*](#), [*isprint\(\)*](#), [*ispunct\(\)*](#), [*isspace\(\)*](#), [*isupper\(\)*](#), [*isxdigit\(\)*](#), [*setlocale\(\)*](#), [*uselocale\(\)*](#)

XBD [Chapter 7](#) (on page 135), [`<ctype.h>`](#), [`<locale.h>`](#)

CHANGE HISTORY

First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6

The normative text is updated to avoid use of the term “must” for application requirements and an example is added.

Issue 7

The *islower_l()* function is added from The Open Group Technical Standard, 2006, Extended API Set Part 4.

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0292 [302], XSH/TC1-2008/0293 [283], XSH/TC1-2008/0294 [283], XSH/TC1-2008/0295 [302], and XSH/TC1-2008/0296 [304] are applied.

39205 **NAME**

39206 isnan — test for a NaN

39207 **SYNOPSIS**

39208 #include <math.h>

39209 int isnan(real-floating x);

39210 **DESCRIPTION**

39211 CX The functionality described on this reference page is aligned with the ISO C standard. Any
39212 conflict between the requirements described here and the ISO C standard is unintentional. This
39213 volume of POSIX.1-2008 defers to the ISO C standard.

39214 The *isnan()* macro shall determine whether its argument value is a NaN. First, an argument
39215 represented in a format wider than its semantic type is converted to its semantic type. Then
39216 determination is based on the type of the argument.

39217 **RETURN VALUE**39218 The *isnan()* macro shall return a non-zero value if and only if its argument has a NaN value.39219 **ERRORS**

39220 No errors are defined.

39221 **EXAMPLES**

39222 None.

39223 **APPLICATION USAGE**

39224 None.

39225 **RATIONALE**

39226 None.

39227 **FUTURE DIRECTIONS**

39228 None.

39229 **SEE ALSO**39230 *fpclassify()*, *isfinite()*, *isinf()*, *isnormal()*, *signbit()*

39231 XBD <math.h>

39232 **CHANGE HISTORY**

39233 First released in Issue 3.

39234 **Issue 5**

39235 The DESCRIPTION is updated to indicate the return value when NaN is not supported. This
39236 text was previously published in the APPLICATION USAGE section.

39237 **Issue 6**

39238 Re-written for alignment with the ISO/IEC 9899:1999 standard.

39239 NAME

39240 isnormal — test for a normal value

39241 SYNOPSIS

39242 #include <math.h>

39243 int isnormal(real-floating x);

39244 DESCRIPTION

39245 CX The functionality described on this reference page is aligned with the ISO C standard. Any
39246 conflict between the requirements described here and the ISO C standard is unintentional. This
39247 volume of POSIX.1-2008 defers to the ISO C standard.

39248 The *isnormal()* macro shall determine whether its argument value is normal (neither zero,
39249 subnormal, infinite, nor NaN). First, an argument represented in a format wider than its
39250 semantic type is converted to its semantic type. Then determination is based on the type of the
39251 argument.

39252 RETURN VALUE

39253 The *isnormal()* macro shall return a non-zero value if and only if its argument has a normal
39254 value.

39255 ERRORS

39256 No errors are defined.

39257 EXAMPLES

39258 None.

39259 APPLICATION USAGE

39260 None.

39261 RATIONALE

39262 None.

39263 FUTURE DIRECTIONS

39264 None.

39265 SEE ALSO

39266 [*fpclassify\(\)*](#), [*isfinite\(\)*](#), [*isinf\(\)*](#), [*isnan\(\)*](#), [*signbit\(\)*](#)

39267 XBD [**<math.h>**](#)

39268 CHANGE HISTORY

39269 First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

39270 **NAME**

39271 isprint, isprint_l — test for a printable character

39272 **SYNOPSIS**

39273 #include <ctype.h>

39274 int isprint(int c);

39275 CX int isprint_l(int c, locale_t locale);

39276 **DESCRIPTION**

39277 CX For *isprint()*: The functionality described on this reference page is aligned with the ISO C
 39278 standard. Any conflict between the requirements described here and the ISO C standard is
 39279 unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

39280 CX The *isprint()* and *isprint_l()* functions shall test whether *c* is a character of class **print** in the
 39281 CX current locale, or in the locale represented by *locale*, respectively; see XBD Chapter 7 (on page
 39282 135).

39283 The *c* argument is an **int**, the value of which the application shall ensure is a character
 39284 representable as an **unsigned char** or equal to the value of the macro EOF. If the argument has
 39285 any other value, the behavior is undefined.

39286 CX The behavior is undefined if the *locale* argument to *isprint_l()* is the special locale object +
 39287 LC_GLOBAL_LOCALE or is not a valid locale object handle.

39288 **RETURN VALUE**

39289 CX The *isprint()* and *isprint_l()* functions shall return non-zero if *c* is a printable character;
 39290 otherwise, they shall return 0.

39291 **ERRORS**

39292 No errors are defined.

39293 **EXAMPLES**

39294 None.

39295 **APPLICATION USAGE**

39296 To ensure applications portability, especially across natural languages, only these functions and
 39297 the functions in the reference pages listed in the SEE ALSO section should be used for character
 39298 classification.

39299 **RATIONALE**

39300 None.

39301 **FUTURE DIRECTIONS**

39302 None.

39303 **SEE ALSO**

39304 *isalnum()*, *isalpha()*, *isblank()*, *iscntrl()*, *isdigit()*, *isgraph()*, *islower()*, *ispunct()*, *isspace()*, *isupper()*,
 39305 *isxdigit()*, *setlocale()*, *uselocale()*

39306 XBD Chapter 7 (on page 135), <ctype.h>, <locale.h>

39307 **CHANGE HISTORY**

39308 First released in Issue 1. Derived from Issue 1 of the SVID.

39309 **Issue 6**

39310 The normative text is updated to avoid use of the term “must” for application requirements.

39311 **Issue 7**

39312 The *isprint_l()* function is added from The Open Group Technical Standard, 2006, Extended API
39313 Set Part 4.

39314 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0297 [302], XSH/TC1-2008/0298 [283], +
39315 and XSH/TC1-2008/0299 [283] are applied.

39316 **NAME**

39317 ispunct, ispunct_l — test for a punctuation character

39318 **SYNOPSIS**

39319 #include <ctype.h>

39320 int ispunct(int c);

39321 CX int ispunct_l(int c, locale_t locale);

39322 **DESCRIPTION**

39323 CX For *ispunct()*: The functionality described on this reference page is aligned with the ISO C
 39324 standard. Any conflict between the requirements described here and the ISO C standard is
 39325 unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

39326 CX The *ispunct()* and *ispunct_l()* functions shall test whether *c* is a character of class **punct** in the
 39327 CX current locale, or in the locale represented by *locale*, respectively; see XBD Chapter 7 (on page
 39328 135).

39329 The *c* argument is an **int**, the value of which the application shall ensure is a character
 39330 representable as an **unsigned char** or equal to the value of the macro EOF. If the argument has
 39331 any other value, the behavior is undefined.

39332 CX The behavior is undefined if the *locale* argument to *ispunct_l()* is the special locale object +
 39333 LC_GLOBAL_LOCALE or is not a valid locale object handle.

39334 **RETURN VALUE**

39335 CX The *ispunct()* and *ispunct_l()* functions shall return non-zero if *c* is a punctuation character;
 39336 otherwise, they shall return 0.

39337 **ERRORS**

39338 No errors are defined.

39339 **EXAMPLES**

39340 None.

39341 **APPLICATION USAGE**

39342 To ensure applications portability, especially across natural languages, only these functions and
 39343 the functions in the reference pages listed in the SEE ALSO section should be used for character
 39344 classification.

39345 **RATIONALE**

39346 None.

39347 **FUTURE DIRECTIONS**

39348 None.

39349 **SEE ALSO**

39350 *isalnum()*, *isalpha()*, *isblank()*, *iscntrl()*, *isdigit()*, *isgraph()*, *islower()*, *isprint()*, *isspace()*, *isupper()*,
 39351 *isxdigit()*, *setlocale()*, *uselocale()*

39352 XBD Chapter 7 (on page 135), <ctype.h>, <locale.h>

39353 **CHANGE HISTORY**

39354 First released in Issue 1. Derived from Issue 1 of the SVID.

39355 **Issue 6**

39356 The normative text is updated to avoid use of the term “must” for application requirements.

39357 **Issue 7**

39358 The *ispunct_l()* function is added from The Open Group Technical Standard, 2006, Extended API
39359 Set Part 4.

39360 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0300 [302], XSH/TC1-2008/0301 [283], +
39361 and XSH/TC1-2008/0302 [283] are applied.

39362 **NAME**

39363 isspace, isspace_l — test for a white-space character

39364 **SYNOPSIS**

```
39365       #include <ctype.h>
39366       int isspace(int c);
39367 CX     int isspace_l(int c, locale_t locale);
```

39368 **DESCRIPTION**

39369 CX For *isspace()*: The functionality described on this reference page is aligned with the ISO C
 39370 standard. Any conflict between the requirements described here and the ISO C standard is
 39371 unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

39372 CX The *isspace()* and *isspace_l()* functions shall test whether *c* is a character of class **space** in the
 39373 CX current locale, or in the locale represented by *locale*, respectively; see XBD Chapter 7 (on page
 39374 135).

39375 The *c* argument is an **int**, the value of which the application shall ensure is a character
 39376 representable as an **unsigned char** or equal to the value of the macro EOF. If the argument has
 39377 any other value, the behavior is undefined.

39378 CX The behavior is undefined if the *locale* argument to *isspace_l()* is the special locale object +
 39379 LC_GLOBAL_LOCALE or is not a valid locale object handle.

39380 **RETURN VALUE**

39381 CX The *isspace()* and *isspace_l()* functions shall return non-zero if *c* is a white-space character;
 39382 otherwise, they shall return 0.

39383 **ERRORS**

39384 No errors are defined.

39385 **EXAMPLES**

39386 None.

39387 **APPLICATION USAGE**

39388 To ensure applications portability, especially across natural languages, only these functions and
 39389 the functions in the reference pages listed in the SEE ALSO section should be used for character
 39390 classification.

39391 **RATIONALE**

39392 None.

39393 **FUTURE DIRECTIONS**

39394 None.

39395 **SEE ALSO**

39396 *isalnum()*, *isalpha()*, *iscntrl()*, *isdigit()*, *isgraph()*, *islower()*, *isprint()*, *ispunct()*, *isupper()*,
 39397 *isxdigit()*, *setlocale()*, *uselocale()*

39398 XBD Chapter 7 (on page 135), [<ctype.h>](#), [<locale.h>](#)

39399 **CHANGE HISTORY**

39400 First released in Issue 1. Derived from Issue 1 of the SVID.

39401 **Issue 6**

39402 The normative text is updated to avoid use of the term “must” for application requirements.

39403 **Issue 7**

39404 The *isspace_l()* function is added from The Open Group Technical Standard, 2006, Extended API
39405 Set Part 4.

39406 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0303 [302], XSH/TC1-2008/0304 [283], +
39407 and XSH/TC1-2008/0305 [283] are applied.

39408 **NAME**

39409 isunordered — test if arguments are unordered

39410 **SYNOPSIS**

39411 #include <math.h>

39412 int isunordered(real-floating *x*, real-floating *y*);39413 **DESCRIPTION**

39414 **CX** The functionality described on this reference page is aligned with the ISO C standard. Any
 39415 conflict between the requirements described here and the ISO C standard is unintentional. This
 39416 volume of POSIX.1-2008 defers to the ISO C standard.

39417 The *isunordered()* macro shall determine whether its arguments are unordered.39418 **RETURN VALUE**39419 Upon successful completion, the *isunordered()* macro shall return 1 if its arguments are
 39420 unordered, and 0 otherwise.39421 If *x* or *y* is NaN, 1 shall be returned.39422 **ERRORS**

39423 No errors are defined.

39424 **EXAMPLES**

39425 None.

39426 **APPLICATION USAGE**

39427 The relational and equality operators support the usual mathematical relationships between
 39428 numeric values. For any ordered pair of numeric values, exactly one of the relationships (less,
 39429 greater, and equal) is true. Relational operators may raise the invalid floating-point exception
 39430 when argument values are NaNs. For a NaN and a numeric value, or for two NaNs, just the
 39431 unordered relationship is true. This macro is a quiet (non-floating-point exception raising)
 39432 version of a relational operator. It facilitates writing efficient code that accounts for NaNs
 39433 without suffering the invalid floating-point exception. In the SYNOPSIS section, **real-floating**
 39434 indicates that the argument shall be an expression of **real-floating** type.

39435 **RATIONALE**

39436 None.

39437 **FUTURE DIRECTIONS**

39438 None.

39439 **SEE ALSO**39440 *isgreater()*, *isgreaterequal()*, *isless()*, *islessequal()*, *islessgreater()*

39441 XBD <math.h>

39442 **CHANGE HISTORY**

39443 First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

39444 IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/50 is applied, correcting the RETURN
 39445 VALUE section when *x* or *y* is NaN.

39446 **NAME**

39447 isupper, isupper_l — test for an uppercase letter

39448 **SYNOPSIS**

39449 #include <ctype.h>

39450 int isupper(int c);

39451 CX int isupper_l(int c, locale_t locale);

39452 **DESCRIPTION**

39453 CX For *isupper()*: The functionality described on this reference page is aligned with the ISO C
 39454 standard. Any conflict between the requirements described here and the ISO C standard is
 39455 unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

39456 CX The *isupper()* and *isupper_l()* functions shall test whether *c* is a character of class **upper** in the
 39457 CX current locale, or in the locale represented by *locale*, respectively; see XBD Chapter 7 (on page
 39458 135).

39459 The *c* argument is an **int**, the value of which the application shall ensure is a character
 39460 representable as an **unsigned char** or equal to the value of the macro EOF. If the argument has
 39461 any other value, the behavior is undefined.

39462 CX The behavior is undefined if the *locale* argument to *isupper_l()* is the special locale object +
 39463 LC_GLOBAL_LOCALE or is not a valid locale object handle.

39464 **RETURN VALUE**

39465 CX The *isupper()* and *isupper_l()* functions shall return non-zero if *c* is an uppercase letter;
 39466 otherwise, they shall return 0.

39467 **ERRORS**

39468 No errors are defined.

39469 **EXAMPLES**

39470 None.

39471 **APPLICATION USAGE**

39472 To ensure applications portability, especially across natural languages, only these functions and
 39473 the functions in the reference pages listed in the SEE ALSO section should be used for character
 39474 classification.

39475 **RATIONALE**

39476 None.

39477 **FUTURE DIRECTIONS**

39478 None.

39479 **SEE ALSO**

39480 *isalnum()*, *isalpha()*, *isblank()*, *iscntrl()*, *isdigit()*, *isgraph()*, *islower()*, *isprint()*, *ispunct()*, *isspace()*,
 39481 *isxdigit()*, *setlocale()*, *uselocale()*

39482 XBD Chapter 7 (on page 135), <ctype.h>, <locale.h>

39483 **CHANGE HISTORY**

39484 First released in Issue 1. Derived from Issue 1 of the SVID.

39485 **Issue 6**

39486 The normative text is updated to avoid use of the term “must” for application requirements.

39487 **Issue 7**

39488 The *isupper_l()* function is added from The Open Group Technical Standard, 2006, Extended API
39489 Set Part 4.

39490 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0306 [302], XSH/TC1-2008/0307 [283], +
39491 and XSH/TC1-2008/0308 [283] are applied.

39492 **NAME**

39493 iswalnum, iswalnum_l — test for an alphanumeric wide-character code

39494 **SYNOPSIS**

39495 #include <wctype.h>

39496 int iswalnum(wint_t wc);

39497 CX int iswalnum_l(wint_t wc, locale_t locale);

39498 **DESCRIPTION**39499 CX For *iswalnum()*: The functionality described on this reference page is aligned with the ISO C
39500 standard. Any conflict between the requirements described here and the ISO C standard is
39501 unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.39502 CX The *iswalnum()* and *iswalnum_l()* functions shall test whether *wc* is a wide-character code
39503 CX representing a character of class **alpha** or **digit** in the current locale, or in the locale represented
39504 by *locale*, respectively; see XBD Chapter 7 (on page 135).39505 The *wc* argument is a **wint_t**, the value of which the application shall ensure is a wide-character
39506 code corresponding to a valid character in the current locale, or equal to the value of the macro
39507 WEOF. If the argument has any other value, the behavior is undefined.39508 CX The behavior is undefined if the *locale* argument to *iswalnum_l()* is the special locale object +
39509 LC_GLOBAL_LOCALE or is not a valid locale object handle.39510 **RETURN VALUE**39511 CX The *iswalnum()* and *iswalnum_l()* functions shall return non-zero if *wc* is an alphanumeric
39512 wide-character code; otherwise, they shall return 0.39513 **ERRORS**

39514 No errors are defined.

39515 **EXAMPLES**

39516 None.

39517 **APPLICATION USAGE**39518 To ensure applications portability, especially across natural languages, only these functions and
39519 the functions in the reference pages listed in the SEE ALSO section should be used for character
39520 classification.39521 **RATIONALE**

39522 None.

39523 **FUTURE DIRECTIONS**

39524 None.

39525 **SEE ALSO**39526 *iswalpha()*, *iswcntrl()*, *iswctype()*, *iswdigit()*, *iswgraph()*, *iswlower()*, *iswprint()*, *iswpunct()*,
39527 *iswspace()*, *iswupper()*, *iswxdigit()*, *setlocale()*, *uselocale()*

39528 XBD Chapter 7 (on page 135), <locale.h>, <stdio.h>, <wctype.h>

39529 **CHANGE HISTORY**

39530 First released as a World-wide Portability Interface in Issue 4.

39531 **Issue 5**39532 The following change has been made in this version for alignment with
39533 ISO/IEC 9899:1990/Amendment 1:1995 (E):

- 39534 • The SYNOPSIS has been changed to indicate that this function and associated data types
39535 are now made visible by inclusion of the `<wctype.h>` header rather than `<wchar.h>`.
- 39536 **Issue 6**
- 39537 The normative text is updated to avoid use of the term “must” for application requirements.
- 39538 **Issue 7**
- 39539 The `iswalnum_l()` function is added from The Open Group Technical Standard, 2006, Extended
39540 API Set Part 4.
- 39541 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0309 [302], XSH/TC1-2008/0310 [283], +
39542 and XSH/TC1-2008/0311 [283] are applied.

39543 **NAME**

39544 iswalpha, iswalpha_l — test for an alphabetic wide-character code

39545 **SYNOPSIS**

39546 #include <wctype.h>

39547 int iswalpha(wint_t wc);

39548 CX int iswalpha_l(wint_t wc, locale_t locale);

39549 **DESCRIPTION**

39550 CX For *iswalpha()*: The functionality described on this reference page is aligned with the ISO C
 39551 standard. Any conflict between the requirements described here and the ISO C standard is
 39552 unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

39553 CX The *iswalpha()* and *iswalpha_l()* functions shall test whether *wc* is a wide-character code
 39554 CX representing a character of class **alpha** in the current locale, or in the locale represented by *locale*,
 39555 respectively; see XBD Chapter 7 (on page 135).

39556 The *wc* argument is a **wint_t**, the value of which the application shall ensure is a wide-character
 39557 code corresponding to a valid character in the current locale, or equal to the value of the macro
 39558 WEOF. If the argument has any other value, the behavior is undefined.

39559 CX The behavior is undefined if the *locale* argument to *iswalpha_l()* is the special locale object
 39560 LC_GLOBAL_LOCALE or is not a valid locale object handle.

39561 **RETURN VALUE**

39562 CX The *iswalpha()* and *iswalpha_l()* functions shall return non-zero if *wc* is an alphabetic wide-
 39563 character code; otherwise, they shall return 0.

39564 **ERRORS**

39565 No errors are defined.

39566 **EXAMPLES**

39567 None.

39568 **APPLICATION USAGE**

39569 To ensure applications portability, especially across natural languages, only these functions and
 39570 the functions in the reference pages listed in the SEE ALSO section should be used for character
 39571 classification.

39572 **RATIONALE**

39573 None.

39574 **FUTURE DIRECTIONS**

39575 None.

39576 **SEE ALSO**

39577 *iswalnum()*, *iswcntrl()*, *iswctype()*, *iswdigit()*, *iswgraph()*, *iswlower()*, *iswprint()*, *iswpunct()*,
 39578 *iswspace()*, *iswupper()*, *iswxdigit()*, *setlocale()*, *uselocale()*

39579 XBD Chapter 7 (on page 135), <locale.h>, <wctype.h>

39580 **CHANGE HISTORY**

39581 First released in Issue 4.

39582 **Issue 5**

39583 The following change has been made in this version for alignment with
 39584 ISO/IEC 9899:1990/Amendment 1:1995 (E):

- 39585 • The SYNOPSIS has been changed to indicate that this function and associated data types
39586 are now made visible by inclusion of the `<wctype.h>` header rather than `<wchar.h>`.
- 39587 **Issue 6**
- 39588 The normative text is updated to avoid use of the term “must” for application requirements.
- 39589 **Issue 7**
- 39590 The *iswalpha_l()* function is added from The Open Group Technical Standard, 2006, Extended
39591 API Set Part 4.
- 39592 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0312 [302], XSH/TC1-2008/0313 [283], +
39593 and XSH/TC1-2008/0314 [283] are applied.

39594 **NAME**

39595 iswblank, iswblank_l — test for a blank wide-character code

39596 **SYNOPSIS**

39597 #include <wctype.h>

39598 int iswblank(wint_t wc);

39599 CX int iswblank_l(wint_t wc, locale_t locale);

39600 **DESCRIPTION**

39601 CX For *iswblank()*: The functionality described on this reference page is aligned with the ISO C
 39602 standard. Any conflict between the requirements described here and the ISO C standard is
 39603 unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

39604 CX The *iswblank()* and *iswblank_l()* functions shall test whether *wc* is a wide-character code
 39605 CX representing a character of class **blank** in the current locale, or in the locale represented by
 39606 *locale*, respectively; see XBD Chapter 7 (on page 135).

39607 The *wc* argument is a **wint_t**, the value of which the application shall ensure is a wide-character
 39608 code corresponding to a valid character in the current locale, or equal to the value of the macro
 39609 WEOF. If the argument has any other value, the behavior is undefined.

39610 CX The behavior is undefined if the *locale* argument to *iswblank_l()* is the special locale object +
 39611 LC_GLOBAL_LOCALE or is not a valid locale object handle.

39612 **RETURN VALUE**

39613 CX The *iswblank()* and *iswblank_l()* functions shall return non-zero if *wc* is a blank wide-character
 39614 code; otherwise, they shall return 0.

39615 **ERRORS**

39616 No errors are defined.

39617 **EXAMPLES**

39618 None.

39619 **APPLICATION USAGE**

39620 To ensure applications portability, especially across natural languages, only these functions and
 39621 the functions in the reference pages listed in the SEE ALSO section should be used for character
 39622 classification.

39623 **RATIONALE**

39624 None.

39625 **FUTURE DIRECTIONS**

39626 None.

39627 **SEE ALSO**

39628 *iswalnum()*, *iswalpha()*, *iswcntrl()*, *iswctype()*, *iswdigit()*, *iswgraph()*, *iswlower()*, *iswprint()*,
 39629 *iswpunct()*, *iswspace()*, *iswupper()*, *iswxdigit()*, *setlocale()*, *uselocale()*

39630 XBD Chapter 7 (on page 135), <locale.h>, <wctype.h>

39631 **CHANGE HISTORY**

39632 First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

39633 **Issue 7**

39634 The *iswblank_l()* function is added from The Open Group Technical Standard, 2006, Extended
 39635 API Set Part 4.

39636
39637

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0315 [302], XSH/TC1-2008/0316 [283], +
and XSH/TC1-2008/0317 [283] are applied.

39638 **NAME**

39639 iswcntrl, iswcntrl_l — test for a control wide-character code

39640 **SYNOPSIS**

39641 #include <wctype.h>

39642 int iswcntrl(wint_t wc);

39643 CX int iswcntrl_l(wint_t wc, locale_t locale);

39644 **DESCRIPTION**

39645 CX For *iswcntrl()*: The functionality described on this reference page is aligned with the ISO C
 39646 standard. Any conflict between the requirements described here and the ISO C standard is
 39647 unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

39648 CX The *iswcntrl()* and *iswcntrl_l()* functions shall test whether *wc* is a wide-character code
 39649 CX representing a character of class **cntrl** in the current locale, or in the locale represented by *locale*,
 39650 respectively; see XBD Chapter 7 (on page 135).

39651 The *wc* argument is a **wint_t**, the value of which the application shall ensure is a wide-character
 39652 code corresponding to a valid character in the current locale, or equal to the value of the macro
 39653 WEOF. If the argument has any other value, the behavior is undefined.

39654 CX The behavior is undefined if the *locale* argument to *iswcntrl_l()* is the special locale object
 39655 LC_GLOBAL_LOCALE or is not a valid locale object handle.

39656 **RETURN VALUE**

39657 CX The *iswcntrl()* and *iswcntrl_l()* functions shall return non-zero if *wc* is a control wide-character
 39658 code; otherwise, they shall return 0.

39659 **ERRORS**

39660 No errors are defined.

39661 **EXAMPLES**

39662 None.

39663 **APPLICATION USAGE**

39664 To ensure applications portability, especially across natural languages, only these functions and
 39665 the functions in the reference pages listed in the SEE ALSO section should be used for character
 39666 classification.

39667 **RATIONALE**

39668 None.

39669 **FUTURE DIRECTIONS**

39670 None.

39671 **SEE ALSO**

39672 *iswalnum()*, *iswalpha()*, *iswctype()*, *iswdigit()*, *iswgraph()*, *iswlower()*, *iswprint()*, *iswpunct()*,
 39673 *iswspace()*, *iswupper()*, *iswxdigit()*, *setlocale()*, *uselocale()*

39674 XBD Chapter 7 (on page 135), <locale.h>, <wctype.h>

39675 **CHANGE HISTORY**

39676 First released in Issue 4.

39677 **Issue 5**

39678 The following change has been made in this version for alignment with
 39679 ISO/IEC 9899:1990/Amendment 1:1995 (E):

- 39680 • The SYNOPSIS has been changed to indicate that this function and associated data types
39681 are now made visible by inclusion of the `<wctype.h>` header rather than `<wchar.h>`.
- 39682 **Issue 6**
- 39683 The normative text is updated to avoid use of the term “must” for application requirements.
- 39684 **Issue 7**
- 39685 The `iswcntrl_l()` function is added from The Open Group Technical Standard, 2006, Extended
39686 API Set Part 4.
- 39687 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0318 [302], XSH/TC1-2008/0319 [283], +
39688 and XSH/TC1-2008/0320 [283] are applied.

39689 **NAME**

39690 iswctype, iswctype_l — test character for a specified class

39691 **SYNOPSIS**

```
39692     #include <wctype.h>
39693
39693     int iswctype(wint_t wc, wctype_t charclass);
39694 CX     int iswctype_l(wint_t wc, wctype_t charclass,
39695                     locale_t locale);
```

39696 **DESCRIPTION**

39697 CX For *iswctype()*: The functionality described on this reference page is aligned with the ISO C
 39698 standard. Any conflict between the requirements described here and the ISO C standard is
 39699 unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

39700 CX The *iswctype()* and *iswctype_l()* functions shall determine whether the wide-character code *wc*
 39701 CX has the character class *charclass*, returning true or false. The *iswctype()* and *iswctype_l()*
 39702 functions are defined on WEOF and wide-character codes corresponding to the valid character
 39703 CX encodings in the current locale, or in the locale represented by *locale*, respectively. If the *wc*
 39704 CX argument is not in the domain of the function, the result is undefined. If the value of *charclass* is
 39705 invalid (that is, not obtained by a call to *wctype()* or *charclass* is invalidated by a subsequent call
 39706 to *setlocale()* that has affected category *LC_CTYPE*) the result is unspecified.

39707 CX The behavior is undefined if the *locale* argument to *iswctype_l()* is the special locale object
 39708 LC_GLOBAL_LOCALE or is not a valid locale object handle.

39709 **RETURN VALUE**

39710 CX The *iswctype()* and *iswctype_l()* functions shall return non-zero (true) if and only if *wc* has the
 39711 CX property described by *charclass*. If *charclass* is 0, these functions shall return 0.

39712 **ERRORS**

39713 No errors are defined.

39714 **EXAMPLES**39715 **Testing for a Valid Character**

```
39716     #include <wctype.h>
39717     ...
39718     int yes_or_no;
39719     wint_t wc;
39720     wctype_t valid_class;
39721     ...
39722     if ((valid_class=wctype("vowel")) == (wctype_t)0)
39723         /* Invalid character class. */
39724     yes_or_no=iswctype(wc,valid_class);
```

39725 **APPLICATION USAGE**

39726 The twelve strings "alnum", "alpha", "blank", "cntrl", "digit", "graph", "lower",
 39727 "print", "punct", "space", "upper", and "xdigit" are reserved for the standard
 39728 character classes. In the table below, the functions in the left column are equivalent to the
 39729 functions in the right column.

39730 iswalnum(<i>wc</i>)	iswctype(<i>wc</i> , wctype("alnum"))
39731 iswalnum_l(<i>wc</i> , <i>locale</i>)	iswctype_l(<i>wc</i> , wctype("alnum"), <i>locale</i>)
39732 iswalpna(<i>wc</i>)	iswctype(<i>wc</i> , wctype("alpha"))
39733 iswalpna_l(<i>wc</i> , <i>locale</i>)	iswctype_l(<i>wc</i> , wctype("alpha"), <i>locale</i>)

```

39734      iswblank(wc)           iswctype(wc, wctype("blank"))
39735      iswblank_l(wc, locale) iswctype_l(wc, wctype("blank"), locale)
39736      iswcntrl(wc)           iswctype(wc, wctype("cntrl"))
39737      iswcntrl_l(wc, locale) iswctype_l(wc, wctype("cntrl"), locale)
39738      iswdigit(wc)           iswctype(wc, wctype("digit"))
39739      iswdigit_l(wc, locale) iswctype_l(wc, wctype("digit"), locale)
39740      iswgraph(wc)           iswctype(wc, wctype("graph"))
39741      iswgraph_l(wc, locale) iswctype_l(wc, wctype("graph"), locale)
39742      iswlower(wc)           iswctype(wc, wctype("lower"))
39743      iswlower_l(wc, locale) iswctype_l(wc, wctype("lower"), locale)
39744      iswprint(wc)           iswctype(wc, wctype("print"))
39745      iswprint_l(wc, locale) iswctype_l(wc, wctype("print"), locale)
39746      iswpunct(wc)           iswctype(wc, wctype("punct"))
39747      iswpunct_l(wc, locale) iswctype_l(wc, wctype("punct"), locale)
39748      iswspace(wc)           iswctype(wc, wctype("space"))
39749      iswspace_l(wc, locale) iswctype_l(wc, wctype("space"), locale)
39750      iswupper(wc)           iswctype(wc, wctype("upper"))
39751      iswupper_l(wc, locale) iswctype_l(wc, wctype("upper"), locale)
39752      iswxdigit(wc)          iswctype(wc, wctype("xdigit"))
39753      iswxdigit_l(wc, locale) iswctype_l(wc, wctype("xdigit"), locale)

```

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

iswalnum(), *iswalpha()*, *iswcntrl()*, *iswdigit()*, *iswgraph()*, *iswlower()*, *iswprint()*, *iswpunct()*, *iswspace()*, *iswupper()*, *iswxdigit()*, *setlocale()*, *uselocale()*, *wctype()*

XBD **<locale.h>**, **<wctype.h>**

CHANGE HISTORY

First released as World-wide Portability Interfaces in Issue 4.

Issue 5

The following change has been made in this version for alignment with ISO/IEC 9899:1990/Amendment 1:1995 (E):

- The SYNOPSIS has been changed to indicate that this function and associated data types are now made visible by inclusion of the **<wctype.h>** header rather than **<wchar.h>**.

Issue 6

The behavior of $n=0$ is now described.

An example is added.

A new function, *iswblank()*, is added to the list in the APPLICATION USAGE.

Issue 7

The *iswctype_l()* function is added from The Open Group Technical Standard, 2006, Extended API Set Part 4.

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0321 [283] and XSH/TC1-2008/0322 + [283] are applied.

39778 **NAME**

39779 iswdigit, iswdigit_l — test for a decimal digit wide-character code

39780 **SYNOPSIS**

39781 #include <wctype.h>

39782 int iswdigit(wint_t wc);

39783 CX int iswdigit_l(wint_t wc, locale_t locale);

39784 **DESCRIPTION**

39785 CX For *iswdigit()*: The functionality described on this reference page is aligned with the ISO C
 39786 standard. Any conflict between the requirements described here and the ISO C standard is
 39787 unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

39788 CX The *iswdigit()* and *iswdigit_l()* functions shall test whether *wc* is a wide-character code
 39789 CX representing a character of class **digit** in the current locale, or in the locale represented by *locale*,
 39790 respectively; see XBD Chapter 7 (on page 135).

39791 The *wc* argument is a **wint_t**, the value of which the application shall ensure is a wide-character
 39792 code corresponding to a valid character in the current locale, or equal to the value of the macro
 39793 WEOF. If the argument has any other value, the behavior is undefined.

39794 CX The behavior is undefined if the *locale* argument to *iswdigit_l()* is the special locale object
 39795 LC_GLOBAL_LOCALE or is not a valid locale object handle.

39796 **RETURN VALUE**

39797 CX The *iswdigit()* and *iswdigit_l()* functions shall return non-zero if *wc* is a decimal digit wide-
 39798 character code; otherwise, they shall return 0.

39799 **ERRORS**

39800 No errors are defined.

39801 **EXAMPLES**

39802 None.

39803 **APPLICATION USAGE**

39804 To ensure applications portability, especially across natural languages, only these functions and
 39805 the functions in the reference pages listed in the SEE ALSO section should be used for character
 39806 classification.

39807 **RATIONALE**

39808 None.

39809 **FUTURE DIRECTIONS**

39810 None.

39811 **SEE ALSO**

39812 *iswalnum()*, *iswalpha()*, *iswcntrl()*, *iswctype()*, *iswgraph()*, *iswlower()*, *iswprint()*, *iswpunct()*,
 39813 *iswspace()*, *iswupper()*, *iswxdigit()*, *setlocale()*, *uselocale()*

39814 XBD Chapter 7 (on page 135), <locale.h>, <wctype.h>

39815 **CHANGE HISTORY**

39816 First released in Issue 4.

39817 **Issue 5**

39818 The following change has been made in this version for alignment with
 39819 ISO/IEC 9899:1990/Amendment 1:1995 (E):

- 39820 • The SYNOPSIS has been changed to indicate that this function and associated data types
39821 are now made visible by inclusion of the `<wctype.h>` header rather than `<wchar.h>`.
- 39822 **Issue 6**
- 39823 The normative text is updated to avoid use of the term “must” for application requirements.
- 39824 **Issue 7**
- 39825 The `iswdigit_l()` function is added from The Open Group Technical Standard, 2006, Extended
39826 API Set Part 4.
- 39827 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0323 [302], XSH/TC1-2008/0324 [283], +
39828 and XSH/TC1-2008/0325 [283] are applied.

39829 **NAME**

39830 iswgraph, iswgraph_l — test for a visible wide-character code

39831 **SYNOPSIS**

39832 #include <wctype.h>

39833 int iswgraph(wint_t wc);

39834 CX int iswgraph_l(wint_t wc, locale_t locale);

39835 **DESCRIPTION**

39836 CX For *iswgraph()*: The functionality described on this reference page is aligned with the ISO C
 39837 standard. Any conflict between the requirements described here and the ISO C standard is
 39838 unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

39839 CX The *iswgraph()* and *iswgraph_l()* functions shall test whether *wc* is a wide-character code
 39840 CX representing a character of class **graph** in the current locale, or in the locale represented by
 39841 *locale*, respectively; see XBD [Chapter 7](#) (on page 135).

39842 The *wc* argument is a **wint_t**, the value of which the application shall ensure is a wide-character
 39843 code corresponding to a valid character in the current locale, or equal to the value of the macro
 39844 WEOF. If the argument has any other value, the behavior is undefined.

39845 CX The behavior is undefined if the *locale* argument to *iswgraph_l()* is the special locale object
 39846 LC_GLOBAL_LOCALE or is not a valid locale object handle.

39847 **RETURN VALUE**

39848 CX The *iswgraph()* and *iswgraph_l()* functions shall return non-zero if *wc* is a wide-character code
 39849 with a visible representation; otherwise, they shall return 0.

39850 **ERRORS**

39851 No errors are defined.

39852 **EXAMPLES**

39853 None.

39854 **APPLICATION USAGE**

39855 To ensure applications portability, especially across natural languages, only these functions and
 39856 the functions in the reference pages listed in the SEE ALSO section should be used for character
 39857 classification.

39858 **RATIONALE**

39859 None.

39860 **FUTURE DIRECTIONS**

39861 None.

39862 **SEE ALSO**

39863 [iswalnum\(\)](#), [iswalpha\(\)](#), [iswcntrl\(\)](#), [iswctype\(\)](#), [iswdigit\(\)](#), [iswlower\(\)](#), [iswprint\(\)](#), [iswpunct\(\)](#),
 39864 [iswspace\(\)](#), [iswupper\(\)](#), [iswxdigit\(\)](#), [setlocale\(\)](#), [uselocale\(\)](#)

39865 XBD [Chapter 7](#) (on page 135), [<locale.h>](#), [<wctype.h>](#)39866 **CHANGE HISTORY**

39867 First released in Issue 4.

39868 **Issue 5**

39869 The following change has been made in this version for alignment with
 39870 ISO/IEC 9899:1990/Amendment 1:1995 (E):

- 39871 • The SYNOPSIS has been changed to indicate that this function and associated data types
39872 are now made visible by inclusion of the `<wctype.h>` header rather than `<wchar.h>`.
- 39873 **Issue 6**
- 39874 The normative text is updated to avoid use of the term “must” for application requirements.
- 39875 **Issue 7**
- 39876 The `iswgraph_l()` function is added from The Open Group Technical Standard, 2006, Extended
39877 API Set Part 4.
- 39878 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0326 [302], XSH/TC1-2008/0327 [283], +
39879 and XSH/TC1-2008/0328 [283] are applied.

39880 **NAME**

39881 iswlower, iswlower_l — test for a lowercase letter wide-character code

39882 **SYNOPSIS**

39883 #include <wctype.h>

39884 int iswlower(wint_t wc);

39885 CX int iswlower_l(wint_t wc, locale_t locale);

39886 **DESCRIPTION**

39887 CX For *iswlower()*: The functionality described on this reference page is aligned with the ISO C
 39888 standard. Any conflict between the requirements described here and the ISO C standard is
 39889 unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

39890 CX The *iswlower()* and *iswlower_l()* functions shall test whether *wc* is a wide-character code
 39891 CX representing a character of class **lower** in the current locale, or in the locale represented by
 39892 *locale*, respectively; see XBD Chapter 7 (on page 135).

39893 The *wc* argument is a **wint_t**, the value of which the application shall ensure is a wide-character
 39894 code corresponding to a valid character in the current locale, or equal to the value of the macro
 39895 WEOF. If the argument has any other value, the behavior is undefined.

39896 CX The behavior is undefined if the *locale* argument to *iswlower_l()* is the special locale object +
 39897 LC_GLOBAL_LOCALE or is not a valid locale object handle.

39898 **RETURN VALUE**

39899 CX The *iswlower()* and *iswlower_l()* functions shall return non-zero if *wc* is a lowercase letter wide-
 39900 character code; otherwise, they shall return 0.

39901 **ERRORS**

39902 No errors are defined.

39903 **EXAMPLES**

39904 None.

39905 **APPLICATION USAGE**

39906 To ensure applications portability, especially across natural languages, only these functions and
 39907 the functions in the reference pages listed in the SEE ALSO section should be used for character
 39908 classification.

39909 **RATIONALE**

39910 None.

39911 **FUTURE DIRECTIONS**

39912 None.

39913 **SEE ALSO**

39914 *iswalnum()*, *iswalpha()*, *iswcntrl()*, *iswctype()*, *iswdigit()*, *iswgraph()*, *iswprint()*, *iswpunct()*,
 39915 *iswspace()*, *iswupper()*, *iswxdigit()*, *setlocale()*, *uselocale()* (on page 2182) 1

39916 XBD Chapter 7 (on page 135), <locale.h>, <wctype.h>

39917 **CHANGE HISTORY**

39918 First released in Issue 4.

39919 **Issue 5**

39920 The following change has been made in this version for alignment with
 39921 ISO/IEC 9899:1990/Amendment 1:1995 (E):

- 39922 • The SYNOPSIS has been changed to indicate that this function and associated data types
39923 are now made visible by inclusion of the `<wctype.h>` header rather than `<wchar.h>`.
- 39924 **Issue 6**
- 39925 The normative text is updated to avoid use of the term “must” for application requirements.
- 39926 **Issue 7**
- 39927 The `iswlower_l()` function is added from The Open Group Technical Standard, 2006, Extended
39928 API Set Part 4.
- 39929 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0329 [302], XSH/TC1-2008/0330 [283], +
39930 and XSH/TC1-2008/0331 [283] are applied.

NAME

iswprint, iswprint_l — test for a printable wide-character code

SYNOPSIS

```
#include <wctype.h>
```

```
int iswprint(wint_t wc);
```

```
CX int iswprint_l(wint_t wc, locale_t locale);
```

DESCRIPTION

CX For *iswprint()*: The functionality described on this reference page is aligned with the ISO C standard. Any conflict between the requirements described here and the ISO C standard is unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

CX The *iswprint()* and *iswprint_l()* functions shall test whether *wc* is a wide-character code representing a character of class **print** in the current locale, or in the locale represented by *locale*, respectively; see XBD Chapter 7 (on page 135).

The *wc* argument is a **wint_t**, the value of which the application shall ensure is a wide-character code corresponding to a valid character in the current locale, or equal to the value of the macro WEOF. If the argument has any other value, the behavior is undefined.

CX The behavior is undefined if the *locale* argument to *iswprint_l()* is the special locale object LC_GLOBAL_LOCALE or is not a valid locale object handle.

RETURN VALUE

CX The *iswprint()* and *iswprint_l()* functions shall return non-zero if *wc* is a printable wide-character code; otherwise, they shall return 0.

ERRORS

No errors are defined.

EXAMPLES

None.

APPLICATION USAGE

To ensure applications portability, especially across natural languages, only these functions and the functions in the reference pages listed in the SEE ALSO section should be used for character classification.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

iswalnum(), *iswalpha()*, *iswcntrl()*, *iswctype()*, *iswdigit()*, *iswgraph()*, *iswlower()*, *iswpunct()*, *iswspace()*, *iswupper()*, *iswxdigit()*, *setlocale()*, *uselocale()*

XBD Chapter 7 (on page 135), [<locale.h>](#), [<wctype.h>](#)

CHANGE HISTORY

First released in Issue 4.

Issue 5

The following change has been made in this version for alignment with ISO/IEC 9899:1990/Amendment 1:1995 (E):

- 39973 • The SYNOPSIS has been changed to indicate that this function and associated data types
39974 are now made visible by inclusion of the **<wctype.h>** header rather than **<wchar.h>**.
- 39975 **Issue 6**
39976 The normative text is updated to avoid use of the term “must” for application requirements.
- 39977 **Issue 7**
39978 The *iswprint_l()* function is added from The Open Group Technical Standard, 2006, Extended
39979 API Set Part 4.
- 39980 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0332 [302], XSH/TC1-2008/0333 [283], +
39981 and XSH/TC1-2008/0334 [283] are applied.

39982 **NAME**

39983 iswpunct, iswpunct_l — test for a punctuation wide-character code

39984 **SYNOPSIS**

```
39985     #include <wctype.h>
39986
39987     int iswpunct(wint_t wc);
39987 CX    int iswpunct_l(wint_t wc, locale_t locale);
```

39988 **DESCRIPTION**

39989 CX For *iswpunct()*: The functionality described on this reference page is aligned with the ISO C
 39990 standard. Any conflict between the requirements described here and the ISO C standard is
 39991 unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

39992 CX The *iswpunct()* and *iswpunct_l()* functions shall test whether *wc* is a wide-character code
 39993 CX representing a character of class **punct** in the current locale, or in the locale represented by
 39994 *locale*, respectively; see XBD Chapter 7 (on page 135).

39995 The *wc* argument is a **wint_t**, the value of which the application shall ensure is a wide-character
 39996 code corresponding to a valid character in the current locale, or equal to the value of the macro
 39997 WEOF. If the argument has any other value, the behavior is undefined.

39998 CX The behavior is undefined if the *locale* argument to *iswpunct_l()* is the special locale object +
 39999 LC_GLOBAL_LOCALE or is not a valid locale object handle.

40000 **RETURN VALUE**

40001 CX The *iswpunct()* and *iswpunct_l()* functions shall return non-zero if *wc* is a punctuation wide-
 40002 character code; otherwise, they shall return 0.

40003 **ERRORS**

40004 No errors are defined.

40005 **EXAMPLES**

40006 None.

40007 **APPLICATION USAGE**

40008 To ensure applications portability, especially across natural languages, only these functions and
 40009 the functions in the reference pages listed in the SEE ALSO section should be used for character
 40010 classification.

40011 **RATIONALE**

40012 None.

40013 **FUTURE DIRECTIONS**

40014 None.

40015 **SEE ALSO**

40016 *iswalnum()*, *iswalpha()*, *iswcntrl()*, *iswctype()*, *iswdigit()*, *iswgraph()*, *iswlower()*, *iswprint()*,
 40017 *iswspace()*, *iswupper()*, *iswxdigit()*, *setlocale()*, *uselocale()*

40018 XBD Chapter 7 (on page 135), [<locale.h>](#), [<wctype.h>](#)

40019 **CHANGE HISTORY**

40020 First released in Issue 4.

40021 **Issue 5**

40022 The following change has been made in this version for alignment with
 40023 ISO/IEC 9899:1990/Amendment 1:1995 (E):

- 40024 • The SYNOPSIS has been changed to indicate that this function and associated data types
40025 are now made visible by inclusion of the **<wctype.h>** header rather than **<wchar.h>**.
- 40026 **Issue 6**
- 40027 The normative text is updated to avoid use of the term “must” for application requirements.
- 40028 **Issue 7**
- 40029 The *iswpunct_l()* function is added from The Open Group Technical Standard, 2006, Extended
40030 API Set Part 4.
- 40031 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0335 [302], XSH/TC1-2008/0336 [283], +
40032 and XSH/TC1-2008/0337 [283] are applied.

40033 **NAME**

40034 iswspace, iswspace_l — test for a white-space wide-character code

40035 **SYNOPSIS**

40036 #include <wctype.h>

40037 int iswspace(wint_t wc);

40038 CX int iswspace_l(wint_t wc, locale_t locale);

40039 **DESCRIPTION**

40040 CX For *iswspace()*: The functionality described on this reference page is aligned with the ISO C
 40041 standard. Any conflict between the requirements described here and the ISO C standard is
 40042 unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

40043 CX The *iswspace()* and *iswspace_l()* functions shall test whether *wc* is a wide-character code
 40044 CX representing a character of class **space** in the current locale, or in the locale represented by *locale*,
 40045 respectively; see XBD Chapter 7 (on page 135).

40046 The *wc* argument is a **wint_t**, the value of which the application shall ensure is a wide-character
 40047 code corresponding to a valid character in the current locale, or equal to the value of the macro
 40048 WEOF. If the argument has any other value, the behavior is undefined.

40049 CX The behavior is undefined if the *locale* argument to *iswspace_l()* is the special locale object
 40050 LC_GLOBAL_LOCALE or is not a valid locale object handle.

40051 **RETURN VALUE**

40052 CX The *iswspace()* and *iswspace_l()* functions shall return non-zero if *wc* is a white-space wide-
 40053 character code; otherwise, they shall return 0.

40054 **ERRORS**

40055 No errors are defined.

40056 **EXAMPLES**

40057 None.

40058 **APPLICATION USAGE**

40059 To ensure applications portability, especially across natural languages, only these functions and
 40060 the functions in the reference pages listed in the SEE ALSO section should be used for character
 40061 classification.

40062 **RATIONALE**

40063 None.

40064 **FUTURE DIRECTIONS**

40065 None.

40066 **SEE ALSO**

40067 *iswalnum()*, *iswalpha()*, *iswcntrl()*, *iswctype()*, *iswdigit()*, *iswgraph()*, *iswlower()*, *iswprint()*,
 40068 *iswpunct()*, *iswupper()*, *iswxdigit()*, *setlocale()*, *uselocale()*

40069 XBD Chapter 7 (on page 135), <locale.h>, <wctype.h>

40070 **CHANGE HISTORY**

40071 First released in Issue 4.

40072 **Issue 5**

40073 The following change has been made in this version for alignment with
 40074 ISO/IEC 9899:1990/Amendment 1:1995 (E):

- 40075 • The SYNOPSIS has been changed to indicate that this function and associated data types
40076 are now made visible by inclusion of the **<wctype.h>** header rather than **<wchar.h>**.
- 40077 **Issue 6**
- 40078 The normative text is updated to avoid use of the term “must” for application requirements.
- 40079 **Issue 7**
- 40080 The *iswspace_l()* function is added from The Open Group Technical Standard, 2006, Extended
40081 API Set Part 4.
- 40082 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0338 [302], XSH/TC1-2008/0339 [283], +
40083 and XSH/TC1-2008/0340 [283] are applied.

40084 **NAME**

40085 iswupper, iswupper_l — test for an uppercase letter wide-character code

40086 **SYNOPSIS**

40087 #include <wctype.h>

40088 int iswupper(wint_t wc);

40089 CX int iswupper_l(wint_t wc, locale_t locale);

40090 **DESCRIPTION**

40091 CX For *iswupper()*: The functionality described on this reference page is aligned with the ISO C
 40092 standard. Any conflict between the requirements described here and the ISO C standard is
 40093 unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

40094 CX The *iswupper()* and *iswupper_l()* functions shall test whether *wc* is a wide-character code
 40095 CX representing a character of class **upper** in the current locale, or in the locale represented by
 40096 *locale*, respectively; see XBD Chapter 7 (on page 135).

40097 The *wc* argument is a **wint_t**, the value of which the application shall ensure is a wide-character
 40098 code corresponding to a valid character in the current locale, or equal to the value of the macro
 40099 WEOF. If the argument has any other value, the behavior is undefined.

40100 CX The behavior is undefined if the *locale* argument to *iswupper_l()* is the special locale object +
 40101 LC_GLOBAL_LOCALE or is not a valid locale object handle.

40102 **RETURN VALUE**

40103 CX The *iswupper()* and *iswupper_l()* functions shall return non-zero if *wc* is an uppercase letter
 40104 wide-character code; otherwise, they shall return 0.

40105 **ERRORS**

40106 No errors are defined.

40107 **EXAMPLES**

40108 None.

40109 **APPLICATION USAGE**

40110 To ensure applications portability, especially across natural languages, only these functions and
 40111 the functions in the reference pages listed in the SEE ALSO section should be used for character
 40112 classification.

40113 **RATIONALE**

40114 None.

40115 **FUTURE DIRECTIONS**

40116 None.

40117 **SEE ALSO**

40118 *iswalnum()*, *iswalpha()*, *iswcntrl()*, *iswctype()*, *iswdigit()*, *iswgraph()*, *iswlower()*, *iswprint()*,
 40119 *iswpunct()*, *iswspace()*, *iswxdigit()*, *setlocale()*, *uselocale()*

40120 XBD Chapter 7 (on page 135), <locale.h>, <wctype.h>

40121 **CHANGE HISTORY**

40122 First released in Issue 4.

40123 **Issue 5**

40124 The following change has been made in this version for alignment with
 40125 ISO/IEC 9899:1990/Amendment 1:1995 (E):

- 40126 • The SYNOPSIS has been changed to indicate that this function and associated data types
- 40127 are now made visible by inclusion of the **<wctype.h>** header rather than **<wchar.h>**.
- 40128 **Issue 6**
- 40129 The normative text is updated to avoid use of the term “must” for application requirements.
- 40130 **Issue 7**
- 40131 The *iswupper_l()* function is added from The Open Group Technical Standard, 2006, Extended
- 40132 API Set Part 4.
- 40133 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0341 [302], XSH/TC1-2008/0342 [283], +
- 40134 and XSH/TC1-2008/0343 [283] are applied.

40135 **NAME**

40136 iswxdigit, iswxdigit_l — test for a hexadecimal digit wide-character code

40137 **SYNOPSIS**

40138 #include <wctype.h>

40139 int iswxdigit(wint_t wc);

40140 CX int iswxdigit_l(wint_t wc, locale_t locale);

40141 **DESCRIPTION**

40142 CX For *iswxdigit()*: The functionality described on this reference page is aligned with the ISO C
 40143 standard. Any conflict between the requirements described here and the ISO C standard is
 40144 unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

40145 CX The *iswxdigit()* and *iswxdigit_l()* functions shall test whether *wc* is a wide-character code
 40146 CX representing a character of class **xdigit** in the current locale, or in the locale represented by
 40147 *locale*, respectively; see XBD Chapter 7 (on page 135).

40148 The *wc* argument is a **wint_t**, the value of which the application shall ensure is a wide-character
 40149 code corresponding to a valid character in the current locale, or equal to the value of the macro
 40150 WEOF. If the argument has any other value, the behavior is undefined.

40151 CX The behavior is undefined if the *locale* argument to *iswxdigit_l()* is the special locale object +
 40152 LC_GLOBAL_LOCALE or is not a valid locale object handle.

40153 **RETURN VALUE**

40154 CX The *iswxdigit()* and *iswxdigit_l()* functions shall return non-zero if *wc* is a hexadecimal digit
 40155 wide-character code; otherwise, they shall return 0.

40156 **ERRORS**

40157 No errors are defined.

40158 **EXAMPLES**

40159 None.

40160 **APPLICATION USAGE**

40161 To ensure applications portability, especially across natural languages, only these functions and
 40162 the functions in the reference pages listed in the SEE ALSO section should be used for character
 40163 classification.

40164 **RATIONALE**

40165 None.

40166 **FUTURE DIRECTIONS**

40167 None.

40168 **SEE ALSO**

40169 *iswalnum()*, *iswalpha()*, *iswcntrl()*, *iswctype()*, *iswdigit()*, *iswgraph()*, *iswlower()*, *iswprint()*,
 40170 *iswpunct()*, *iswspace()*, *iswupper()*, *setlocale()*, *uselocale()*

40171 XBD Chapter 7 (on page 135), <locale.h>, <wctype.h>

40172 **CHANGE HISTORY**

40173 First released in Issue 4.

40174 **Issue 5**

40175 The following change has been made in this version for alignment with
 40176 ISO/IEC 9899:1990/Amendment 1:1995 (E):

- 40177 • The SYNOPSIS has been changed to indicate that this function and associated data types
40178 are now made visible by inclusion of the **<wctype.h>** header rather than **<wchar.h>**.
- 40179 **Issue 6**
- 40180 The normative text is updated to avoid use of the term “must” for application requirements.
- 40181 **Issue 7**
- 40182 The *iswxdigit_l()* function is added from The Open Group Technical Standard, 2006, Extended
40183 API Set Part 4.
- 40184 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0344 [302], XSH/TC1-2008/0345 [283], +
40185 and XSH/TC1-2008/0346 [283] are applied.

40186 **NAME**

40187 isxdigit, isxdigit_l — test for a hexadecimal digit

40188 **SYNOPSIS**

```
40189     #include <ctype.h>
40190     int isxdigit(int c);
40191 CX    int isxdigit_l(int c, locale_t locale);
```

40192 **DESCRIPTION**

40193 CX For *isxdigit()*: The functionality described on this reference page is aligned with the ISO C
 40194 standard. Any conflict between the requirements described here and the ISO C standard is
 40195 unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

40196 CX The *isxdigit()* and *isxdigit_l()* functions shall test whether *c* is a character of class **xdigit** in the
 40197 CX current locale, or in the locale represented by *locale*, respectively; see XBD Chapter 7 (on page
 40198 135).

40199 The *c* argument is an **int**, the value of which the application shall ensure is a character
 40200 representable as an **unsigned char** or equal to the value of the macro EOF. If the argument has
 40201 any other value, the behavior is undefined.

40202 CX The behavior is undefined if the *locale* argument to *isxdigit_l()* is the special locale object +
 40203 LC_GLOBAL_LOCALE or is not a valid locale object handle.

40204 **RETURN VALUE**

40205 CX The *isxdigit()* and *isxdigit_l()* functions shall return non-zero if *c* is a hexadecimal digit;
 40206 otherwise, they shall return 0.

40207 **ERRORS**

40208 No errors are defined.

40209 **EXAMPLES**

40210 None.

40211 **APPLICATION USAGE**

40212 To ensure applications portability, especially across natural languages, only these functions and
 40213 the functions in the reference pages listed in the SEE ALSO section should be used for character
 40214 classification.

40215 **RATIONALE**

40216 None.

40217 **FUTURE DIRECTIONS**

40218 None.

40219 **SEE ALSO**

40220 *isalnum()*, *isalpha()*, *isblank()*, *iscntrl()*, *isdigit()*, *isgraph()*, *islower()*, *isprint()*, *ispunct()*, *isspace()*,
 40221 *isupper()*

40222 XBD Chapter 7 (on page 135), [<ctype.h>](#)40223 **CHANGE HISTORY**

40224 First released in Issue 1. Derived from Issue 1 of the SVID.

40225 **Issue 6**

40226 The normative text is updated to avoid use of the term “must” for application requirements.

40227 **Issue 7**

40228 The *isxdigit_l()* function is added from The Open Group Technical Standard, 2006, Extended API
40229 Set Part 4.

40230 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0347 [302], XSH/TC1-2008/0348 [283], +
40231 and XSH/TC1-2008/0349 [283] are applied.

40232 **NAME**

40233 j0, j1, jn — Bessel functions of the first kind

40234 **SYNOPSIS**

```

40235 XSI      #include <math.h>
40236
40236         double j0(double x);
40237         double j1(double x);
40238         double jn(int n, double x);

```

40239 **DESCRIPTION**

40240 The *j0()*, *j1()*, and *jn()* functions shall compute Bessel functions of *x* of the first kind of orders 0, 1, and *n*, respectively.

40242 An application wishing to check for error situations should set *errno* to zero and call *feclearexcept*(FE_ALL_EXCEPT) before calling these functions. On return, if *errno* is non-zero or *fetestexcept*(FE_INVALID | FE_DIVBYZERO | FE_OVERFLOW | FE_UNDERFLOW) is non-zero, an error has occurred.

40246 **RETURN VALUE**

40247 Upon successful completion, these functions shall return the relevant Bessel value of *x* of the first kind.

40249 If the *x* argument is too large in magnitude, or the correct result would cause underflow, 0 shall be returned and a range error may occur.

40251 MXX If *x* is NaN, a NaN shall be returned.

40252 **ERRORS**

40253 These functions may fail if:

40254 Range Error The value of *x* was too large in magnitude, or an underflow occurred.

40255 If the integer expression (*math_errhandling* & MATH_ERRNO) is non-zero, then *errno* shall be set to [ERANGE]. If the integer expression (*math_errhandling* & MATH_ERREXCEPT) is non-zero, then the underflow floating-point exception shall be raised.

40259 No other errors shall occur.

40260 **EXAMPLES**

40261 None.

40262 **APPLICATION USAGE**

40263 On error, the expressions (*math_errhandling* & MATH_ERRNO) and (*math_errhandling* & MATH_ERREXCEPT) are independent of each other, but at least one of them must be non-zero.

40265 **RATIONALE**

40266 None.

40267 **FUTURE DIRECTIONS**

40268 None.

40269 **SEE ALSO**

40270 *feclearexcept()*, *fetestexcept()*, *isnan()*, *y0()*

40271 XBD Section 4.19 (on page 116), *<math.h>*

CHANGE HISTORY

40272 First released in Issue 1. Derived from Issue 1 of the SVID.
40273

Issue 5

40274 The DESCRIPTION is updated to indicate how an application should check for an error. This
40275 text was previously published in the APPLICATION USAGE section.
40276

Issue 6

40277 The may fail [EDOM] error is removed for the case for NaN.
40278

40279 The RETURN VALUE and ERRORS sections are reworked for alignment of the error handling
40280 with the ISO/IEC 9899:1999 standard.

Issue 7

40281 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0350 [68] is applied.
40282

+

40283 NAME

40284 jrand48 — generate a uniformly distributed pseudo-random long signed integer

40285 SYNOPSIS

```
40286 XSI      #include <stdlib.h>  
40287          long jrand48(unsigned short xsubi[3]);
```

40288 DESCRIPTION

40289 Refer to *drand48()*.

NAME

kill — send a signal to a process or a group of processes

SYNOPSIS

```
CX      #include <signal.h>
      int kill(pid_t pid, int sig);
```

DESCRIPTION

The *kill()* function shall send a signal to a process or a group of processes specified by *pid*. The signal to be sent is specified by *sig* and is either one from the list given in **<signal.h>** or 0. If *sig* is 0 (the null signal), error checking is performed but no signal is actually sent. The null signal can be used to check the validity of *pid*.

For a process to have permission to send a signal to a process designated by *pid*, unless the sending process has appropriate privileges, the real or effective user ID of the sending process shall match the real or saved set-user-ID of the receiving process.

If *pid* is greater than 0, *sig* shall be sent to the process whose process ID is equal to *pid*.

If *pid* is 0, *sig* shall be sent to all processes (excluding an unspecified set of system processes) whose process group ID is equal to the process group ID of the sender, and for which the process has permission to send a signal.

If *pid* is -1, *sig* shall be sent to all processes (excluding an unspecified set of system processes) for which the process has permission to send that signal.

If *pid* is negative, but not -1, *sig* shall be sent to all processes (excluding an unspecified set of system processes) whose process group ID is equal to the absolute value of *pid*, and for which the process has permission to send a signal.

If the value of *pid* causes *sig* to be generated for the sending process, and if *sig* is not blocked for the calling thread and if no other thread has *sig* unblocked or is waiting in a *sigwait()* function for *sig*, either *sig* or at least one pending unblocked signal shall be delivered to the sending thread before *kill()* returns.

The user ID tests described above shall not be applied when sending SIGCONT to a process that is a member of the same session as the sending process.

An implementation that provides extended security controls may impose further implementation-defined restrictions on the sending of signals, including the null signal. In particular, the system may deny the existence of some or all of the processes specified by *pid*.

The *kill()* function is successful if the process has permission to send *sig* to any of the processes specified by *pid*. If *kill()* fails, no signal shall be sent.

RETURN VALUE

Upon successful completion, 0 shall be returned. Otherwise, -1 shall be returned and *errno* set to indicate the error.

ERRORS

The *kill()* function shall fail if:

[EINVAL]	The value of the <i>sig</i> argument is an invalid or unsupported signal number.
[EPERM]	The process does not have permission to send the signal to any receiving process.

40331 [ESRCH] No process or process group can be found corresponding to that specified by
40332 *pid*.

40333 EXAMPLES

40334 None.

40335 APPLICATION USAGE

40336 None.

40337 RATIONALE

40338 The semantics for permission checking for *kill()* differed between System V and most other
40339 implementations, such as Version 7 or 4.3 BSD. The semantics chosen for this volume of
40340 POSIX.1-2008 agree with System V. Specifically, a set-user-ID process cannot protect itself
40341 against signals (or at least not against SIGKILL) unless it changes its real user ID. This choice
40342 allows the user who starts an application to send it signals even if it changes its effective user ID.
40343 The other semantics give more power to an application that wants to protect itself from the user
40344 who ran it.

40345 Some implementations provide semantic extensions to the *kill()* function when the absolute
40346 value of *pid* is greater than some maximum, or otherwise special, value. Negative values are a
40347 flag to *kill()*. Since most implementations return [ESRCH] in this case, this behavior is not
40348 included in this volume of POSIX.1-2008, although a conforming implementation could provide
40349 such an extension.

40350 The unspecified processes to which a signal cannot be sent may include the scheduler or *init*.

40351 There was initially strong sentiment to specify that, if *pid* specifies that a signal be sent to the
40352 calling process and that signal is not blocked, that signal would be delivered before *kill()*
40353 returns. This would permit a process to call *kill()* and be guaranteed that the call never return.
40354 However, historical implementations that provide only the *signal()* function make only the
40355 weaker guarantee in this volume of POSIX.1-2008, because they only deliver one signal each
40356 time a process enters the kernel. Modifications to such implementations to support the
40357 *sigaction()* function generally require entry to the kernel following return from a signal-catching
40358 function, in order to restore the signal mask. Such modifications have the effect of satisfying the
40359 stronger requirement, at least when *sigaction()* is used, but not necessarily when *signal()* is used.
40360 The standard developers considered making the stronger requirement except when *signal()* is
40361 used, but felt this would be unnecessarily complex. Implementors are encouraged to meet the
40362 stronger requirement whenever possible. In practice, the weaker requirement is the same, except
40363 in the rare case when two signals arrive during a very short window. This reasoning also applies
40364 to a similar requirement for *sigprocmask()*.

40365 In 4.2 BSD, the SIGCONT signal can be sent to any descendant process regardless of user-ID
40366 security checks. This allows a job control shell to continue a job even if processes in the job have
40367 altered their user IDs (as in the *su* command). In keeping with the addition of the concept of
40368 sessions, similar functionality is provided by allowing the SIGCONT signal to be sent to any
40369 process in the same session regardless of user ID security checks. This is less restrictive than BSD
40370 in the sense that ancestor processes (in the same session) can now be the recipient. It is more
40371 restrictive than BSD in the sense that descendant processes that form new sessions are now
40372 subject to the user ID checks. A similar relaxation of security is not necessary for the other job
40373 control signals since those signals are typically sent by the terminal driver in recognition of
40374 special characters being typed; the terminal driver bypasses all security checks.

40375 In secure implementations, a process may be restricted from sending a signal to a process having
40376 a different security label. In order to prevent the existence or nonexistence of a process from
40377 being used as a covert channel, such processes should appear nonexistent to the sender; that is,
40378 [ESRCH] should be returned, rather than [EPERM], if *pid* refers only to such processes.

Existing implementations vary on the result of a *kill()* with *pid* indicating an inactive process (a terminated process that has not been waited for by its parent). Some indicate success on such a call (subject to permission checking), while others give an error of [ESRCH]. Since the definition of process lifetime in this volume of POSIX.1-2008 covers inactive processes, the [ESRCH] error as described is inappropriate in this case. In particular, this means that an application cannot have a parent process check for termination of a particular child with *kill()*. (Usually this is done with the null signal; this can be done reliably with *waitpid()*.)

There is some belief that the name *kill()* is misleading, since the function is not always intended to cause process termination. However, the name is common to all historical implementations, and any change would be in conflict with the goal of minimal changes to existing application code.

FUTURE DIRECTIONS

None.

SEE ALSO

getpid(), *raise()*, *setsid()*, *sigaction()*, *sigqueue()*, *wait()*

XBD [*<signal.h>*](#), [*<sys/types.h>*](#)

CHANGE HISTORY

First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 5

The DESCRIPTION is updated for alignment with the POSIX Threads Extension.

Issue 6

In the SYNOPSIS, the optional include of the [*<sys/types.h>*](#) header is removed.

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- In the DESCRIPTION, the second paragraph is reworded to indicate that the saved set-user-ID of the calling process is checked in place of its effective user ID. This is a FIPS requirement.
- The requirement to include [*<sys/types.h>*](#) has been removed. Although [*<sys/types.h>*](#) was required for conforming implementations of previous POSIX specifications, it was not required for UNIX applications.
- The behavior when *pid* is *-1* is now specified. It was previously explicitly unspecified in the POSIX.1-1988 standard.

The normative text is updated to avoid use of the term “must” for application requirements.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/51 is applied, correcting the RATIONALE section.

NAME

killpg — send a signal to a process group

SYNOPSIS

```
XSI      #include <signal.h>
int killpg(pid_t pgrp, int sig);
```

DESCRIPTION

The *killpg()* function shall send the signal specified by *sig* to the process group specified by *pgrp*.

If *pgrp* is greater than 1, *killpg(pgrp, sig)* shall be equivalent to *kill(-pgrp, sig)*. If *pgrp* is less than or equal to 1, the behavior of *killpg()* is undefined.

RETURN VALUE

Refer to *kill()*.

ERRORS

Refer to *kill()*.

EXAMPLES**Sending a Signal to All Other Members of a Process Group**

The following example shows how the calling process could send a signal to all other members of its process group. To prevent itself from receiving the signal it first makes itself immune to the signal by ignoring it.

```
#include <signal.h>
#include <unistd.h>
...
    if (signal(SIGUSR1, SIG_IGN) == SIG_ERR)
        /* Handle error */;

    if (killpg(getpgrp(), SIGUSR1) == -1)
        /* Handle error */;
```

APPLICATION USAGE

None.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

getpgid(), *getpid()*, *kill()*, *raise()*

XBD *<signal.h>*

CHANGE HISTORY

First released in Issue 4, Version 2.

Issue 5

Moved from X/OPEN UNIX extension to BASE.

40452 **Issue 6**

40453

40454

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/52 is applied, adding the example to the EXAMPLES section.

40455 **NAME**

40456 l64a — convert a 32-bit integer to a radix-64 ASCII string

40457 **SYNOPSIS**

```
40458 XSI      #include <stdlib.h>
40459          char *l64a(long value);
```

40460 **DESCRIPTION**40461 Refer to *a64l()*.

40462 **NAME**

40463 labs, llabs — return a long integer absolute value

40464 **SYNOPSIS**

40465 #include <stdlib.h>

40466 long labs(long i);

40467 long long llabs(long long i);

40468 **DESCRIPTION**

40469 CX The functionality described on this reference page is aligned with the ISO C standard. Any
40470 conflict between the requirements described here and the ISO C standard is unintentional. This
40471 volume of POSIX.1-2008 defers to the ISO C standard.

40472 The *labs()* function shall compute the absolute value of the **long** integer operand *i*. The *llabs()*
40473 function shall compute the absolute value of the **long long** integer operand *i*. If the result
40474 cannot be represented, the behavior is undefined.

40475 **RETURN VALUE**40476 The *labs()* function shall return the absolute value of the **long** integer operand.40477 The *llabs()* function shall return the absolute value of the **long long** integer operand.40478 **ERRORS**

40479 No errors are defined.

40480 **EXAMPLES**

40481 None.

40482 **APPLICATION USAGE**

40483 None.

40484 **RATIONALE**

40485 None.

40486 **FUTURE DIRECTIONS**

40487 None.

40488 **SEE ALSO**40489 [abs\(\)](#)40490 XBD [<stdlib.h>](#)40491 **CHANGE HISTORY**

40492 First released in Issue 4. Derived from the ISO C standard.

40493 **Issue 6**40494 The *llabs()* function is added for alignment with the ISO/IEC 9899:1999 standard.40495 **Issue 7**

40496 SD5-XSH-ERN-152 is applied, correcting the RETURN VALUE section.

40497 **NAME**

40498 lchown — change the owner and group of a symbolic link

40499 **SYNOPSIS**

40500 #include <unistd.h>

40501 int lchown(const char *path, uid_t owner, gid_t group);

40502 **DESCRIPTION**

40503 The *lchown()* function shall be equivalent to *chown()*, except in the case where the named file is a
 40504 symbolic link. In this case, *lchown()* shall change the ownership of the symbolic link file itself,
 40505 while *chown()* changes the ownership of the file or directory to which the symbolic link refers.

40506 **RETURN VALUE**

40507 Upon successful completion, *lchown()* shall return 0. Otherwise, it shall return -1 and set *errno* to
 40508 indicate an error.

40509 **ERRORS**40510 The *lchown()* function shall fail if:

40511 [EACCES] Search permission is denied on a component of the path prefix of *path*.
 40512 [EINVAL] The owner or group ID is not a value supported by the implementation.
 40513 [ELOOP] A loop exists in symbolic links encountered during resolution of the *path*
 40514 argument.

40515 [ENAMETOOLONG]

40516 The length of a component of a pathname is longer than {NAME_MAX}.

40517 [ENOENT] A component of *path* does not name an existing file or *path* is an empty string.

40518 [ENOTDIR] A component of the path prefix names an existing file that is neither a
 40519 directory nor a symbolic link to a directory, or the *path* argument contains at
 40520 least one non-*<slash>* character and ends with one or more trailing *<slash>*
 40521 characters and the last pathname component names an existing file that is
 40522 neither a directory nor a symbolic link to a directory.

40523 [EPERM] The effective user ID does not match the owner of the file and the process does
 40524 not have appropriate privileges.

40525 [EROFS] The file resides on a read-only file system.

40526 The *lchown()* function may fail if:

40527 [EIO] An I/O error occurred while reading or writing to the file system.

40528 [EINTR] A signal was caught during execution of the function.

40529 [ELOOP] More than {SYMLOOP_MAX} symbolic links were encountered during
 40530 resolution of the *path* argument.

40531 [ENAMETOOLONG]

40532 The length of a pathname exceeds {PATH_MAX}, or pathname resolution of a
 40533 symbolic link produced an intermediate result with a length that exceeds
 40534 {PATH_MAX}.

40535 **EXAMPLES**40536 **Changing the Current Owner of a File**

40537 The following example shows how to change the ownership of the symbolic link named
 40538 **/modules/pass1** to the user ID associated with “jones” and the group ID associated with “cnd”.

40539 The numeric value for the user ID is obtained by using the *getpwnam()* function. The numeric
 40540 value for the group ID is obtained by using the *getgrnam()* function.

```
40541 #include <sys/types.h>
40542 #include <unistd.h>
40543 #include <pwd.h>
40544 #include <grp.h>

40545 struct passwd *pwd;
40546 struct group *grp;
40547 char          *path = "/modules/pass1";
40548 ...
40549 pwd = getpwnam("jones");
40550 grp = getgrnam("cnd");
40551 lchown(path, pwd->pw_uid, grp->gr_gid);
```

40552 **APPLICATION USAGE**

40553 On implementations which support symbolic links as directory entries rather than files, *lchown()*
 40554 may fail.

40555 **RATIONALE**

40556 None.

40557 **FUTURE DIRECTIONS**

40558 None.

40559 **SEE ALSO**

40560 *chown()*, *symlink()*

40561 XBD **<unistd.h>**

40562 **CHANGE HISTORY**

40563 First released in Issue 4, Version 2.

40564 **Issue 5**

40565 Moved from X/OPEN UNIX extension to BASE.

40566 **Issue 6**

40567 The wording of the mandatory [ELOOP] error condition is updated, and a second optional
 40568 [ELOOP] error condition is added.

40569 The Open Group Base Resolution bwg2001-013 is applied, adding wording to the
 40570 APPLICATION USAGE.

40571 **Issue 7**

40572 Austin Group Interpretation 1003.1-2001 #143 is applied.

40573 The *lchown()* function is moved from the XSI option to the Base.

40574 The [EOPNOTSUPP] error is removed.

40575 The [ENOTDIR] error condition is clarified to cover the condition where the last component of a
 40576 pathname exists but is not a directory or a symbolic link to a directory.

40577

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0351 [324] is applied.

+

40578 **NAME**

40579 lcong48 — seed a uniformly distributed pseudo-random signed long integer generator

40580 **SYNOPSIS**

```
40581 XSI      #include <stdlib.h>  
40582          void lcong48(unsigned short param[7]);
```

40583 **DESCRIPTION**40584 Refer to *drand48()*.

NAME

ldexp, ldexpf, ldexpl — load exponent of a floating-point number

SYNOPSIS

```
#include <math.h>

double ldexp(double x, int exp);
float ldexpf(float x, int exp);
long double ldexpl(long double x, int exp);
```

DESCRIPTION

The functionality described on this reference page is aligned with the ISO C standard. Any conflict between the requirements described here and the ISO C standard is unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

These functions shall compute the quantity $x * 2^{exp}$.

An application wishing to check for error situations should set *errno* to zero and call *feclearexcept*(FE_ALL_EXCEPT) before calling these functions. On return, if *errno* is non-zero or *fetestexcept*(FE_INVALID | FE_DIVBYZERO | FE_OVERFLOW | FE_UNDERFLOW) is non-zero, an error has occurred.

RETURN VALUE

Upon successful completion, these functions shall return *x* multiplied by 2, raised to the power *exp*.

If these functions would cause overflow, a range error shall occur and *ldexp*(), *ldexpf*(), and *ldexpl*() shall return $\pm\text{HUGE\_VAL}$, $\pm\text{HUGE\_VALF}$, and $\pm\text{HUGE\_VALL}$ (according to the sign of *x*), respectively.

If the correct value would cause underflow, and is not representable, a range error may occur, and *ldexp*(), *ldexpf*(), and *ldexpl*() shall return 0.0, or (if IEC 60559 Floating-Point is not supported) an implementation-defined value no greater in magnitude than DBL_MIN, FLT_MIN, and LDBL_MIN, respectively.

If *x* is NaN, a NaN shall be returned.

If *x* is ± 0 or $\pm\text{Inf}$, *x* shall be returned.

If *exp* is 0, *x* shall be returned.

If the correct value would cause underflow, and is representable, a range error may occur and the correct value shall be returned.

ERRORS

These functions shall fail if:

Range Error The result overflows.

If the integer expression (*math_errhandling* & MATH_ERRNO) is non-zero, then *errno* shall be set to [ERANGE]. If the integer expression (*math_errhandling* & MATH_ERREXCEPT) is non-zero, then the overflow floating-point exception shall be raised.

These functions may fail if:

Range Error The result underflows.

If the integer expression (*math_errhandling* & MATH_ERRNO) is non-zero, then *errno* shall be set to [ERANGE]. If the integer expression (*math_errhandling* & MATH_ERREXCEPT) is non-zero, then the underflow

40628 floating-point exception shall be raised.

40629 **EXAMPLES**

40630 None.

40631 **APPLICATION USAGE**

40632 On error, the expressions (*math_errhandling* & MATH_ERRNO) and (*math_errhandling* &
40633 MATH_ERREXCEPT) are independent of each other, but at least one of them must be non-zero.

40634 **RATIONALE**

40635 None.

40636 **FUTURE DIRECTIONS**

40637 None.

40638 **SEE ALSO**

40639 *feclearexcept()*, *fetestexcept()*, *frexp()*, *isnan()*

40640 XBD [Section 4.19](#) (on page 116), **<math.h>**

40641 **CHANGE HISTORY**

40642 First released in Issue 1. Derived from Issue 1 of the SVID.

40643 **Issue 5**

40644 The DESCRIPTION is updated to indicate how an application should check for an error. This
40645 text was previously published in the APPLICATION USAGE section.

40646 **Issue 6**

40647 The *ldexpf()* and *ldexpl()* functions are added for alignment with the ISO/IEC 9899:1999
40648 standard.

40649 The DESCRIPTION, RETURN VALUE, ERRORS, and APPLICATION USAGE sections are
40650 revised to align with the ISO/IEC 9899:1999 standard.

40651 IEC 60559:1989 standard floating-point extensions over the ISO/IEC 9899:1999 standard are
40652 marked.

40653 **Issue 7**

40654 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0352 [68] and XSH/TC1-2008/0353 +
40655 [68] are applied.

NAME

ldiv, lldiv — compute quotient and remainder of a long division

SYNOPSIS

```
#include <stdlib.h>
```

```
ldiv_t ldiv(long numer, long denom);
```

```
lldiv_t lldiv(long long numer, long long denom);
```

DESCRIPTION

CX The functionality described on this reference page is aligned with the ISO C standard. Any conflict between the requirements described here and the ISO C standard is unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

These functions shall compute the quotient and remainder of the division of the numerator *numer* by the denominator *denom*. If the division is inexact, the resulting quotient is the **long** integer (for the *ldiv()* function) or **long long** integer (for the *lldiv()* function) of lesser magnitude that is the nearest to the algebraic quotient. If the result cannot be represented, the behavior is undefined; otherwise, *quot* * *denom* + *rem* shall equal *numer*.

RETURN VALUE

The *ldiv()* function shall return a structure of type **ldiv_t**, comprising both the quotient and the remainder. The structure shall include the following members, in any order:

```
long    quot;    /* Quotient */
long    rem;     /* Remainder */
```

The *lldiv()* function shall return a structure of type **lldiv_t**, comprising both the quotient and the remainder. The structure shall include the following members, in any order:

```
long long quot;    /* Quotient */
long long rem;     /* Remainder */
```

ERRORS

No errors are defined.

EXAMPLES

None.

APPLICATION USAGE

None.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

[*div\(\)*](#)

XBD [`<stdlib.h>`](#)

CHANGE HISTORY

First released in Issue 4. Derived from the ISO C standard.

Issue 6

The *lldiv()* function is added for alignment with the ISO/IEC 9899:1999 standard.

40697 **NAME**

40698 lfind — find entry in a linear search table

40699 **SYNOPSIS**

```
40700 XSI      #include <search.h>
40701          void *lfind(const void *key, const void *base, size_t *nelp,
40702                      size_t width, int (*compar)(const void *, const void *));
```

40703 **DESCRIPTION**40704 Refer to *lsearch()*.

40705 **NAME**

40706 lgamma, lgammaf, lgammal, signgam — log gamma function

40707 **SYNOPSIS**

```
40708 #include <math.h>
40709 double lgamma(double x);
40710 float lgammaf(float x);
40711 long double lgammal(long double x);
40712 XSI extern int signgam;
```

40713 **DESCRIPTION**

40714 CX The functionality described on this reference page is aligned with the ISO C standard. Any
 40715 conflict between the requirements described here and the ISO C standard is unintentional. This
 40716 volume of POSIX.1-2008 defers to the ISO C standard.

40717 These functions shall compute $\log_e |\Gamma(x)|$ where $\Gamma(x)$ is defined as $\int_0^{\infty} e^{-t} t^{x-1} dt$. The argument x
 40718 need not be a non-positive integer ($\Gamma(x)$ is defined over the reals, except the non-positive
 40719 integers).

40720 XSI If x is NaN, $-\text{Inf}$, or a negative integer, the value of *signgam* is unspecified.

40721 CX These functions need not be thread-safe.

40722 An application wishing to check for error situations should set *errno* to zero and call
 40723 *feclearexcept*(FE_ALL_EXCEPT) before calling these functions. On return, if *errno* is non-zero or
 40724 *fetestexcept*(FE_INVALID | FE_DIVBYZERO | FE_OVERFLOW | FE_UNDERFLOW) is non-
 40725 zero, an error has occurred.

40726 **RETURN VALUE**

40727 Upon successful completion, these functions shall return the logarithmic gamma of x .

40728 If x is a non-positive integer, a pole error shall occur and *lgamma*(), *lgammaf*(), and *lgammal*()
 40729 shall return +HUGE_VAL, +HUGE_VALF, and +HUGE_VALL, respectively.

40730 If the correct value would cause overflow, a range error shall occur and *lgamma*(), *lgammaf*(),
 40731 and *lgammal*() shall return $\pm\text{HUGE\_VAL}$, $\pm\text{HUGE\_VALF}$, and $\pm\text{HUGE\_VALL}$ (having the same
 40732 sign as the correct value), respectively.

40733 MX If x is NaN, a NaN shall be returned.

40734 If x is 1 or 2, +0 shall be returned.

40735 If x is $\pm\text{Inf}$, +Inf shall be returned.

40736 **ERRORS**

40737 These functions shall fail if:

40738 Pole Error The x argument is a negative integer or zero.

40739 If the integer expression (*math_errhandling* & MATH_ERRNO) is non-zero,
 40740 then *errno* shall be set to [ERANGE]. If the integer expression
 40741 (*math_errhandling* & MATH_ERREXCEPT) is non-zero, then the divide-by-zero
 40742 floating-point exception shall be raised.

40743 Range Error The result overflows.

40744 If the integer expression (*math_errhandling* & MATH_ERRNO) is non-zero,
 40745 then *errno* shall be set to [ERANGE]. If the integer expression

40746 (math_errhandling & MATH_ERREXCEPT) is non-zero, then the overflow
 40747 floating-point exception shall be raised.

40748 EXAMPLES

40749 None.

40750 APPLICATION USAGE

40751 On error, the expressions (math_errhandling & MATH_ERRNO) and (math_errhandling &
 40752 MATH_ERREXCEPT) are independent of each other, but at least one of them must be non-zero.

40753 RATIONALE

40754 None.

40755 FUTURE DIRECTIONS

40756 None.

40757 SEE ALSO

40758 *exp()*, *feclearexcept()*, *fetestexcept()*, *isnan()*

40759 XBD Section 4.19 (on page 116), <math.h>

40760 CHANGE HISTORY

40761 First released in Issue 3.

40762 Issue 5

40763 The DESCRIPTION is updated to indicate how an application should check for an error. This
 40764 text was previously published in the APPLICATION USAGE section.

40765 A note indicating that this function need not be reentrant is added to the DESCRIPTION.

40766 Issue 6

40767 The *lgamma()* function is no longer marked as an extension.

40768 The *lgammaf()* and *lgammal()* functions are added for alignment with the ISO/IEC 9899:1999
 40769 standard.

40770 The DESCRIPTION, RETURN VALUE, ERRORS, and APPLICATION USAGE sections are
 40771 revised to align with the ISO/IEC 9899:1999 standard.

40772 IEC 60559:1989 standard floating-point extensions over the ISO/IEC 9899:1999 standard are
 40773 marked.

40774 Functionality relating to the XSI option is marked.

40775 Issue 7

40776 Austin Group Interpretation 1003.1-2001 #156 is applied.

40777 The DESCRIPTION is clarified regarding the value of *signgam* when *x* is Nan, -Inf, or a negative
 40778 integer.

NAME

link, linkat — link one file to another file relative to two directory file descriptors

SYNOPSIS

```
#include <unistd.h>
```

```
int link(const char *path1, const char *path2);
int linkat(int fd1, const char *path1, int fd2,
           const char *path2, int flag);
```

DESCRIPTION

The *link()* function shall create a new link (directory entry) for the existing file, *path1*.

The *path1* argument points to a pathname naming an existing file. The *path2* argument points to a pathname naming the new directory entry to be created. The *link()* function shall atomically create a new link for the existing file and the link count of the file shall be incremented by one.

If *path1* names a directory, *link()* shall fail unless the process has appropriate privileges and the implementation supports using *link()* on directories.

If *path1* names a symbolic link, it is implementation-defined whether *link()* follows the symbolic link, or creates a new link to the symbolic link itself.

Upon successful completion, *link()* shall mark for update the last file status change timestamp of the file. Also, the last data modification and last file status change timestamps of the directory that contains the new entry shall be marked for update.

If *link()* fails, no link shall be created and the link count of the file shall remain unchanged.

The implementation may require that the calling process has permission to access the existing file.

The *linkat()* function shall be equivalent to the *link()* function except that symbolic links shall be handled as specified by the value of *flag* (see below) and except in the case where either *path1* or *path2* or both are relative paths. In this case a relative path *path1* is interpreted relative to the directory associated with the file descriptor *fd1* instead of the current working directory and similarly for *path2* and the file descriptor *fd2*. If the file descriptor was opened without *O_SEARCH*, the function shall check whether directory searches are permitted using the current permissions of the directory underlying the file descriptor. If the file descriptor was opened with *O_SEARCH*, the function shall not perform the check.

Values for *flag* are constructed by a bitwise-inclusive OR of flags from the following list, defined in *<fcntl.h>*:

AT_SYMLINK_FOLLOW

If *path1* names a symbolic link, a new link for the target of the symbolic link is created.

If *linkat()* is passed the special value *AT_FDCWD* in the *fd1* or *fd2* parameter, the current working directory shall be used for the respective *path* argument. If both *fd1* and *fd2* have value *AT_FDCWD*, the behavior shall be identical to a call to *link()*, except that symbolic links shall be handled as specified by the value of *flag*.

If the *AT_SYMLINK_FOLLOW* flag is clear in the *flag* argument and the *path1* argument names a symbolic link, a new link is created for the symbolic link *path1* and not its target.

RETURN VALUE

Upon successful completion, these functions shall return 0. Otherwise, these functions shall return -1 and set *errno* to indicate the error.

40822 **ERRORS**

40823 These functions shall fail if:

40824 [EACCES] A component of either path prefix denies search permission, or the requested
 40825 link requires writing in a directory that denies write permission, or the calling
 40826 process does not have permission to access the existing file and this is required
 40827 by the implementation.

40828 [EEXIST] The *path2* argument resolves to an existing directory entry or refers to a
 40829 symbolic link.

40830 [ELOOP] A loop exists in symbolic links encountered during resolution of the *path1* or
 40831 *path2* argument.

40832 [EMLINK] The number of links to the file named by *path1* would exceed {LINK_MAX}.

40833 [ENAMETOOLONG]

40834 The length of a component of a pathname is longer than {NAME_MAX}.

40835 [ENOENT] A component of either path prefix does not exist; the file named by *path1* does
 40836 not exist; or *path1* or *path2* points to an empty string.

40837 [ENOSPC] The directory to contain the link cannot be extended.

40838 [ENOTDIR] A component of either path prefix names an existing file that is neither a
 40839 directory nor a symbolic link to a directory, or the *path1* argument contains at
 40840 least one non-`<slash>` character and ends with one or more trailing `<slash>`
 40841 characters and the last pathname component names an existing file that is
 40842 neither a directory nor a symbolic link to a directory, or the *path1* argument
 40843 names an existing non-directory file and the *path2* argument names a
 40844 nonexistent file, contains at least one non-`<slash>` character, and ends with
 40845 one or more trailing `<slash>` characters.

40846 [EPERM] The file named by *path1* is a directory and either the calling process does not
 40847 have appropriate privileges or the implementation prohibits using *link()* on
 40848 directories.

40849 [EROFS] The requested link requires writing in a directory on a read-only file system.

40850 [EXDEV] The link named by *path2* and the file named by *path1* are on different file
 40851 systems and the implementation does not support links between file systems.

40852 OB XSR [EXDEV] *path1* refers to a named STREAM.

40853 The *linkat()* function shall fail if:

40854 [EBADF] The *path1* or *path2* argument does not specify an absolute path and the *fd1* or
 40855 *fd2* argument, respectively, is neither AT_FDCWD nor a valid file descriptor
 40856 open for reading or searching.

40857 [ENOTDIR] The *path1* or *path2* argument is not an absolute path and *fd1* or *fd2*,
 40858 respectively, is a file descriptor associated with a non-directory file.

40859 These functions may fail if:

40860 [ELOOP] More than {SYMLOOP_MAX} symbolic links were encountered during
 40861 resolution of the *path1* or *path2* argument.

40862 [ENAMETOOLONG]

40863 The length of a pathname exceeds {PATH_MAX}, or pathname resolution of a
 40864 symbolic link produced an intermediate result with a length that exceeds

40865 {PATH_MAX}.

40866 The *linkat()* function may fail if:

40867 [EINVAL] The value of the *flag* argument is not valid. -

40868 EXAMPLES

40869 Creating a Link to a File

40870 The following example shows how to create a link to a file named **/home/cnd/mod1** by creating
40871 a new directory entry named **/modules/pass1**.

```
40872 #include <unistd.h>
40873 char *path1 = "/home/cnd/mod1";
40874 char *path2 = "/modules/pass1";
40875 int status;
40876 ...
40877 status = link (path1, path2);
```

40878 Creating a Link to a File Within a Program

40879 In the following program example, the *link()* function links the **/etc/passwd** file (defined as
40880 **PASSWDFILE**) to a file named **/etc/opasswd** (defined as **SAVEFILE**), which is used to save the
40881 current password file. Then, after removing the current password file (defined as
40882 **PASSWDFILE**), the new password file is saved as the current password file using the *link()*
40883 function again.

```
40884 #include <unistd.h>
40885 #define LOCKFILE "/etc/ptmp"
40886 #define PASSWDFILE "/etc/passwd"
40887 #define SAVEFILE "/etc/opasswd"
40888 ...
40889 /* Save current password file */
40890 link (PASSWDFILE, SAVEFILE);
40891
40892 /* Remove current password file. */
40893 unlink (PASSWDFILE);
40894
40895 /* Save new password file as current password file. */
40896 link (LOCKFILE, PASSWDFILE);
```

40895 APPLICATION USAGE

40896 Some implementations do allow links between file systems.

40897 If *path1* refers to a symbolic link, application developers should use *linkat()* with appropriate
40898 flags to select whether or not the symbolic link should be resolved. |

40899 RATIONALE

40900 Linking to a directory is restricted to the superuser in most historical implementations because
40901 this capability may produce loops in the file hierarchy or otherwise corrupt the file system. This
40902 volume of POSIX.1-2008 continues that philosophy by prohibiting *link()* and *unlink()* from
40903 doing this. Other functions could do it if the implementor designed such an extension. |

40904 Some historical implementations allow linking of files on different file systems. Wording was
40905 added to explicitly allow this optional behavior. |

The exception for cross-file system links is intended to apply only to links that are programmatically indistinguishable from “hard” links.

The purpose of the *linkat()* function is to link files in directories other than the current working directory without exposure to race conditions. Any part of the path of a file could be changed in parallel to a call to *link()*, resulting in unspecified behavior. By opening a file descriptor for the directory of both the existing file and the target location and using the *linkat()* function it can be guaranteed that the both filenames are in the desired directories.

The AT_SYMLINK_FOLLOW flag allows for implementing both common behaviors of the *link()* function. The POSIX specification requires that if *path1* is a symbolic link, a new link for the target of the symbolic link is created. Many systems by default or as an alternative provide a mechanism to avoid the implicit symbolic link lookup and create a new link for the symbolic link itself.

Earlier versions of this standard specified only the *link()* function, and required it to behave like *linkat()* with the AT_SYMLINK_FOLLOW flag. However, historical practice from SVR4 and Linux kernels had *link()* behaving like *linkat()* with no flags, and many systems that attempted to provide a conforming *link()* function did so in a way that was rarely used, and when it was used did not conform to the standard (e.g., by not being atomic, or by dereferencing the symbolic link incorrectly). Since applications could not rely on *link()* following links in practice, the *linkat()* function was added taking a flag to specify the desired behavior for the application.

FUTURE DIRECTIONS

None.

SEE ALSO

rename(), *symlink()*, *unlink()*

XBD <fcntl.h>, <unistd.h>

CHANGE HISTORY

First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The [ELOOP] mandatory error condition is added.
- A second [ENAMETOOLONG] is added as an optional error condition.

The following changes were made to align with the IEEE P1003.1a draft standard:

- An explanation is added of the action when *path2* refers to a symbolic link.
- The [ELOOP] optional error condition is added.

Issue 7

Austin Group Interpretation 1003.1-2001 #143 is applied.

SD5-XSH-ERN-93 is applied, adding RATIONALE.

The *linkat()* function is added from The Open Group Technical Standard, 2006, Extended API Set Part 2.

Functionality relating to XSI STREAMS is marked obsolescent.

Changes are made related to support for finegrained timestamps.

The [EOPNOTSUPP] error is removed.

40948 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0354 [326], XSH/TC1-2008/0355 [461], +
40949 XSH/TC1-2008/0356 [326], XSH/TC1-2008/0357 [324], XSH/TC1-2008/0358 [147,429], +
40950 XSH/TC1-2008/0359 [277], XSH/TC1-2008/0360 [278], and XSH/TC1-2008/0361 [278] are +
40951 applied.

NAME

lio_listio — list directed I/O

SYNOPSIS

```
#include <aio.h>

int lio_listio(int mode, struct aiocb *restrict const list[restrict],
               int nent, struct sigevent *restrict sig);
```

DESCRIPTION

The *lio_listio()* function shall initiate a list of I/O requests with a single function call.

The *mode* argument takes one of the values LIO_WAIT or LIO_NOWAIT declared in **<aio.h>** and determines whether the function returns when the I/O operations have been completed, or as soon as the operations have been queued. If the *mode* argument is LIO_WAIT, the function shall wait until all I/O is complete and the *sig* argument shall be ignored.

If the *mode* argument is LIO_NOWAIT, the function shall return immediately, and asynchronous notification shall occur, according to the *sig* argument, when all the I/O operations complete. If *sig* is NULL, then no asynchronous notification shall occur. If *sig* is not NULL, asynchronous notification occurs as specified in [Section 2.4.1](#) (on page 488) when all the requests in *list* have completed.

The I/O requests enumerated by *list* are submitted in an unspecified order.

The *list* argument is an array of pointers to **aiocb** structures. The array contains *nent* elements. The array may contain NULL elements, which shall be ignored.

If the buffer pointed to by *list* or the **aiocb** structures pointed to by the elements of the array *list* become illegal addresses before all asynchronous I/O completed and, if necessary, the notification is sent, then the behavior is undefined. If the buffers pointed to by the *aio_buf* member of the **aiocb** structure pointed to by the elements of the array *list* become illegal addresses prior to the asynchronous I/O associated with that **aiocb** structure being completed, the behavior is undefined.

The *aio_lio_opcode* field of each **aiocb** structure specifies the operation to be performed. The supported operations are LIO_READ, LIO_WRITE, and LIO_NOP; these symbols are defined in **<aio.h>**. The LIO_NOP operation causes the list entry to be ignored. If the *aio_lio_opcode* element is equal to LIO_READ, then an I/O operation is submitted as if by a call to *aio_read()* with the *aio_cbp* equal to the address of the **aiocb** structure. If the *aio_lio_opcode* element is equal to LIO_WRITE, then an I/O operation is submitted as if by a call to *aio_write()* with the *aio_cbp* equal to the address of the **aiocb** structure.

The *aio_fildes* member specifies the file descriptor on which the operation is to be performed.

The *aio_buf* member specifies the address of the buffer to or from which the data is transferred.

The *aio_nbytes* member specifies the number of bytes of data to be transferred.

The members of the **aiocb** structure further describe the I/O operation to be performed, in a manner identical to that of the corresponding **aiocb** structure when used by the *aio_read()* and *aio_write()* functions.

The *nent* argument specifies how many elements are members of the list; that is, the length of the array.

The behavior of this function is altered according to the definitions of synchronized I/O data integrity completion and synchronized I/O file integrity completion if synchronized I/O is enabled on the file associated with *aio_fildes*.

For regular files, no data transfer shall occur past the offset maximum established in the open file description associated with *aio*cbp->*aio_fildes*.

If *sig*->*sigev_notify* is SIGEV_THREAD and *sig*->*sigev_notify_attributes* is a non-null pointer and the block pointed to by this pointer becomes an illegal address prior to all asynchronous I/O being completed, then the behavior is undefined.

RETURN VALUE

If the *mode* argument has the value LIO_NOWAIT, the *lio_listio*() function shall return the value zero if the I/O operations are successfully queued; otherwise, the function shall return the value -1 and set *errno* to indicate the error.

If the *mode* argument has the value LIO_WAIT, the *lio_listio*() function shall return the value zero when all the indicated I/O has completed successfully. Otherwise, *lio_listio*() shall return a value of -1 and set *errno* to indicate the error.

In either case, the return value only indicates the success or failure of the *lio_listio*() call itself, not the status of the individual I/O requests. In some cases one or more of the I/O requests contained in the list may fail. Failure of an individual request does not prevent completion of any other individual request. To determine the outcome of each I/O request, the application shall examine the error status associated with each **aio**cb control block. The error statuses so returned are identical to those returned as the result of an *aio_read*() or *aio_write*() function.

ERRORS

The *lio_listio*() function shall fail if:

41016 41017 41018	[EAGAIN]	The resources necessary to queue all the I/O requests were not available. The application may check the error status for each aio cb to determine the individual request(s) that failed.
41019 41020	[EAGAIN]	The number of entries indicated by <i>nent</i> would cause the system-wide limit {AIO_MAX} to be exceeded.
41021 41022	[EINVAL]	The <i>mode</i> argument is not a proper value, or the value of <i>nent</i> was greater than {AIO_LISTIO_MAX}.
41023 41024 41025 41026 41027 41028 41029	[EINTR]	A signal was delivered while waiting for all I/O requests to complete during an LIO_WAIT operation. Note that, since each I/O operation invoked by <i>lio_listio</i> () may possibly provoke a signal when it completes, this error return may be caused by the completion of one (or more) of the very I/O operations being awaited. Outstanding I/O requests are not canceled, and the application shall examine each list element to determine whether the request was initiated, canceled, or completed.
41030 41031 41032	[EIO]	One or more of the individual I/O operations failed. The application may check the error status for each aio cb structure to determine the individual request(s) that failed.

In addition to the errors returned by the *lio_listio*() function, if the *lio_listio*() function succeeds or fails with errors of [EAGAIN], [EINTR], or [EIO], then some of the I/O specified by the list may have been initiated. If the *lio_listio*() function fails with an error code other than [EAGAIN], [EINTR], or [EIO], no operations from the list shall have been initiated. The I/O operation indicated by each list element can encounter errors specific to the individual read or write function being performed. In this event, the error status for each **aio**cb control block contains the associated error code. The error codes that can be set are the same as would be set by a *read*() or *write*() function, with the following additional error codes possible:

- 41041 [EAGAIN] The requested I/O operation was not queued due to resource limitations.
- 41042 [ECANCELED] The requested I/O was canceled before the I/O completed due to an explicit *aio_cancel()* request.
- 41043
- 41044 [EFBIG] The *aiocbp->aio_lio_opcode* is LIO_WRITE, the file is a regular file, *aiocbp->aio_nbytes* is greater than 0, and the *aiocbp->aio_offset* is greater than or equal to the offset maximum in the open file description associated with *aiocbp->aio_fildes*.
- 41045
- 41046
- 41047
- 41048 [EINPROGRESS] The requested I/O is in progress.
- 41049 [EOVERFLOW] The *aiocbp->aio_lio_opcode* is LIO_READ, the file is a regular file, *aiocbp->aio_nbytes* is greater than 0, and the *aiocbp->aio_offset* is before the end-of-file and is greater than or equal to the offset maximum in the open file description associated with *aiocbp->aio_fildes*.
- 41050
- 41051
- 41052

EXAMPLES

41053 None.

41054

APPLICATION USAGE

41055 None.

41056

RATIONALE

41057 Although it may appear that there are inconsistencies in the specified circumstances for error codes, the [EIO] error condition applies when any circumstance relating to an individual operation makes that operation fail. This might be due to a badly formulated request (for example, the *aio_lio_opcode* field is invalid, and *aio_error()* returns [EINVAL]) or might arise from application behavior (for example, the file descriptor is closed before the operation is initiated, and *aio_error()* returns [EBADF]).

41058

41059

41060

41061

41062

41063

41064 The limitation on the set of error codes returned when operations from the list shall have been initiated enables applications to know when operations have been started and whether *aio_error()* is valid for a specific operation.

41065

41066

FUTURE DIRECTIONS

41067 None.

41068

SEE ALSO

41069 *aio_read()*, *aio_write()*, *aio_error()*, *aio_return()*, *aio_cancel()*, *close()*, *exec*, *exit()*, *fork()*, *lseek()*, *read()*

41070

41071

41072 XBD <*aio.h*>

CHANGE HISTORY

41073 First released in Issue 5. Included for alignment with the POSIX Realtime Extension.

41074

41075 Large File Summit extensions are added.

Issue 6

41076 The [ENOSYS] error condition has been removed as stubs need not be provided if an implementation does not support the Asynchronous Input and Output option.

41077

41078

41079 The *lio_listio()* function is marked as part of the Asynchronous Input and Output option.

41080 The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

41081

- In the DESCRIPTION, text is added to indicate that for regular files no data transfer occurs past the offset maximum established in the open file description associated with *aiocbp*→*aio_fildes*. This change is to support large files.

- The [EBIG] and [EOVERFLOW] error conditions are defined. This change is to support large files.

The normative text is updated to avoid use of the term “must” for application requirements.

The **restrict** keyword is added to the *lio_listio()* prototype for alignment with the ISO/IEC 9899:1999 standard.

Issue 6

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/53 is applied, adding new text for symmetry with the *aio_read()* and *aio_write()* functions to the DESCRIPTION.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/54 is applied, adding text to the DESCRIPTION making it explicit that the user is required to keep the structure pointed to by *sig*→*sigev_notify_attributes* valid until the last asynchronous operation finished and the notification has been sent.

Issue 7

The *lio_listio()* function is moved from the Asynchronous Input and Output option to the Base.

41099 **NAME**

41100 listen — listen for socket connections and limit the queue of incoming connections

41101 **SYNOPSIS**

41102 #include <sys/socket.h>

41103 int listen(int *socket*, int *backlog*);41104 **DESCRIPTION**41105 The *listen()* function shall mark a connection-mode socket, specified by the *socket* argument, as
41106 accepting connections.41107 The *backlog* argument provides a hint to the implementation which the implementation shall use
41108 to limit the number of outstanding connections in the socket's listen queue. Implementations
41109 may impose a limit on *backlog* and silently reduce the specified value. Normally, a larger *backlog*
41110 argument value shall result in a larger or equal length of the listen queue. Implementations shall
41111 support values of *backlog* up to SOMAXCONN, defined in <sys/socket.h>.41112 The implementation may include incomplete connections in its listen queue. The limits on the
41113 number of incomplete connections and completed connections queued may be different.41114 The implementation may have an upper limit on the length of the listen queue—either global or
41115 per accepting socket. If *backlog* exceeds this limit, the length of the listen queue is set to the limit.41116 If *listen()* is called with a *backlog* argument value that is less than 0, the function behaves as if it
41117 had been called with a *backlog* argument value of 0.41118 A *backlog* argument of 0 may allow the socket to accept connections, in which case the length of
41119 the listen queue may be set to an implementation-defined minimum value.41120 The socket in use may require the process to have appropriate privileges to use the *listen()*
41121 function.41122 **RETURN VALUE**41123 Upon successful completions, *listen()* shall return 0; otherwise, -1 shall be returned and *errno* set
41124 to indicate the error.41125 **ERRORS**41126 The *listen()* function shall fail if:41127 [EBADF] The *socket* argument is not a valid file descriptor.

41128 [EDESTADDRREQ]

41129 The socket is not bound to a local address, and the protocol does not support
41130 listening on an unbound socket.41131 [EINVAL] The *socket* is already connected.41132 [ENOTSOCK] The *socket* argument does not refer to a socket.41133 [EOPNOTSUPP] The socket protocol does not support *listen()*.41134 The *listen()* function may fail if:

41135 [EACCES] The calling process does not have appropriate privileges.

41136 [EINVAL] The *socket* has been shut down.

41137 [ENOBUFS] Insufficient resources are available in the system to complete the call.

41138 EXAMPLES

41139 None.

41140 APPLICATION USAGE

41141 None.

41142 RATIONALE

41143 None.

41144 FUTURE DIRECTIONS

41145 None.

41146 SEE ALSO

41147 [*accept\(\)*](#), [*connect\(\)*](#), [*socket\(\)*](#)

41148 XBD [*<sys/socket.h>*](#)

41149 CHANGE HISTORY

41150 First released in Issue 6. Derived from the XNS, Issue 5.2 specification.

41151 The DESCRIPTION is updated to describe the relationship of SOMAXCONN and the *backlog*
41152 argument.

41153 **NAME**

41154 llabs — return a long integer absolute value

41155 **SYNOPSIS**

41156 #include <stdlib.h>

41157 long long llabs(long long *i*);41158 **DESCRIPTION**41159 Refer to *labs()*.

41160 **NAME**

41161 lldiv — compute quotient and remainder of a long division

41162 **SYNOPSIS**

41163 #include <stdlib.h>

41164 lldiv_t lldiv(long long *numer*, long long *denom*);

41165 **DESCRIPTION**

41166 Refer to *ldiv()*.

41167 **NAME**

41168 llrint, llrintf, llrintl — round to the nearest integer value using current rounding direction

41169 **SYNOPSIS**

```
41170 #include <math.h>
41171 long long llrint(double x);
41172 long long llrintf(float x);
41173 long long llrintl(long double x);
```

41174 **DESCRIPTION**

41175 CX The functionality described on this reference page is aligned with the ISO C standard. Any
 41176 conflict between the requirements described here and the ISO C standard is unintentional. This
 41177 volume of POSIX.1-2008 defers to the ISO C standard.

41178 These functions shall round their argument to the nearest integer value, rounding according to
 41179 the current rounding direction.

41180 An application wishing to check for error situations should set *errno* to zero and call
 41181 *feclearexcept*(FE_ALL_EXCEPT) before calling these functions. On return, if *errno* is non-zero or
 41182 *fetestexcept*(FE_INVALID | FE_DIVBYZERO | FE_OVERFLOW | FE_UNDERFLOW) is non-
 41183 zero, an error has occurred.

41184 **RETURN VALUE**

41185 Upon successful completion, these functions shall return the rounded integer value.

41186 MX If *x* is NaN, a domain error shall occur, and an unspecified value is returned.

41187 If *x* is +Inf, a domain error shall occur and an unspecified value is returned.

41188 If *x* is -Inf, a domain error shall occur and an unspecified value is returned.

41189 If the correct value is positive and too large to represent as a **long long**, an unspecified value
 41190 MX shall be returned. On systems that support the IEC 60559 Floating-Point option, a domain error
 41191 CX shall occur; otherwise, a domain error may occur.

41192 If the correct value is negative and too large to represent as a **long long**, an unspecified value
 41193 MX shall be returned. On systems that support the IEC 60559 Floating-Point option, a domain error
 41194 CX shall occur; otherwise, a domain error may occur.

41195 **ERRORS**

41196 These functions shall fail if:

41197 MX **Domain Error** The *x* argument is NaN or $\pm$ Inf, or the correct value is not representable as an
 41198 integer.

41199 If the integer expression (*math_errhandling* & MATH_ERRNO) is non-zero,
 41200 then *errno* shall be set to [EDOM]. If the integer expression (*math_errhandling*
 41201 & MATH_ERREXCEPT) is non-zero, then the invalid floating-point exception
 41202 shall be raised.

41203 These functions may fail if:

41204 **Domain Error** The correct value is not representable as an integer.

41205 If the integer expression (*math_errhandling* & MATH_ERRNO) is non-zero,
 41206 then *errno* shall be set to [EDOM]. If the integer expression (*math_errhandling*
 41207 & MATH_ERREXCEPT) is non-zero, then the invalid floating-point exception
 41208 shall be raised.

41209 EXAMPLES

41210 None.

41211 APPLICATION USAGE

41212 On error, the expressions *(math_errhandling & MATH_ERRNO)* and *(math_errhandling &*
41213 *MATH_ERREXCEPT)* are independent of each other, but at least one of them must be non-zero.

41214 RATIONALE

41215 These functions provide floating-to-integer conversions. They round according to the current
41216 rounding direction. If the rounded value is outside the range of the return type, the numeric
41217 result is unspecified and the invalid floating-point exception is raised. When they raise no other
41218 floating-point exception and the result differs from the argument, they raise the inexact floating-
41219 point exception.

41220 FUTURE DIRECTIONS

41221 None.

41222 SEE ALSO

41223 *feclearexcept()*, *fetestexcept()*, *lrint()*

41224 XBD [Section 4.19](#) (on page 116), [<math.h>](#)

41225 CHANGE HISTORY

41226 First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

41227 Issue 7

41228 ISO/IEC 9899:1999 standard, Technical Corrigendum 2 #53 is applied.

41229 **NAME**

41230 llround, llroundf, llroundl — round to nearest integer value

41231 **SYNOPSIS**

```
41232 #include <math.h>
41233 long long llround(double x);
41234 long long llroundf(float x);
41235 long long llroundl(long double x);
```

41236 **DESCRIPTION**

41237 CX The functionality described on this reference page is aligned with the ISO C standard. Any
 41238 conflict between the requirements described here and the ISO C standard is unintentional. This
 41239 volume of POSIX.1-2008 defers to the ISO C standard.

41240 These functions shall round their argument to the nearest integer value, rounding halfway cases
 41241 away from zero, regardless of the current rounding direction.

41242 An application wishing to check for error situations should set *errno* to zero and call
 41243 *feclearexcept*(FE_ALL_EXCEPT) before calling these functions. On return, if *errno* is non-zero or
 41244 *fetestexcept*(FE_INVALID | FE_DIVBYZERO | FE_OVERFLOW | FE_UNDERFLOW) is non-
 41245 zero, an error has occurred.

41246 **RETURN VALUE**

41247 Upon successful completion, these functions shall return the rounded integer value.

41248 MX If *x* is NaN, a domain error shall occur, and an unspecified value is returned.

41249 If *x* is +Inf, a domain error shall occur and an unspecified value is returned.

41250 If *x* is -Inf, a domain error shall occur and an unspecified value is returned.

41251 If the correct value is positive and too large to represent as a **long long**, an unspecified value
 41252 MX shall be returned. On systems that support the IEC 60559 Floating-Point option, a domain error
 41253 CX shall occur; otherwise, a domain error may occur.

41254 If the correct value is negative and too large to represent as a **long long**, an unspecified value
 41255 MX shall be returned. On systems that support the IEC 60559 Floating-Point option, a domain error
 41256 CX shall occur; otherwise, a domain error may occur.

41257 **ERRORS**

41258 These functions shall fail if:

41259 MX **Domain Error** The *x* argument is NaN or $\pm$ Inf, or the correct value is not representable as an
 41260 integer.

41261 If the integer expression (*math_errhandling* & MATH_ERRNO) is non-zero,
 41262 then *errno* shall be set to [EDOM]. If the integer expression (*math_errhandling*
 41263 & MATH_ERREXCEPT) is non-zero, then the invalid floating-point exception
 41264 shall be raised.

41265 These functions may fail if:

41266 **Domain Error** The correct value is not representable as an integer.

41267 If the integer expression (*math_errhandling* & MATH_ERRNO) is non-zero,
 41268 then *errno* shall be set to [EDOM]. If the integer expression (*math_errhandling*
 41269 & MATH_ERREXCEPT) is non-zero, then the invalid floating-point exception
 41270 shall be raised.

41271 EXAMPLES

41272 None.

41273 APPLICATION USAGE

41274 On error, the expressions (*math_errhandling* & MATH_ERRNO) and (*math_errhandling* &
41275 MATH_ERREXCEPT) are independent of each other, but at least one of them must be non-zero.

41276 RATIONALE

41277 These functions differ from the *llrint()* functions in that the default rounding direction for the
41278 *llround()* functions round halfway cases away from zero and need not raise the inexact floating-
41279 point exception for non-integer arguments that round to within the range of the return type.

41280 FUTURE DIRECTIONS

41281 None.

41282 SEE ALSO

41283 *feclearexcept()*, *fetestexcept()*, *lround()*

41284 XBD [Section 4.19](#) (on page 116), [<math.h>](#)

41285 CHANGE HISTORY

41286 First released in Issue 6. Derived from the ISO/IEC 9899: 1999 standard.

41287 Issue 7

41288 ISO/IEC 9899: 1999 standard, Technical Corrigendum 2 #54 (SD5-XSH-ERN-75) is applied.

41289 **NAME**

41290 localeconv — return locale-specific information

41291 **SYNOPSIS**

41292 #include <locale.h>

41293 struct lconv *localeconv(void);

41294 **DESCRIPTION**

41295 CX The functionality described on this reference page is aligned with the ISO C standard. Any
 41296 conflict between the requirements described here and the ISO C standard is unintentional. This
 41297 volume of POSIX.1-2008 defers to the ISO C standard.

41298 The *localeconv()* function shall set the components of an object with the type **struct lconv** with
 41299 the values appropriate for the formatting of numeric quantities (monetary and otherwise)
 41300 according to the rules of the current locale.

41301 The members of the structure with type **char *** are pointers to strings, any of which (except
 41302 **decimal_point**) can point to "", to indicate that the value is not available in the current locale or
 41303 is of zero length. The members with type **char** are non-negative numbers, any of which can be
 41304 {CHAR_MAX} to indicate that the value is not available in the current locale.

41305 The members include the following:

41306 **char *decimal_point**

41307 The radix character used to format non-monetary quantities.

41308 **char *thousands_sep**

41309 The character used to separate groups of digits before the decimal-point character in
 41310 formatted non-monetary quantities.

41311 **char *grouping**

41312 A string whose elements taken as one-byte integer values indicate the size of each group of
 41313 digits in formatted non-monetary quantities.

41314 **char *int_curr_symbol**

41315 The international currency symbol applicable to the current locale. The first three characters
 41316 contain the alphabetic international currency symbol in accordance with those specified in
 41317 the ISO 4217:2001 standard. The fourth character (immediately preceding the null byte) is
 41318 the character used to separate the international currency symbol from the monetary
 41319 quantity.

41320 **char *currency_symbol**

41321 The local currency symbol applicable to the current locale.

41322 **char *mon_decimal_point**

41323 The radix character used to format monetary quantities.

41324 **char *mon_thousands_sep**

41325 The separator for groups of digits before the decimal-point in formatted monetary
 41326 quantities.

41327 **char *mon_grouping**

41328 A string whose elements taken as one-byte integer values indicate the size of each group of
 41329 digits in formatted monetary quantities.

41330 **char *positive_sign**

41331 The string used to indicate a non-negative valued formatted monetary quantity.

41332 **char *negative_sign**
41333 The string used to indicate a negative valued formatted monetary quantity.

41334 **char int_frac_digits**
41335 The number of fractional digits (those after the decimal-point) to be displayed in an
41336 internationally formatted monetary quantity.

41337 **char frac_digits**
41338 The number of fractional digits (those after the decimal-point) to be displayed in a
41339 formatted monetary quantity.

41340 **char p_cs_precedes**
41341 Set to 1 if the **currency_symbol** precedes the value for a non-negative formatted monetary
41342 quantity. Set to 0 if the symbol succeeds the value.

41343 **char p_sep_by_space**
41344 Set to a value indicating the separation of the **currency_symbol**, the sign string, and the
41345 value for a non-negative formatted monetary quantity.

41346 **char n_cs_precedes**
41347 Set to 1 if the **currency_symbol** precedes the value for a negative formatted monetary
41348 quantity. Set to 0 if the symbol succeeds the value.

41349 **char n_sep_by_space**
41350 Set to a value indicating the separation of the **currency_symbol**, the sign string, and the
41351 value for a negative formatted monetary quantity.

41352 **char p_sign_posn**
41353 Set to a value indicating the positioning of the **positive_sign** for a non-negative formatted
41354 monetary quantity.

41355 **char n_sign_posn**
41356 Set to a value indicating the positioning of the **negative_sign** for a negative formatted
41357 monetary quantity.

41358 **char int_p_cs_precedes**
41359 Set to 1 or 0 if the **int_curr_symbol** respectively precedes or succeeds the value for a non-
41360 negative internationally formatted monetary quantity.

41361 **char int_n_cs_precedes**
41362 Set to 1 or 0 if the **int_curr_symbol** respectively precedes or succeeds the value for a
41363 negative internationally formatted monetary quantity.

41364 **char int_p_sep_by_space**
41365 Set to a value indicating the separation of the **int_curr_symbol**, the sign string, and the
41366 value for a non-negative internationally formatted monetary quantity.

41367 **char int_n_sep_by_space**
41368 Set to a value indicating the separation of the **int_curr_symbol**, the sign string, and the
41369 value for a negative internationally formatted monetary quantity.

41370 **char int_p_sign_posn**
41371 Set to a value indicating the positioning of the **positive_sign** for a non-negative
41372 internationally formatted monetary quantity.

41373 **char int_n_sign_posn**
41374 Set to a value indicating the positioning of the **negative_sign** for a negative internationally
41375 formatted monetary quantity.

41376 The elements of **grouping** and **mon_grouping** are interpreted according to the following:

41377 {CHAR_MAX} No further grouping is to be performed.

41378 0 The previous element is to be repeatedly used for the remainder of the digits.

41379 *other* The integer value is the number of digits that comprise the current group. The

41380 next element is examined to determine the size of the next group of digits

41381 before the current group.

41382 The values of **p_sep_by_space**, **n_sep_by_space**, **int_p_sep_by_space**, and **int_n_sep_by_space**

41383 are interpreted according to the following:

41384 0 No space separates the currency symbol and value.

41385 1 If the currency symbol and sign string are adjacent, a space separates them from the value;

41386 otherwise, a space separates the currency symbol from the value.

41387 2 If the currency symbol and sign string are adjacent, a space separates them; otherwise, a

41388 space separates the sign string from the value.

41389 For **int_p_sep_by_space** and **int_n_sep_by_space**, the fourth character of **int_curr_symbol** is

41390 used instead of a space.

41391 The values of **p_sign_posn**, **n_sign_posn**, **int_p_sign_posn**, and **int_n_sign_posn** are

41392 interpreted according to the following:

41393 0 Parentheses surround the quantity and **currency_symbol** or **int_curr_symbol**.

41394 1 The sign string precedes the quantity and **currency_symbol** or **int_curr_symbol**.

41395 2 The sign string succeeds the quantity and **currency_symbol** or **int_curr_symbol**.

41396 3 The sign string immediately precedes the **currency_symbol** or **int_curr_symbol**.

41397 4 The sign string immediately succeeds the **currency_symbol** or **int_curr_symbol**.

41398 The implementation shall behave as if no function in this volume of POSIX.1-2008 calls

41399 *localeconv()*.

41400 CX The *localeconv()* function need not be thread-safe.

41401 RETURN VALUE

41402 The *localeconv()* function shall return a pointer to the filled-in object. The application shall not

41403 CX modify the structure to which the return value points, nor any storage areas pointed to by

41404 pointers within the structure. The returned pointer, and pointers within the structure, might be

41405 CX invalidated or the structure or the storage areas might be overwritten by a subsequent call to

41406 CX *localeconv()*. In addition, the returned pointer, and pointers within the structure, might be

41407 CX invalidated or the structure or the storage areas might be overwritten by subsequent calls to

41408 CX *setlocale()* with the categories LC_ALL, LC_MONETARY, or LC_NUMERIC, or by calls to

41409 *uselocale()* which change the categories LC_MONETARY or LC_NUMERIC.

41410 ERRORS

41411 No errors are defined.

EXAMPLES

None.

APPLICATION USAGE

The following table illustrates the rules which may be used by four countries to format monetary quantities.

Country	Positive Format	Negative Format	International Format
Italy	€1.230	−€1.230	EUR.1.230
Netherlands	€ 1.234,56	€ −1.234,56	EUR 1.234,56
Norway	kr1.234,56	kr1.234,56−	NOK 1.234,56
Switzerland	SFrs.1,234.56	SFrs.1,234.56C	CHF 1,234.56

For these four countries, the respective values for the monetary members of the structure returned by *localeconv()* are:

	Italy	Netherlands	Norway	Switzerland
int_curr_symbol	"EUR. "	"EUR "	"NOK "	"CHF "
currency_symbol	"€. "	"€"	"kr"	"SFrs. "
mon_decimal_point	" "	", "	", "	". "
mon_thousands_sep	". "	". "	". "	", "
mon_grouping	"\3"	"\3"	"\3"	"\3"
positive_sign	" "	" "	" "	" "
negative_sign	"_ "	"_ "	"_ "	"C "
int_frac_digits	0	2	2	2
frac_digits	0	2	2	2
p_cs_precedes	1	1	1	1
p_sep_by_space	0	1	0	0
n_cs_precedes	1	1	1	1
n_sep_by_space	0	1	0	0
p_sign_posn	1	1	1	1
n_sign_posn	1	4	2	2
int_p_cs_precedes	1	1	1	1
int_n_cs_precedes	1	1	1	1
int_p_sep_by_space	0	0	0	0
int_n_sep_by_space	0	0	0	0
int_p_sign_posn	1	1	1	1
int_n_sign_posn	1	4	4	2

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

fprintf(), *fscanf()*, *isalpha()*, *isascii()*, *nl_langinfo()*, *setlocale()*, *strcat()*, *strchr()*, *strcmp()*, *strcoll()*, *strcpy()*, *strptime()*, *strlen()*, *strpbrk()*, *strspn()*, *strtok()*, *strxfrm()*, *strtod()*, *uselocale()*

XBD <langinfo.h>, <locale.h>

41454 **CHANGE HISTORY**

41455 First released in Issue 4. Derived from the ANSI C standard.

41456 **Issue 6**

41457 A note indicating that this function need not be reentrant is added to the DESCRIPTION.

41458 The RETURN VALUE section is rewritten to avoid use of the term “must”.

41459 This reference page is updated for alignment with the ISO/IEC 9899: 1999 standard.

41460 ISO/IEC 9899: 1999 standard, Technical Corrigendum 1 is incorporated. -

41461 IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/31 is applied, removing references to
 41462 **int_curr_symbol** and updating the descriptions of **p_sep_by_space** and **n_sep_by_space**. These
 41463 changes are for alignment with the ISO C standard.

41464 **Issue 7**

41465 Austin Group Interpretation 1003.1-2001 #156 is applied.

41466 The definitions of **int_curr_symbol** and **currency_symbol** are updated.

41467 The examples in the APPLICATION USAGE section are updated.

41468 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0362 [75] is applied. +

41469 **NAME**

41470 localtime, localtime_r — convert a time value to a broken-down local time

41471 **SYNOPSIS**

```
41472     #include <time.h>

41473     struct tm *localtime(const time_t *timer);
41474 CX    struct tm *localtime_r(const time_t *restrict timer,
41475                             struct tm *restrict result);
```

41476 **DESCRIPTION**

41477 CX For *localtime()*: The functionality described on this reference page is aligned with the ISO C
 41478 standard. Any conflict between the requirements described here and the ISO C standard is
 41479 unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

41480 The *localtime()* function shall convert the time in seconds since the Epoch pointed to by *timer*
 41481 into a broken-down time, expressed as a local time. The function corrects for the timezone and
 41482 CX any seasonal time adjustments. Local timezone information is used as though *localtime()* calls
 41483 *tzset()*.

41484 The relationship between a time in seconds since the Epoch used as an argument to *localtime()*
 41485 and the **tm** structure (defined in the **<time.h>** header) is that the result shall be as specified in
 41486 the expression given in the definition of seconds since the Epoch (see XBD [Section 4.15](#), on page
 41487 113) corrected for timezone and any seasonal time adjustments, where the names in the structure
 41488 and in the expression correspond.

41489 The same relationship shall apply for *localtime_r()*.

41490 The *localtime()* function need not be thread-safe.

41491 The *asctime()*, *ctime()*, *gmtime()*, and *localtime()* functions shall return values in one of two static
 41492 objects: a broken-down time structure and an array of type **char**. Execution of any of the
 41493 functions may overwrite the information returned in either of these objects by any of the other
 41494 functions.

41495 The *localtime_r()* function shall convert the time in seconds since the Epoch pointed to by *timer*
 41496 into a broken-down time stored in the structure to which *result* points. The *localtime_r()* function
 41497 shall also return a pointer to that same structure.

41498 Unlike *localtime()*, the *localtime_r()* function is not required to set *tzname*. If *localtime_r()* does
 41499 not set *tzname*, it shall not set *daylight* and shall not set *timezone*.

41500 **RETURN VALUE**

41501 Upon successful completion, the *localtime()* function shall return a pointer to the broken-down
 41502 CX time structure. If an error is detected, *localtime()* shall return a null pointer and set *errno* to
 41503 indicate the error.

41504 Upon successful completion, *localtime_r()* shall return a pointer to the structure pointed to by the
 41505 argument *result*. If an error is detected, *localtime_r()* shall return a null pointer and set *errno* to
 41506 indicate the error.

41507 **ERRORS**

41508 CX The *localtime()* and *localtime_r()* functions shall fail if:

41509 CX **[EOVERFLOW]** The result cannot be represented.

41510 **EXAMPLES**41511 **Getting the Local Date and Time**

41512 The following example uses the *time()* function to calculate the time elapsed, in seconds, since
 41513 January 1, 1970 0:00 UTC (the Epoch), *localtime()* to convert that value to a broken-down time,
 41514 and *asctime()* to convert the broken-down time values into a printable string.

```
41515 #include <stdio.h>
41516 #include <time.h>

41517 int main(void)
41518 {
41519     time_t result;

41520     result = time(NULL);
41521     printf("%s%ju secs since the Epoch\n",
41522           asctime(localtime(&result)),
41523           (uintmax_t)result);
41524     return(0);
41525 }
```

41526 This example writes the current time to *stdout* in a form like this:

```
41527 Wed Jun 26 10:32:15 1996
41528 835810335 secs since the Epoch
```

41529 **Getting the Modification Time for a File**

41530 The following example prints the last data modification timestamp in the local timezone for a
 41531 given file.

```
41532 #include <stdio.h>
41533 #include <time.h>
41534 #include <sys/stat.h>

41535 int
41536 print_file_time(const char *pathname)
41537 {
41538     struct stat statbuf;
41539     struct tm *tm;
41540     char timestr[BUFSIZ];

41541     if(stat(pathname, &statbuf) == -1)
41542         return -1;
41543     if((tm = localtime(&statbuf.st_mtime)) == NULL)
41544         return -1;
41545     if(strftime(timestr, sizeof(timestr), "%Y-%m-%d %H:%M:%S", tm) == 0)
41546         return -1;
41547     printf("%s: %s.%09ld\n", pathname, timestr, statbuf.st_mtim.tv_nsec);
41548     return 0;
41549 }
```

Timing an Event

The following example gets the current time, converts it to a string using *localtime()* and *asctime()*, and prints it to standard output using *fputs()*. It then prints the number of minutes to an event being timed.

```
#include <time.h>
#include <stdio.h>
...
time_t now;
int minutes_to_event;
...
time(&now);
printf("The time is ");
fputs(asctime(localtime(&now)), stdout);
printf("There are still %d minutes to the event.\n",
       minutes_to_event);
...
```

APPLICATION USAGE

The *localtime_r()* function is thread-safe and returns values in a user-supplied buffer instead of possibly using a static data area that may be overwritten by each call.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

asctime(), *clock()*, *ctime()*, *difftime()*, *getdate()*, *gmtime()*, *mktime()*, *strftime()*, *strptime()*, *time()*, *tzset()*, *utime()*

XBD Section 4.15 (on page 113), [<time.h>](#)

CHANGE HISTORY

First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 5

A note indicating that the *localtime()* function need not be reentrant is added to the DESCRIPTION.

The *localtime_r()* function is included for alignment with the POSIX Threads Extension.

Issue 6

The *localtime_r()* function is marked as part of the Thread-Safe Functions option.

Extensions beyond the ISO C standard are marked.

The APPLICATION USAGE section is updated to include a note on the thread-safe function and its avoidance of possibly using a static data area.

The **restrict** keyword is added to the *localtime_r()* prototype for alignment with the ISO/IEC 9899:1999 standard.

Examples are added.

IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/32 is applied, adding the [EOVERFLOW] error.

- 41593 IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/55 is applied, updating the error handling
41594 for *localtime_r()*.
- 41595 IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/56 is applied, adding a requirement that if
41596 *localtime_r()* does not set the *tzname* variable, it shall not set the *daylight* or *timezone* variables. On
41597 systems supporting XSI, the *daylight*, *timezone*, and *tzname* variables should all be set to provide
41598 information for the same timezone. This updates the description of *localtime_r()* to mention
41599 *daylight* and *timezone* as well as *tzname*. The SEE ALSO section is updated.
- 41600 **Issue 7**
- 41601 Austin Group Interpretation 1003.1-2001 #156 is applied.
- 41602 The *localtime_r()* function is moved from the Thread-Safe Functions option to the Base.
- 41603 Changes are made to the EXAMPLES section related to support for finegrained timestamps.
- 41604 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0363 [291] is applied. +

NAME

lockf — record locking on files

SYNOPSIS

```
#include <unistd.h>

int lockf(int fildest, int function, off_t size);
```

DESCRIPTION

The *lockf()* function shall lock sections of a file with advisory-mode locks. Calls to *lockf()* from threads in other processes which attempt to lock the locked file section shall either return an error value or block until the section becomes unlocked. All the locks for a process are removed when the process terminates. Record locking with *lockf()* shall be supported for regular files and may be supported for other files.

The *fildest* argument is an open file descriptor. To establish a lock with this function, the file descriptor shall be opened with write-only permission (O_WRONLY) or with read/write permission (O_RDWR).

The *function* argument is a control value which specifies the action to be taken. The permissible values for *function* are defined in **<unistd.h>** as follows:

Function	Description
F_ULOCK	Unlock locked sections.
F_LOCK	Lock a section for exclusive use.
F_TLOCK	Test and lock a section for exclusive use.
F_TEST	Test a section for locks by other processes.

F_TEST shall detect if a lock by another process is present on the specified section.

F_LOCK and F_TLOCK shall both lock a section of a file if the section is available.

F_ULOCK shall remove locks from a section of the file.

The *size* argument is the number of contiguous bytes to be locked or unlocked. The section to be locked or unlocked starts at the current offset in the file and extends forward for a positive size or backward for a negative size (the preceding bytes up to but not including the current offset). If *size* is 0, the section from the current offset through the largest possible file offset shall be locked (that is, from the current offset through the present or any future end-of-file). An area need not be allocated to the file to be locked because locks may exist past the end-of-file.

The sections locked with F_LOCK or F_TLOCK may, in whole or in part, contain or be contained by a previously locked section for the same process. When this occurs, or if adjacent locked sections would occur, the sections shall be combined into a single locked section. If the request would cause the number of locks to exceed a system-imposed limit, the request shall fail.

F_LOCK and F_TLOCK requests differ only by the action taken if the section is not available. F_LOCK shall block the calling thread until the section is available. F_TLOCK shall cause the function to fail if the section is already locked by another process.

File locks shall be released on first close by the locking process of any file descriptor for the file.

F_ULOCK requests may release (wholly or in part) one or more locked sections controlled by the process. Locked sections shall be unlocked starting at the current file offset through *size* bytes or to the end-of-file if *size* is (off_t)0. When all of a locked section is not released (that is, when the beginning or end of the area to be unlocked falls within a locked section), the remaining portions of that section shall remain locked by the process. Releasing the center portion of a locked section shall cause the remaining locked beginning and end portions to become two separate

locked sections. If the request would cause the number of locks in the system to exceed a system-imposed limit, the request shall fail.

A potential for deadlock occurs if the threads of a process controlling a locked section are blocked by accessing a locked section of another process. If the system detects that deadlock would occur, *lockf()* shall fail with an [EDEADLK] error.

The interaction between *fcntl()* and *lockf()* locks is unspecified.

Blocking on a section shall be interrupted by any signal.

An F_ULOCK request in which *size* is non-zero and the offset of the last byte of the requested section is the maximum value for an object of type **off_t**, when the process has an existing lock in which *size* is 0 and which includes the last byte of the requested section, shall be treated as a request to unlock from the start of the requested section with a size equal to 0. Otherwise, an F_ULOCK request shall attempt to unlock only the requested section.

Attempting to lock a section of a file that is associated with a buffered stream produces unspecified results.

RETURN VALUE

Upon successful completion, *lockf()* shall return 0. Otherwise, it shall return -1, set *errno* to indicate an error, and existing locks shall not be changed.

ERRORS

The *lockf()* function shall fail if:

[EBADF] The *fildes* argument is not a valid open file descriptor; or *function* is F_LOCK or F_TLOCK and *fildes* is not a valid file descriptor open for writing.

[EACCES] or [EAGAIN]

The *function* argument is F_TLOCK or F_TEST and the section is already locked by another process.

[EDEADLK] The *function* argument is F_LOCK and a deadlock is detected.

[EINTR] A signal was caught during execution of the function.

[EINVAL] The *function* argument is not one of F_LOCK, F_TLOCK, F_TEST, or F_ULOCK; or *size* plus the current file offset is less than 0.

[EOVERFLOW] The offset of the first, or if *size* is not 0 then the last, byte in the requested section cannot be represented correctly in an object of type **off_t**.

The *lockf()* function may fail if:

[EAGAIN] The *function* argument is F_LOCK or F_TLOCK and the file is mapped with *mmap()*.

[EDEADLK] or [ENOLCK]

The *function* argument is F_LOCK, F_TLOCK, or F_ULOCK, and the request would cause the number of locks to exceed a system-imposed limit.

[EOPNOTSUPP] or [EINVAL]

The implementation does not support the locking of files of the type indicated by the *fildes* argument.

41688 **EXAMPLES**41689 **Locking a Portion of a File**

41690 In the following example, a file named `/home/cnd/mod1` is being modified. Other processes that
41691 use locking are prevented from changing it during this process. Only the first 10 000 bytes are
41692 locked, and the lock call fails if another process has any part of this area locked already.

```
41693 #include <fcntl.h>
41694 #include <unistd.h>
41695
41696 int fildes;
41697 int status;
41698 ...
41699 fildes = open("/home/cnd/mod1", O_RDWR);
41700 status = lockf(fildes, F_TLOCK, (off_t)10000);
```

41700 **APPLICATION USAGE**

41701 Record-locking should not be used in combination with the *fopen()*, *fread()*, *fwrite()*, and other
41702 *stdio* functions. Instead, the more primitive, non-buffered functions (such as *open()*) should be
41703 used. Unexpected results may occur in processes that do buffering in the user address space. The
41704 process may later read/write data which is/was locked. The *stdio* functions are the most
41705 common source of unexpected buffering.

41706 The *alarm()* function may be used to provide a timeout facility in applications requiring it.

41707 **RATIONALE**

41708 None.

41709 **FUTURE DIRECTIONS**

41710 None.

41711 **SEE ALSO**

41712 *alarm()*, *chmod()*, *close()*, *creat()*, *fcntl()*, *fopen()*, *mmap()*, *open()*, *read()*, *write()*

41713 XBD **<unistd.h>**

41714 **CHANGE HISTORY**

41715 First released in Issue 4, Version 2.

41716 **Issue 5**

41717 Moved from X/OPEN UNIX extension to BASE.

41718 Large File Summit extensions are added. In particular, the description of [EINVAL] is clarified
41719 and moved from optional to mandatory status.

41720 A note is added to the DESCRIPTION indicating the effects of attempting to lock a section of a
41721 file that is associated with a buffered stream.

41722 **Issue 6**

41723 The normative text is updated to avoid use of the term “must” for application requirements.

41724 **Issue 7**

41725 Austin Group Interpretation 1003.1-2001 #054 is applied, updating the DESCRIPTION.

41726 **NAME**41727 `log, logf, logl` — natural logarithm function41728 **SYNOPSIS**

```
41729 #include <math.h>
41730 double log(double x);
41731 float logf(float x);
41732 long double logl(long double x);
```

41733 **DESCRIPTION**

41734 CX The functionality described on this reference page is aligned with the ISO C standard. Any
 41735 conflict between the requirements described here and the ISO C standard is unintentional. This
 41736 volume of POSIX.1-2008 defers to the ISO C standard.

41737 These functions shall compute the natural logarithm of their argument x , $\log_e(x)$.

41738 An application wishing to check for error situations should set *errno* to zero and call
 41739 *feclearexcept*(FE_ALL_EXCEPT) before calling these functions. On return, if *errno* is non-zero or
 41740 *fetestexcept*(FE_INVALID | FE_DIVBYZERO | FE_OVERFLOW | FE_UNDERFLOW) is non-
 41741 zero, an error has occurred.

41742 **RETURN VALUE**

41743 Upon successful completion, these functions shall return the natural logarithm of x .

41744 If x is ± 0 , a pole error shall occur and *log()*, *logf()*, and *logl()* shall return `-HUGE_VAL`,
 41745 `-HUGE_VALF`, and `-HUGE_VALL`, respectively.

41746 MX For finite values of x that are less than 0, or if x is `-Inf`, a domain error shall occur, and either a
 41747 NaN (if supported), or an implementation-defined value shall be returned.

41748 MX If x is NaN, a NaN shall be returned.

41749 If x is 1, `+0` shall be returned.

41750 If x is `+Inf`, x shall be returned.

41751 **ERRORS**

41752 These functions shall fail if:

41753	MX	Domain Error	The finite value of x is negative, or x is <code>-Inf</code> .
41754			If the integer expression <i>(math_errhandling & MATH_ERRNO)</i> is non-zero,
41755			then <i>errno</i> shall be set to [EDOM]. If the integer expression <i>(math_errhandling</i>
41756			<i>& MATH_ERREXCEPT)</i> is non-zero, then the invalid floating-point exception
41757			shall be raised.

41758		Pole Error	The value of x is zero.

41759		If the integer expression <i>(math_errhandling & MATH_ERRNO)</i> is non-zero,
41760		then <i>errno</i> shall be set to [ERANGE]. If the integer expression
41761		<i>(math_errhandling & MATH_ERREXCEPT)</i> is non-zero, then the divide-by-zero
41762		floating-point exception shall be raised.

41763 **EXAMPLES**

41764 None.

41765 **APPLICATION USAGE**

41766 On error, the expressions (*math_errhandling* & MATH_ERRNO) and (*math_errhandling* &
41767 MATH_ERREXCEPT) are independent of each other, but at least one of them must be non-zero.

41768 **RATIONALE**

41769 None.

41770 **FUTURE DIRECTIONS**

41771 None.

41772 **SEE ALSO**41773 *exp()*, *feclearexcept()*, *fetestexcept()*, *isnan()*, *log10()*, *log1p()*

41774 XBD Section 4.19 (on page 116), <math.h>

41775 **CHANGE HISTORY**

41776 First released in Issue 1. Derived from Issue 1 of the SVID.

41777 **Issue 5**

41778 The DESCRIPTION is updated to indicate how an application should check for an error. This
41779 text was previously published in the APPLICATION USAGE section.

41780 **Issue 6**

41781 The normative text is updated to avoid use of the term “must” for application requirements.

41782 The *logf()* and *logl()* functions are added for alignment with the ISO/IEC 9899:1999 standard.

41783 The DESCRIPTION, RETURN VALUE, ERRORS, and APPLICATION USAGE sections are
41784 revised to align with the ISO/IEC 9899:1999 standard.

41785 IEC 60559:1989 standard floating-point extensions over the ISO/IEC 9899:1999 standard are
41786 marked.

41787 **NAME**

41788 log10, log10f, log10l — base 10 logarithm function

41789 **SYNOPSIS**

```
41790       #include <math.h>
41791       double log10(double x);
41792       float log10f(float x);
41793       long double log10l(long double x);
```

41794 **DESCRIPTION**

41795 CX The functionality described on this reference page is aligned with the ISO C standard. Any
 41796 conflict between the requirements described here and the ISO C standard is unintentional. This
 41797 volume of POSIX.1-2008 defers to the ISO C standard.

41798 These functions shall compute the base 10 logarithm of their argument x , $\log_{10}(x)$.

41799 An application wishing to check for error situations should set *errno* to zero and call
 41800 *feclearexcept*(FE_ALL_EXCEPT) before calling these functions. On return, if *errno* is non-zero or
 41801 *fetestexcept*(FE_INVALID | FE_DIVBYZERO | FE_OVERFLOW | FE_UNDERFLOW) is non-
 41802 zero, an error has occurred.

41803 **RETURN VALUE**

41804 Upon successful completion, these functions shall return the base 10 logarithm of x .

41805 If x is ± 0 , a pole error shall occur and *log10()*, *log10f()*, and *log10l()* shall return $-\text{HUGE\_VAL}$,
 41806 $-\text{HUGE\_VALF}$, and $-\text{HUGE\_VALL}$, respectively.

41807 MX For finite values of x that are less than 0, or if x is $-\text{Inf}$, a domain error shall occur, and either a
 41808 NaN (if supported), or an implementation-defined value shall be returned.

41809 MX If x is NaN, a NaN shall be returned.

41810 If x is 1, +0 shall be returned.

41811 If x is $+\text{Inf}$, $+\text{Inf}$ shall be returned.

41812 **ERRORS**

41813 These functions shall fail if:

41814	MX	Domain Error	The finite value of x is negative, or x is $-\text{Inf}$.
41815			If the integer expression (<i>math_errhandling</i> & MATH_ERRNO) is non-zero,
41816			then <i>errno</i> shall be set to [EDOM]. If the integer expression (<i>math_errhandling</i>
41817			& MATH_ERREXCEPT) is non-zero, then the invalid floating-point exception
41818			shall be raised.

41819		Pole Error	The value of x is zero.

41820		If the integer expression (<i>math_errhandling</i> & MATH_ERRNO) is non-zero,
41821		then <i>errno</i> shall be set to [ERANGE]. If the integer expression
41822		(<i>math_errhandling</i> & MATH_ERREXCEPT) is non-zero, then the divide-by-zero
41823		floating-point exception shall be raised.

41824 **EXAMPLES**

41825 None.

41826 **APPLICATION USAGE**

41827 On error, the expressions (*math_errhandling* & MATH_ERRNO) and (*math_errhandling* &
41828 MATH_ERREXCEPT) are independent of each other, but at least one of them must be non-zero.

41829 **RATIONALE**

41830 None.

41831 **FUTURE DIRECTIONS**

41832 None.

41833 **SEE ALSO**41834 *feclearexcept()*, *fetestexcept()*, *isnan()*, *log()*, *pow()*

41835 XBD Section 4.19 (on page 116), <math.h>

41836 **CHANGE HISTORY**

41837 First released in Issue 1. Derived from Issue 1 of the SVID.

41838 **Issue 5**

41839 The DESCRIPTION is updated to indicate how an application should check for an error. This
41840 text was previously published in the APPLICATION USAGE section.

41841 **Issue 6**

41842 The normative text is updated to avoid use of the term “must” for application requirements.

41843 The *log10f()* and *log10l()* functions are added for alignment with the ISO/IEC 9899:1999
41844 standard.

41845 The DESCRIPTION, RETURN VALUE, ERRORS, and APPLICATION USAGE sections are
41846 revised to align with the ISO/IEC 9899:1999 standard.

41847 IEC 60559:1989 standard floating-point extensions over the ISO/IEC 9899:1999 standard are
41848 marked.

41849 **NAME**41850 `log1p`, `log1pf`, `log1pl` — compute a natural logarithm41851 **SYNOPSIS**

```
41852     #include <math.h>

41853     double log1p(double x);
41854     float log1pf(float x);
41855     long double log1pl(long double x);
```

41856 **DESCRIPTION**

41857 **CX** The functionality described on this reference page is aligned with the ISO C standard. Any
 41858 conflict between the requirements described here and the ISO C standard is unintentional. This
 41859 volume of POSIX.1-2008 defers to the ISO C standard.

41860 These functions shall compute $\log_e(1.0 + x)$.

41861 An application wishing to check for error situations should set *errno* to zero and call
 41862 *feclearexcept*(FE_ALL_EXCEPT) before calling these functions. On return, if *errno* is non-zero or
 41863 *fetestexcept*(FE_INVALID | FE_DIVBYZERO | FE_OVERFLOW | FE_UNDERFLOW) is non-
 41864 zero, an error has occurred.

41865 **RETURN VALUE**

41866 Upon successful completion, these functions shall return the natural logarithm of $1.0 + x$.

41867 If x is -1 , a pole error shall occur and *log1p()*, *log1pf()*, and *log1pl()* shall return $-\text{HUGE\_VAL}$,
 41868 $-\text{HUGE\_VALF}$, and $-\text{HUGE\_VALL}$, respectively.

41869 **MX** For finite values of x that are less than -1 , or if x is $-\text{Inf}$, a domain error shall occur, and either a
 41870 NaN (if supported), or an implementation-defined value shall be returned.

41871 **MX** If x is NaN, a NaN shall be returned.

41872 If x is ± 0 , or $+\text{Inf}$, x shall be returned.

41873 If x is subnormal, a range error may occur

41874 **MX** and x should be returned.

41875 **MX** If x is not returned, *log1p()*, *log1pf()*, and *log1pl()* shall return an implementation-defined value
 41876 no greater in magnitude than DBL_MIN, FLT_MIN, and LDBL_MIN, respectively.

41877 **ERRORS**

41878 These functions shall fail if:

41879 **MX** Domain Error The finite value of x is less than -1 , or x is $-\text{Inf}$.

41880 If the integer expression (*math_errhandling* & MATH_ERRNO) is non-zero,
 41881 then *errno* shall be set to [EDOM]. If the integer expression (*math_errhandling*
 41882 & MATH_ERREXCEPT) is non-zero, then the invalid floating-point exception
 41883 shall be raised.

41884 Pole Error The value of x is -1 .

41885 If the integer expression (*math_errhandling* & MATH_ERRNO) is non-zero,
 41886 then *errno* shall be set to [ERANGE]. If the integer expression
 41887 (*math_errhandling* & MATH_ERREXCEPT) is non-zero, then the divide-by-zero
 41888 floating-point exception shall be raised.

41889 These functions may fail if:

41890	MX	Range Error	The value of x is subnormal.
41891			If the integer expression $(\text{math\_errhandling} \ \& \ \text{MATH\_ERRNO})$ is non-zero,
41892			then <i>errno</i> shall be set to [ERANGE]. If the integer expression
41893			$(\text{math\_errhandling} \ \& \ \text{MATH\_ERREXCEPT})$ is non-zero, then the underflow
41894			floating-point exception shall be raised.

41895 EXAMPLES

41896 None.

41897 APPLICATION USAGE

41898 On error, the expressions $(\text{math\_errhandling} \ \& \ \text{MATH\_ERRNO})$ and $(\text{math\_errhandling} \ \& \ \text{MATH\_ERREXCEPT})$ are independent of each other, but at least one of them must be non-zero.

41900 RATIONALE

41901 None.

41902 FUTURE DIRECTIONS

41903 None.

41904 SEE ALSO

41905 *feclearexcept()*, *fetestexcept()*, *log()*

41906 XBD [Section 4.19](#) (on page 116), `<math.h>`

41907 CHANGE HISTORY

41908 First released in Issue 4, Version 2.

41909 Issue 5

41910 Moved from X/OPEN UNIX extension to BASE.

41911 Issue 6

41912 The normative text is updated to avoid use of the term “must” for application requirements.

41913 The *log1p()* function is no longer marked as an extension.

41914 The *log1pf()* and *log1pl()* functions are added for alignment with the ISO/IEC 9899:1999 standard.

41916 The DESCRIPTION, RETURN VALUE, ERRORS, and APPLICATION USAGE sections are revised to align with the ISO/IEC 9899:1999 standard.

41918 IEC 60559:1989 standard floating-point extensions over the ISO/IEC 9899:1999 standard are marked.

41920 Issue 7

41921 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0364 [68] is applied.

+

41922 **NAME**

41923 log2, log2f, log2l — compute base 2 logarithm functions

41924 **SYNOPSIS**

```
41925 #include <math.h>
41926 double log2(double x);
41927 float log2f(float x);
41928 long double log2l(long double x);
```

41929 **DESCRIPTION**

41930 CX The functionality described on this reference page is aligned with the ISO C standard. Any
 41931 conflict between the requirements described here and the ISO C standard is unintentional. This
 41932 volume of POSIX.1-2008 defers to the ISO C standard.

41933 These functions shall compute the base 2 logarithm of their argument x , $\log_2(x)$.

41934 An application wishing to check for error situations should set *errno* to zero and call
 41935 *feclearexcept*(FE_ALL_EXCEPT) before calling these functions. On return, if *errno* is non-zero or
 41936 *fetestexcept*(FE_INVALID | FE_DIVBYZERO | FE_OVERFLOW | FE_UNDERFLOW) is non-
 41937 zero, an error has occurred.

41938 **RETURN VALUE**

41939 Upon successful completion, these functions shall return the base 2 logarithm of x .

41940 If x is ± 0 , a pole error shall occur and *log2()*, *log2f()*, and *log2l()* shall return $-\text{HUGE\_VAL}$,
 41941 $-\text{HUGE\_VALF}$, and $-\text{HUGE\_VALL}$, respectively.

41942 MX For finite values of x that are less than 0, or if x is $-\text{Inf}$, a domain error shall occur, and either a
 41943 NaN (if supported), or an implementation-defined value shall be returned.

41944 MX If x is NaN, a NaN shall be returned.

41945 If x is 1, +0 shall be returned.

41946 If x is $+\text{Inf}$, x shall be returned.

41947 **ERRORS**

41948 These functions shall fail if:

41949 MX Domain Error The finite value of x is less than zero, or x is $-\text{Inf}$.

41950 If the integer expression (*math_errhandling* & MATH_ERRNO) is non-zero,
 41951 then *errno* shall be set to [EDOM]. If the integer expression (*math_errhandling*
 41952 & MATH_ERREXCEPT) is non-zero, then the invalid floating-point exception
 41953 shall be raised.

41954 Pole Error The value of x is zero.

41955 If the integer expression (*math_errhandling* & MATH_ERRNO) is non-zero,
 41956 then *errno* shall be set to [ERANGE]. If the integer expression
 41957 (*math_errhandling* & MATH_ERREXCEPT) is non-zero, then the divide-by-zero
 41958 floating-point exception shall be raised.

41959 EXAMPLES

41960 None.

41961 APPLICATION USAGE

41962 On error, the expressions (*math_errhandling* & MATH_ERRNO) and (*math_errhandling* &
41963 MATH_ERREXCEPT) are independent of each other, but at least one of them must be non-zero.

41964 RATIONALE

41965 None.

41966 FUTURE DIRECTIONS

41967 None.

41968 SEE ALSO

41969 *feclearexcept()*, *fetestexcept()*, *log()*

41970 XBD [Section 4.19](#) (on page 116), [<math.h>](#)

41971 CHANGE HISTORY

41972 First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

41973 **NAME**41974 `logb`, `logbf`, `logbl` — radix-independent exponent41975 **SYNOPSIS**

```
41976     #include <math.h>
41977
41977     double logb(double x);
41978     float logbf(float x);
41979     long double logbl(long double x);
```

41980 **DESCRIPTION**

41981 CX The functionality described on this reference page is aligned with the ISO C standard. Any
 41982 conflict between the requirements described here and the ISO C standard is unintentional. This
 41983 volume of POSIX.1-2008 defers to the ISO C standard.

41984 These functions shall compute the exponent of x , which is the integral part of $\log_r |x|$, as a
 41985 signed floating-point value, for non-zero x , where r is the radix of the machine's floating-point
 41986 arithmetic, which is the value of `FLT_RADIX` defined in the `<float.h>` header.

41987 If x is subnormal it is treated as though it were normalized; thus for finite positive x :

41988 $1 \leq x * \text{FLT\_RADIX}^{-\log_b(x)} < \text{FLT\_RADIX}$

41989 An application wishing to check for error situations should set `errno` to zero and call
 41990 `feclearexcept(FE_ALL_EXCEPT)` before calling these functions. On return, if `errno` is non-zero or
 41991 `fetestexcept(FE_INVALID | FE_DIVBYZERO | FE_OVERFLOW | FE_UNDERFLOW)` is non-
 41992 zero, an error has occurred.

41993 **RETURN VALUE**

41994 Upon successful completion, these functions shall return the exponent of x .

41995 If x is ± 0 , `logb()`, `logbf()`, and `logbl()` shall return `-HUGE_VAL`, `-HUGE_VALF`, and
 41996 `-HUGE_VALL`, respectively.

41997 MX On systems that support the IEC 60559 Floating-Point option, a pole error shall occur;

41998 CX otherwise, a pole error may occur.

41999 MX If x is NaN, a NaN shall be returned.

42000 MX If x is $\pm\text{Inf}$, $+\text{Inf}$ shall be returned.

42001 **ERRORS**

42002 These functions shall fail if:

42003 MX **Pole Error** The value of x is ± 0 .

42004 If the integer expression `(math_errhandling & MATH_ERRNO)` is non-zero,
 42005 then `errno` shall be set to `[ERANGE]`. If the integer expression
 42006 `(math_errhandling & MATH_ERREXCEPT)` is non-zero, then the divide-by-zero
 42007 floating-point exception shall be raised.

42008 These functions may fail if:

42009 **Pole Error** The value of x is 0.

42010 If the integer expression `(math_errhandling & MATH_ERRNO)` is non-zero,
 42011 then `errno` shall be set to `[ERANGE]`. If the integer expression
 42012 `(math_errhandling & MATH_ERREXCEPT)` is non-zero, then the divide-by-zero
 42013 floating-point exception shall be raised.

42014 **EXAMPLES**

42015 None.

42016 **APPLICATION USAGE**

42017 On error, the expressions (*math_errhandling* & MATH_ERRNO) and (*math_errhandling* &
42018 MATH_ERREXCEPT) are independent of each other, but at least one of them must be non-zero.

42019 **RATIONALE**

42020 None.

42021 **FUTURE DIRECTIONS**

42022 None.

42023 **SEE ALSO**42024 *feclearexcept()*, *fetestexcept()*, *ilogb()*, *scalbln()*

42025 XBD Section 4.19 (on page 116), <float.h>, <math.h>

42026 **CHANGE HISTORY**

42027 First released in Issue 4, Version 2.

42028 **Issue 5**

42029 Moved from X/OPEN UNIX extension to BASE.

42030 **Issue 6**42031 The *logb()* function is no longer marked as an extension.42032 The *logbf()* and *logbl()* functions are added for alignment with the ISO/IEC 9899:1999 standard.

42033 The DESCRIPTION, RETURN VALUE, ERRORS, and APPLICATION USAGE sections are
42034 revised to align with the ISO/IEC 9899:1999 standard.

42035 IEC 60559:1989 standard floating-point extensions over the ISO/IEC 9899:1999 standard are
42036 marked.

42037 **Issue 7**

42038 ISO/IEC 9899:1999 standard, Technical Corrigendum 2 #50 (SD5-XSH-ERN-76) is applied.

42039 **NAME**

42040 logf, logl — natural logarithm function

42041 **SYNOPSIS**

42042 #include <math.h>

42043 float logf(float x);

42044 long double logl(long double x);

42045 **DESCRIPTION**42046 Refer to *log()*.

42047 **NAME**

42048 longjmp — non-local goto

42049 **SYNOPSIS**

42050 #include <setjmp.h>

42051 void longjmp(jmp_buf env, int val);

42052 **DESCRIPTION**

42053 CX The functionality described on this reference page is aligned with the ISO C standard. Any
 42054 conflict between the requirements described here and the ISO C standard is unintentional. This
 42055 volume of POSIX.1-2008 defers to the ISO C standard.

42056 The *longjmp()* function shall restore the environment saved by the most recent invocation of
 42057 *setjmp()* in the same process, with the corresponding **jmp_buf** argument. If the most recent
 42058 invocation of *setjmp()* with the corresponding **jmp_buf** occurred in another thread, or if there is
 42059 no such invocation, or if the function containing the invocation of *setjmp()* has terminated
 42060 execution in the interim, or if the invocation of *setjmp()* was within the scope of an identifier
 42061 with variably modified type and execution has left that scope in the interim, the behavior is
 42062 CX undefined. It is unspecified whether *longjmp()* restores the signal mask, leaves the signal mask
 42063 unchanged, or restores it to its value at the time *setjmp()* was called.

42064 All accessible objects have values, and all other components of the abstract machine have state
 42065 (for example, floating-point status flags and open files), as of the time *longjmp()* was called,
 42066 except that the values of objects of automatic storage duration are unspecified if they meet all
 42067 the following conditions:

- 42068 • They are local to the function containing the corresponding *setjmp()* invocation.
- 42069 • They do not have volatile-qualified type.
- 42070 • They are changed between the *setjmp()* invocation and *longjmp()* call.

42071 CX As it bypasses the usual function call and return mechanisms, *longjmp()* shall execute correctly
 42072 in contexts of interrupts, signals, and any of their associated functions. However, if *longjmp()* is
 42073 invoked from a nested signal handler (that is, from a function invoked as a result of a signal
 42074 raised during the handling of another signal), the behavior is undefined.

42075 The effect of a call to *longjmp()* where initialization of the **jmp_buf** structure was not performed
 42076 in the calling thread is undefined.

42077 **RETURN VALUE**

42078 After *longjmp()* is completed, program execution continues as if the corresponding invocation of
 42079 *setjmp()* had just returned the value specified by *val*. The *longjmp()* function shall not cause
 42080 *setjmp()* to return 0; if *val* is 0, *setjmp()* shall return 1.

42081 **ERRORS**

42082 No errors are defined.

42083 **EXAMPLES**

42084 None.

42085 **APPLICATION USAGE**

42086 Applications whose behavior depends on the value of the signal mask should not use *longjmp()*
 42087 and *setjmp()*, since their effect on the signal mask is unspecified, but should instead use the
 42088 *siglongjmp()* and *sigsetjmp()* functions (which can save and restore the signal mask under
 42089 application control).

42090 **RATIONALE**

42091 None.

42092 **FUTURE DIRECTIONS**

42093 None.

42094 **SEE ALSO**42095 *setjmp()*, *sigaction()*, *siglongjmp()*, *sigsetjmp()*

42096 XBD <setjmp.h>

42097 **CHANGE HISTORY**

42098 First released in Issue 1. Derived from Issue 1 of the SVID.

42099 **Issue 5**

42100 The DESCRIPTION is updated for alignment with the POSIX Threads Extension.

42101 **Issue 6**

42102 Extensions beyond the ISO C standard are marked.

42103 The following new requirements on POSIX implementations derive from alignment with the
42104 Single UNIX Specification:

- 42105 • The DESCRIPTION now explicitly makes *longjmp()*'s effect on the signal mask
- 42106 unspecified.

42107 The DESCRIPTION is updated for alignment with the ISO/IEC 9899:1999 standard.

42108 **Issue 7**

42109 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0365 [394] is applied.

+

42110 NAME

42111 lrand48 — generate uniformly distributed pseudo-random non-negative long integers

42112 SYNOPSIS

```
42113 XSI      #include <stdlib.h>  
42114          long lrand48(void);
```

42115 DESCRIPTION

42116 Refer to *drand48()*.

42117 **NAME**

42118 lrint, lrintf, lrintl — round to nearest integer value using current rounding direction

42119 **SYNOPSIS**

```
42120 #include <math.h>
42121 long lrint(double x);
42122 long lrintf(float x);
42123 long lrintl(long double x);
```

42124 **DESCRIPTION**

42125 CX The functionality described on this reference page is aligned with the ISO C standard. Any
 42126 conflict between the requirements described here and the ISO C standard is unintentional. This
 42127 volume of POSIX.1-2008 defers to the ISO C standard.

42128 These functions shall round their argument to the nearest integer value, rounding according to
 42129 the current rounding direction.

42130 An application wishing to check for error situations should set *errno* to zero and call
 42131 *feclearexcept*(FE_ALL_EXCEPT) before calling these functions. On return, if *errno* is non-zero or
 42132 *fetestexcept*(FE_INVALID | FE_DIVBYZERO | FE_OVERFLOW | FE_UNDERFLOW) is non-
 42133 zero, an error has occurred.

42134 **RETURN VALUE**

42135 Upon successful completion, these functions shall return the rounded integer value.

42136 MX If *x* is NaN, a domain error shall occur and an unspecified value is returned.

42137 If *x* is +Inf, a domain error shall occur and an unspecified value is returned.

42138 If *x* is -Inf, a domain error shall occur and an unspecified value is returned.

42139 If the correct value is positive and too large to represent as a **long**, an unspecified value shall be
 42140 MX returned. On systems that support the IEC 60559 Floating-Point option, a domain error shall
 42141 CX occur; otherwise, a domain error may occur.

42142 If the correct value is negative and too large to represent as a **long**, an unspecified value shall be
 42143 MX returned. On systems that support the IEC 60559 Floating-Point option, a domain error shall
 42144 CX occur; otherwise, a domain error may occur.

42145 **ERRORS**

42146 These functions shall fail if:

42147 MX **Domain Error** The *x* argument is NaN or ±Inf, or the correct value is not representable as an
 42148 integer.

42149 If the integer expression (*math_errhandling* & MATH_ERRNO) is non-zero,
 42150 then *errno* shall be set to [EDOM]. If the integer expression (*math_errhandling*
 42151 & MATH_ERREXCEPT) is non-zero, then the invalid floating-point exception
 42152 shall be raised.

42153 These functions may fail if:

42154 **Domain Error** The correct value is not representable as an integer.

42155 If the integer expression (*math_errhandling* & MATH_ERRNO) is non-zero,
 42156 then *errno* shall be set to [EDOM]. If the integer expression (*math_errhandling*
 42157 & MATH_ERREXCEPT) is non-zero, then the invalid floating-point exception
 42158 shall be raised.

42159 EXAMPLES

42160 None.

42161 APPLICATION USAGE

42162 On error, the expressions *(math_errhandling & MATH_ERRNO)* and *(math_errhandling &*
42163 *MATH_ERREXCEPT)* are independent of each other, but at least one of them must be non-zero.

42164 RATIONALE

42165 These functions provide floating-to-integer conversions. They round according to the current
42166 rounding direction. If the rounded value is outside the range of the return type, the numeric
42167 result is unspecified and the invalid floating-point exception is raised. When they raise no other
42168 floating-point exception and the result differs from the argument, they raise the inexact floating-
42169 point exception.

42170 FUTURE DIRECTIONS

42171 None.

42172 SEE ALSO

42173 *feclearexcept()*, *fetestexcept()*, *llrint()*

42174 XBD [Section 4.19](#) (on page 116), [<math.h>](#)

42175 CHANGE HISTORY

42176 First released in Issue 6. Derived from the ISO/IEC 9899: 1999 standard.

42177 Issue 7

42178 ISO/IEC 9899: 1999 standard, Technical Corrigendum 2 #53 (SD5-XSH-ERN-77) is applied.

42179 **NAME**

42180 lround, lroundf, lroundl — round to nearest integer value

42181 **SYNOPSIS**

```
42182 #include <math.h>
42183 long lround(double x);
42184 long lroundf(float x);
42185 long lroundl(long double x);
```

42186 **DESCRIPTION**

42187 CX The functionality described on this reference page is aligned with the ISO C standard. Any
 42188 conflict between the requirements described here and the ISO C standard is unintentional. This
 42189 volume of POSIX.1-2008 defers to the ISO C standard.

42190 These functions shall round their argument to the nearest integer value, rounding halfway cases
 42191 away from zero, regardless of the current rounding direction.

42192 An application wishing to check for error situations should set *errno* to zero and call
 42193 *feclearexcept*(FE_ALL_EXCEPT) before calling these functions. On return, if *errno* is non-zero or
 42194 *fetestexcept*(FE_INVALID | FE_DIVBYZERO | FE_OVERFLOW | FE_UNDERFLOW) is non-
 42195 zero, an error has occurred.

42196 **RETURN VALUE**

42197 Upon successful completion, these functions shall return the rounded integer value.

42198 MX If *x* is NaN, a domain error shall occur and an unspecified value is returned.

42199 If *x* is +Inf, a domain error shall occur and an unspecified value is returned.

42200 If *x* is -Inf, a domain error shall occur and an unspecified value is returned.

42201 If the correct value is positive and too large to represent as a **long**, an unspecified value shall be
 42202 MX returned. On systems that support the IEC 60559 Floating-Point option, a domain shall occur;
 42203 CX otherwise, a domain error may occur.

42204 If the correct value is negative and too large to represent as a **long**, an unspecified value shall be
 42205 MX returned. On systems that support the IEC 60559 Floating-Point option, a domain shall occur;
 42206 CX otherwise, a domain error may occur.

42207 **ERRORS**

42208 These functions shall fail if:

42209 MX **Domain Error** The *x* argument is NaN or ±Inf, or the correct value is not representable as an
 42210 integer.

42211 If the integer expression (*math_errhandling* & MATH_ERRNO) is non-zero,
 42212 then *errno* shall be set to [EDOM]. If the integer expression (*math_errhandling*
 42213 & MATH_ERREXCEPT) is non-zero, then the invalid floating-point exception
 42214 shall be raised.

42215 These functions may fail if:

42216 **Domain Error** The correct value is not representable as an integer.

42217 If the integer expression (*math_errhandling* & MATH_ERRNO) is non-zero,
 42218 then *errno* shall be set to [EDOM]. If the integer expression (*math_errhandling*
 42219 & MATH_ERREXCEPT) is non-zero, then the invalid floating-point exception
 42220 shall be raised.

42221 **EXAMPLES**

42222 None.

42223 **APPLICATION USAGE**

42224 On error, the expressions (*math_errhandling* & MATH_ERRNO) and (*math_errhandling* &
42225 MATH_ERREXCEPT) are independent of each other, but at least one of them must be non-zero.

42226 **RATIONALE**

42227 These functions differ from the *lrint()* functions in the default rounding direction, with the
42228 *lround()* functions rounding halfway cases away from zero and needing not to raise the inexact
42229 floating-point exception for non-integer arguments that round to within the range of the return
42230 type.

42231 **FUTURE DIRECTIONS**

42232 None.

42233 **SEE ALSO**42234 *feclearexcept()*, *fetestexcept()*, *llround()*

42235 XBD Section 4.19 (on page 116), <math.h>

42236 **CHANGE HISTORY**

42237 First released in Issue 6. Derived from the ISO/IEC 9899: 1999 standard.

42238 **Issue 7**

42239 ISO/IEC 9899: 1999 standard, Technical Corrigendum 2 #54 (SD5-XSH-ERN-78) is applied.

NAME

`lsearch`, `lfind` — linear search and update

SYNOPSIS

```
XSI
#include <search.h>

void *lsearch(const void *key, void *base, size_t *nel, size_t width,
              int (*compar)(const void *, const void *));
void *lfind(const void *key, const void *base, size_t *nel,
            size_t width, int (*compar)(const void *, const void *));
```

DESCRIPTION

The `lsearch()` function shall linearly search the table and return a pointer into the table for the matching entry. If the entry does not occur, it shall be added at the end of the table. The *key* argument points to the entry to be sought in the table. The *base* argument points to the first element in the table. The *width* argument is the size of an element in bytes. The *nel* argument points to an integer containing the current number of elements in the table. The integer to which *nel* points shall be incremented if the entry is added to the table. The *compar* argument points to a comparison function which the application shall supply (for example, `strcmp()`). It is called with two arguments that point to the elements being compared. The application shall ensure that the function returns 0 if the elements are equal, and non-zero otherwise.

The `lfind()` function shall be equivalent to `lsearch()`, except that if the entry is not found, it is not added to the table. Instead, a null pointer is returned.

RETURN VALUE

If the searched for entry is found, both `lsearch()` and `lfind()` shall return a pointer to it. Otherwise, `lfind()` shall return a null pointer and `lsearch()` shall return a pointer to the newly added element.

Both functions shall return a null pointer in case of error.

ERRORS

No errors are defined.

EXAMPLES**Storing Strings in a Table**

This fragment reads in less than or equal to TABSIZE strings of length less than or equal to ELSIZE and stores them in a table, eliminating duplicates.

```
#include <stdio.h>
#include <string.h>
#include <search.h>

#define TABSIZE 50
#define ELSIZE 120

...
    char line[ELSIZE], tab[TABSIZE][ELSIZE];
    size_t nel = 0;
    ...
    while (fgets(line, ELSIZE, stdin) != NULL && nel < TABSIZE)
        (void) lsearch(line, tab, &nel,
                       ELSIZE, (int (*)(const void *, const void *)) strcmp);
    ...
```

Finding a Matching Entry

The following example finds any line that reads "This is a test.".

```
#include <search.h>
#include <string.h>
...
char line[ELSIZE], tab[TABSIZE][ELSIZE];
size_t nel = 0;
char *findline;
void *entry;

findline = "This is a test.\n";

entry = lfind(findline, tab, &nel, ELSIZE, (
    int (*)(const void *, const void *)) strcmp);
```

APPLICATION USAGE

The comparison function need not compare every byte, so arbitrary data may be contained in the elements in addition to the values being compared.

Undefined results can occur if there is not enough room in the table to add a new item.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

hcreate(), tdelete()

XBD [**<search.h>**](#)

CHANGE HISTORY

First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6

The normative text is updated to avoid use of the term “must” for application requirements.

42311 **NAME**42312 `lseek` — move the read/write file offset42313 **SYNOPSIS**42314 `#include <unistd.h>`42315 `off_t lseek(int fildes, off_t offset, int whence);`42316 **DESCRIPTION**42317 The `lseek()` function shall set the file offset for the open file description associated with the file
42318 descriptor *fildes*, as follows:

- 42319 • If *whence* is `SEEK_SET`, the file offset shall be set to *offset* bytes.
- 42320 • If *whence* is `SEEK_CUR`, the file offset shall be set to its current location plus *offset*.
- 42321 • If *whence* is `SEEK_END`, the file offset shall be set to the size of the file plus *offset*.

42322 The symbolic constants `SEEK_SET`, `SEEK_CUR`, and `SEEK_END` are defined in `<unistd.h>`.42323 The behavior of `lseek()` on devices which are incapable of seeking is implementation-defined.
42324 The value of the file offset associated with such a device is undefined.42325 The `lseek()` function shall allow the file offset to be set beyond the end of the existing data in the
42326 file. If data is later written at this point, subsequent reads of data in the gap shall return bytes
42327 with the value 0 until data is actually written into the gap.42328 The `lseek()` function shall not, by itself, extend the size of a file.42329 SHM If *fildes* refers to a shared memory object, the result of the `lseek()` function is unspecified.42330 TYM If *fildes* refers to a typed memory object, the result of the `lseek()` function is unspecified.42331 **RETURN VALUE**42332 Upon successful completion, the resulting offset, as measured in bytes from the beginning of the
42333 file, shall be returned. Otherwise, `-1` shall be returned, *errno* shall be set to indicate the error, and
42334 the file offset shall remain unchanged.42335 **ERRORS**42336 The `lseek()` function shall fail if:

- 42337 [EBADF] The *fildes* argument is not an open file descriptor.
- 42338 [EINVAL] The *whence* argument is not a proper value, or the resulting file offset would
42339 be negative for a regular file, block special file, or directory.
- 42340 [EOVERFLOW] The resulting file offset would be a value which cannot be represented
42341 correctly in an object of type `off_t`.
- 42342 [ESPIPE] The *fildes* argument is associated with a pipe, FIFO, or socket.

42343 **EXAMPLES**

42344 None.

42345 **APPLICATION USAGE**

42346 None.

42347 **RATIONALE**42348 The ISO C standard includes the functions `fgetpos()` and `fsetpos()`, which work on very large files
42349 by use of a special positioning type.42350 Although `lseek()` may position the file offset beyond the end of the file, this function does not
42351 itself extend the size of the file. While the only function in POSIX.1-2008 that may directly extend

the size of the file is *write()*, *truncate()*, and *ftruncate()*, several functions originally derived from the ISO C standard, such as *fwrite()*, *fprintf()*, and so on, may do so (by causing calls on *write()*).

An invalid file offset that would cause [EINVAL] to be returned may be both implementation-defined and device-dependent (for example, memory may have few invalid values). A negative file offset may be valid for some devices in some implementations.

The POSIX.1-1990 standard did not specifically prohibit *lseek()* from returning a negative offset. Therefore, an application was required to clear *errno* prior to the call and check *errno* upon return to determine whether a return value of (**off_t**)-1 is a negative offset or an indication of an error condition. The standard developers did not wish to require this action on the part of a conforming application, and chose to require that *errno* be set to [EINVAL] when the resulting file offset would be negative for a regular file, block special file, or directory.

FUTURE DIRECTIONS

None.

SEE ALSO

open()

XBD **<sys/types.h>**, **<unistd.h>**

CHANGE HISTORY

First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 5

The DESCRIPTION is updated for alignment with the POSIX Realtime Extension.

Large File Summit extensions are added.

Issue 6

In the SYNOPSIS, the optional include of the **<sys/types.h>** header is removed.

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The requirement to include **<sys/types.h>** has been removed. Although **<sys/types.h>** was required for conforming implementations of previous POSIX specifications, it was not required for UNIX applications.
- The [EOVERFLOW] error condition is added. This change is to support large files.

An additional [ESPIPE] error condition is added for sockets.

The DESCRIPTION is updated for alignment with IEEE Std 1003.1j-2000 by specifying that *lseek()* results are unspecified for typed memory objects.

Issue 7

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0366 [421] is applied.

+

42386 **NAME**

42387 lstat — get file status

42388 **SYNOPSIS**

42389 #include <sys/stat.h>

42390 int lstat(const char *restrict *path*, struct stat *restrict *buf*);42391 **DESCRIPTION**42392 Refer to *fstatat()*.

NAME

malloc — a memory allocator

SYNOPSIS

```
#include <stdlib.h>
```

```
void *malloc(size_t size);
```

DESCRIPTION

CX The functionality described on this reference page is aligned with the ISO C standard. Any conflict between the requirements described here and the ISO C standard is unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

The *malloc()* function shall allocate unused space for an object whose size in bytes is specified by *size* and whose value is unspecified.

The order and contiguity of storage allocated by successive calls to *malloc()* is unspecified. The pointer returned if the allocation succeeds shall be suitably aligned so that it may be assigned to a pointer to any type of object and then used to access such an object in the space allocated (until the space is explicitly freed or reallocated). Each such allocation shall yield a pointer to an object disjoint from any other object. The pointer returned points to the start (lowest byte address) of the allocated space. If the space cannot be allocated, a null pointer shall be returned. If the size of the space requested is 0, the behavior is implementation-defined: the value returned shall be either a null pointer or a unique pointer.

RETURN VALUE

Upon successful completion with *size* not equal to 0, *malloc()* shall return a pointer to the allocated space. If *size* is 0, either a null pointer or a unique pointer that can be successfully passed to *free()* shall be returned. Otherwise, it shall return a null pointer and set *errno* to indicate the error.

ERRORS

The *malloc()* function shall fail if:

CX [ENOMEM] Insufficient storage space is available.

EXAMPLES

None.

APPLICATION USAGE

None.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

calloc(), *free()*, *getrlimit()*, *posix_memalign()*, *realloc()*

XBD <stdlib.h>

CHANGE HISTORY

First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6

Extensions beyond the ISO C standard are marked.

42435
42436
42437
42438

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- In the RETURN VALUE section, the requirement to set *errno* to indicate an error is added.
- The [ENOMEM] error condition is added.

42439 **NAME**

42440 mblen — get number of bytes in a character

42441 **SYNOPSIS**

42442 #include <stdlib.h>

42443 int mblen(const char *s, size_t n);

42444 **DESCRIPTION**

42445 CX The functionality described on this reference page is aligned with the ISO C standard. Any
 42446 conflict between the requirements described here and the ISO C standard is unintentional. This
 42447 volume of POSIX.1-2008 defers to the ISO C standard.

42448 If *s* is not a null pointer, *mblen()* shall determine the number of bytes constituting the character
 42449 pointed to by *s*. Except that the shift state of *mbtowc()* is not affected, it shall be equivalent to:

42450 mbtowc((wchar_t *)0, s, n);

42451 The implementation shall behave as if no function defined in this volume of POSIX.1-2008 calls
 42452 *mblen()*.

42453 The behavior of this function is affected by the *LC_CTYPE* category of the current locale. For a
 42454 state-dependent encoding, this function shall be placed into its initial state by a call for which its
 42455 character pointer argument, *s*, is a null pointer. Subsequent calls with *s* as other than a null
 42456 pointer shall cause the internal state of the function to be altered as necessary. A call with *s* as a
 42457 null pointer shall cause this function to return a non-zero value if encodings have state
 42458 dependency, and 0 otherwise. If the implementation employs special bytes to change the shift
 42459 state, these bytes shall not produce separate wide-character codes, but shall be grouped with an
 42460 adjacent character. Changing the *LC_CTYPE* category causes the shift state of this function to be
 42461 unspecified.

42462 CX The *mblen()* function need not be thread-safe.

+

42463 **RETURN VALUE**

42464 If *s* is a null pointer, *mblen()* shall return a non-zero or 0 value, if character encodings,
 42465 respectively, do or do not have state-dependent encodings. If *s* is not a null pointer, *mblen()* shall
 42466 either return 0 (if *s* points to the null byte), or return the number of bytes that constitute the
 42467 character (if the next *n* or fewer bytes form a valid character), or return -1 (if they do not form a
 42468 valid character) and may set *errno* to indicate the error. In no case shall the value returned be
 42469 greater than *n* or the value of the {MB_CUR_MAX} macro.

42470 **ERRORS**42471 The *mblen()* function may fail if:

42472 XSI [EILSEQ] An invalid character sequence is detected.

42473 **EXAMPLES**

42474 None.

42475 **APPLICATION USAGE**

42476 None.

42477 **RATIONALE**

42478 None.

42479 **FUTURE DIRECTIONS**

42480 None.

42481 **SEE ALSO**

42482 *mbtowc()*, *mbstowcs()*, *wctomb()*, *wcstombs()*

42483 XBD **<stdlib.h>**

42484 **CHANGE HISTORY**

42485 First released in Issue 4. Aligned with the ISO C standard.

42486 **Issue 7**

42487 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0367 [109] is applied.

+

42488 **NAME**

42489 mbrlen — get number of bytes in a character (restartable)

42490 **SYNOPSIS**

42491 #include <wchar.h>

42492 size_t mbrlen(const char *restrict *s*, size_t *n*,
42493 mbstate_t *restrict *ps*);

42494 **DESCRIPTION**

42495 CX The functionality described on this reference page is aligned with the ISO C standard. Any
42496 conflict between the requirements described here and the ISO C standard is unintentional. This
42497 volume of POSIX.1-2008 defers to the ISO C standard.

42498 If *s* is not a null pointer, *mbrlen()* shall determine the number of bytes constituting the character
42499 pointed to by *s*. It shall be equivalent to:

42500 mbstate_t internal;
42501 mbrtowc(NULL, *s*, *n*, *ps* != NULL ? *ps* : &internal);

42502 If *ps* is a null pointer, the *mbrlen()* function shall use its own internal **mbstate_t** object, which is
42503 initialized at program start-up to the initial conversion state. Otherwise, the **mbstate_t** object
42504 pointed to by *ps* shall be used to completely describe the current conversion state of the
42505 associated character sequence. The implementation shall behave as if no function defined in this
42506 volume of POSIX.1-2008 calls *mbrlen()*.

42507 The behavior of this function is affected by the *LC_CTYPE* category of the current locale.

42508 CX The *mbrlen()* function need not be thread-safe if called with a NULL *ps* argument. +

42509 The *mbrlen()* function shall not change the setting of *errno* if successful.

42510 **RETURN VALUE**

42511 The *mbrlen()* function shall return the first of the following that applies:

42512 0 If the next *n* or fewer bytes complete the character that corresponds to the null
42513 wide character.

42514 *positive* If the next *n* or fewer bytes complete a valid character; the value returned shall
42515 be the number of bytes that complete the character.

42516 (**size_t**)−2 If the next *n* bytes contribute to an incomplete but potentially valid character,
42517 and all *n* bytes have been processed. When *n* has at least the value of the
42518 {MB_CUR_MAX} macro, this case can only occur if *s* points at a sequence of
42519 redundant shift sequences (for implementations with state-dependent
42520 encodings).

42521 (**size_t**)−1 If an encoding error occurs, in which case the next *n* or fewer bytes do not
42522 contribute to a complete and valid character. In this case, [EILSEQ] shall be
42523 stored in *errno* and the conversion state is undefined.

42524 **ERRORS**

42525 The *mbrlen()* function shall fail if:

42526 [EILSEQ] An invalid character sequence is detected.

42527 The *mbrlen()* function may fail if:

42528 [EINVAL] *ps* points to an object that contains an invalid conversion state.

42529 **EXAMPLES**

42530 None.

42531 **APPLICATION USAGE**

42532 None.

42533 **RATIONALE**

42534 None.

42535 **FUTURE DIRECTIONS**

42536 None.

42537 **SEE ALSO**42538 *mbsinit()*, *mbrtowc()*

42539 XBD <wchar.h>

42540 **CHANGE HISTORY**42541 First released in Issue 5. Included for alignment with ISO/IEC 9899:1990/Amendment 1:1995
42542 (E).42543 **Issue 6**42544 The *mbrlen()* prototype is updated for alignment with the ISO/IEC 9899:1999 standard.42545 **Issue 7**

42546 Austin Group Interpretation 1003.1-2001 #170 is applied.

42547 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0368 [109,105] is applied.

+

NAME

`mbrtowc` — convert a character to a wide-character code (restartable)

SYNOPSIS

```
#include <wchar.h>
```

```
size_t mbrtowc(wchar_t *restrict pwc, const char *restrict s,
               size_t n, mbstate_t *restrict ps);
```

DESCRIPTION

The functionality described on this reference page is aligned with the ISO C standard. Any conflict between the requirements described here and the ISO C standard is unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

If *s* is a null pointer, the `mbrtowc()` function shall be equivalent to the call:

```
mbrtowc(NULL, "", 1, ps)
```

In this case, the values of the arguments *pwc* and *n* are ignored.

If *s* is not a null pointer, the `mbrtowc()` function shall inspect at most *n* bytes beginning at the byte pointed to by *s* to determine the number of bytes needed to complete the next character (including any shift sequences). If the function determines that the next character is completed, it shall determine the value of the corresponding wide character and then, if *pwc* is not a null pointer, shall store that value in the object pointed to by *pwc*. If the corresponding wide character is the null wide character, the resulting state described shall be the initial conversion state.

If *ps* is a null pointer, the `mbrtowc()` function shall use its own internal `mbstate_t` object, which shall be initialized at program start-up to the initial conversion state. Otherwise, the `mbstate_t` object pointed to by *ps* shall be used to completely describe the current conversion state of the associated character sequence. The implementation shall behave as if no function defined in this volume of POSIX.1-2008 calls `mbrtowc()`.

The behavior of this function is affected by the `LC_CTYPE` category of the current locale.

CX The `mbrtowc()` function need not be thread-safe if called with a NULL *ps* argument. +

The `mbrtowc()` function shall not change the setting of *errno* if successful.

RETURN VALUE

The `mbrtowc()` function shall return the first of the following that applies:

0 If the next *n* or fewer bytes complete the character that corresponds to the null wide character (which is the value stored).

between 1 and *n* inclusive

If the next *n* or fewer bytes complete a valid character (which is the value stored); the value returned shall be the number of bytes that complete the character.

(`size_t`)−2

If the next *n* bytes contribute to an incomplete but potentially valid character, and all *n* bytes have been processed (no value is stored). When *n* has at least the value of the `{MB_CUR_MAX}` macro, this case can only occur if *s* points at a sequence of redundant shift sequences (for implementations with state-dependent encodings).

(`size_t`)−1

If an encoding error occurs, in which case the next *n* or fewer bytes do not contribute to a complete and valid character (no value is stored). In this case, `[EILSEQ]` shall be stored in *errno* and the conversion state is undefined.

42592 ERRORS

42593 The *mbrtowc()* function shall fail if:

42594 [EILSEQ] An invalid character sequence is detected.

42595 The *mbrtowc()* function may fail if:

42596 CX [EINVAL] *ps* points to an object that contains an invalid conversion state.

42597 EXAMPLES

42598 None.

42599 APPLICATION USAGE

42600 None.

42601 RATIONALE

42602 None.

42603 FUTURE DIRECTIONS

42604 None.

42605 SEE ALSO

42606 *mbsinit()*, *mbsrtowcs()*

42607 XBD <wchar.h>

42608 CHANGE HISTORY

42609 First released in Issue 5. Included for alignment with ISO/IEC 9899:1990/Amendment 1:1995
42610 (E).

42611 Issue 6

42612 The *mbrtowc()* prototype is updated for alignment with the ISO/IEC 9899:1999 standard.

42613 The following new requirements on POSIX implementations derive from alignment with the
42614 Single UNIX Specification:

- 42615 • The [EINVAL] error condition is added.

42616 ISO/IEC 9899:1999 standard, Technical Corrigendum 1 is incorporated.

42617 Issue 7

42618 Austin Group Interpretation 1003.1-2001 #170 is applied.

42619 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0369 [109,105] is applied.

+

NAME

mbsinit — determine conversion object status

SYNOPSIS

```
#include <wchar.h>
```

```
int mbsinit(const mbstate_t *ps);
```

DESCRIPTION

CX The functionality described on this reference page is aligned with the ISO C standard. Any conflict between the requirements described here and the ISO C standard is unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

If *ps* is not a null pointer, the *mbsinit()* function shall determine whether the object pointed to by *ps* describes an initial conversion state.

RETURN VALUE

The *mbsinit()* function shall return non-zero if *ps* is a null pointer, or if the pointed-to object describes an initial conversion state; otherwise, it shall return zero.

If an **mbstate_t** object is altered by any of the functions described as “restartable”, and is then used with a different character sequence, or in the other conversion direction, or with a different *LC_CTYPE* category setting than on earlier function calls, the behavior is undefined.

ERRORS

No errors are defined.

EXAMPLES

None.

APPLICATION USAGE

The **mbstate_t** object is used to describe the current conversion state from a particular character sequence to a wide-character sequence (or *vice versa*) under the rules of a particular setting of the *LC_CTYPE* category of the current locale.

The initial conversion state corresponds, for a conversion in either direction, to the beginning of a new character sequence in the initial shift state. A zero valued **mbstate_t** object is at least one way to describe an initial conversion state. A zero valued **mbstate_t** object can be used to initiate conversion involving any character sequence, in any *LC_CTYPE* category setting.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

[*mbrlen\(\)*](#), [*mbrtowc\(\)*](#), [*mbsrtowcs\(\)*](#), [*wcrtomb\(\)*](#), [*wcsrtombs\(\)*](#)

XBD [**<wchar.h>**](#)

CHANGE HISTORY

First released in Issue 5. Included for alignment with ISO/IEC 9899:1990/Amendment 1:1995 (E).

42659 NAME

42660 mbsnrtowcs, mbsrtowcs — convert a character string to a wide-character string (restartable)

42661 SYNOPSIS

42662 #include <wchar.h>

```
42663 CX size_t mbsnrtowcs(wchar_t *restrict dst, const char **restrict src,
42664 size_t nmc, size_t len, mbstate_t *restrict ps);
42665 size_t mbsrtowcs(wchar_t *restrict dst, const char **restrict src,
42666 size_t len, mbstate_t *restrict ps);
```

42667 DESCRIPTION

42668 CX For *mbsrtowcs()*: The functionality described on this reference page is aligned with the ISO C
 42669 standard. Any conflict between the requirements described here and the ISO C standard is
 42670 unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

42671 The *mbsrtowcs()* function shall convert a sequence of characters, beginning in the conversion
 42672 state described by the object pointed to by *ps*, from the array indirectly pointed to by *src* into a
 42673 sequence of corresponding wide characters. If *dst* is not a null pointer, the converted characters
 42674 shall be stored into the array pointed to by *dst*. Conversion continues up to and including a
 42675 terminating null character, which shall also be stored. Conversion shall stop early in either of the
 42676 following cases:

- 42677 • A sequence of bytes is encountered that does not form a valid character.
- 42678 • *len* codes have been stored into the array pointed to by *dst* (and *dst* is not a null pointer).

42679 Each conversion shall take place as if by a call to the *mbrtowc()* function.

42680 If *dst* is not a null pointer, the pointer object pointed to by *src* shall be assigned either a null
 42681 pointer (if conversion stopped due to reaching a terminating null character) or the address just
 42682 past the last character converted (if any). If conversion stopped due to reaching a terminating
 42683 null character, and if *dst* is not a null pointer, the resulting state described shall be the initial
 42684 conversion state.

42685 If *ps* is a null pointer, the *mbsrtowcs()* function shall use its own internal **mbstate_t** object, which
 42686 is initialized at program start-up to the initial conversion state. Otherwise, the **mbstate_t** object
 42687 pointed to by *ps* shall be used to completely describe the current conversion state of the
 42688 associated character sequence.

42689 CX The *mbsnrtowcs()* function shall be equivalent to the *mbsrtowcs()* function, except that the
 42690 conversion of characters pointed to by *src* is limited to at most *nmc* bytes (the size of the input
 42691 buffer).

42692 The behavior of these functions shall be affected by the *LC_CTYPE* category of the current locale.

42693 The implementation shall behave as if no function defined in this volume of POSIX.1-2008 calls
 42694 these functions.

42695 CX The *mbsnrtowcs()* and *mbsrtowcs()* functions need not be thread-safe if called with a NULL *ps* +
 42696 argument. +

42697 The *mbsrtowcs()* function shall not change the setting of *errno* if successful.

42698 RETURN VALUE

42699 If the input conversion encounters a sequence of bytes that do not form a valid character, an
 42700 encoding error occurs. In this case, these functions shall store the value of the macro [EILSEQ] in
 42701 *errno* and shall return (**size_t**)–1; the conversion state is undefined. Otherwise, these functions
 42702 shall return the number of characters successfully converted, not including the terminating null
 42703 (if any).

42704 **ERRORS**

42705 These functions shall fail if:

42706 [EILSEQ] An invalid character sequence is detected.

42707 These functions may fail if:

42708 CX [EINVAL] *ps* points to an object that contains an invalid conversion state.42709 **EXAMPLES**

42710 None.

42711 **APPLICATION USAGE**

42712 None.

42713 **RATIONALE**

42714 None.

42715 **FUTURE DIRECTIONS**

42716 None.

42717 **SEE ALSO**42718 *iconv()*, *mbrtowc()*, *mbsinit()*

42719 XBD <wchar.h>

42720 **CHANGE HISTORY**42721 First released in Issue 5. Included for alignment with ISO/IEC 9899:1990/Amendment 1:1995
42722 (E).42723 **Issue 6**42724 The *mbsrtowcs()* prototype is updated for alignment with the ISO/IEC 9899:1999 standard.

42725 The [EINVAL] error condition is marked CX.

42726 **Issue 7**

42727 Austin Group Interpretation 1003.1-2001 #170 is applied.

42728 The *mbsnrtowcs()* function is added from The Open Group Technical Standard, 2006, Extended
42729 API Set Part 1.

42730 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0370 [109,105] is applied.

+

42731 **NAME**

42732 mbstowcs — convert a character string to a wide-character string

42733 **SYNOPSIS**

42734 #include <stdlib.h>

42735 size_t mbstowcs(wchar_t *restrict pwcs, const char *restrict s,
42736 size_t n);42737 **DESCRIPTION**42738 CX The functionality described on this reference page is aligned with the ISO C standard. Any
42739 conflict between the requirements described here and the ISO C standard is unintentional. This
42740 volume of POSIX.1-2008 defers to the ISO C standard.42741 The *mbstowcs()* function shall convert a sequence of characters that begins in the initial shift
42742 state from the array pointed to by *s* into a sequence of corresponding wide-character codes and
42743 shall store not more than *n* wide-character codes into the array pointed to by *pwcs*. No
42744 characters that follow a null byte (which is converted into a wide-character code with value 0)
42745 shall be examined or converted. Each character shall be converted as if by a call to *mbtowc()*,
42746 except that the shift state of *mbtowc()* is not affected.42747 No more than *n* elements shall be modified in the array pointed to by *pwcs*. If copying takes
42748 place between objects that overlap, the behavior is undefined.42749 XSI The behavior of this function shall be affected by the *LC_CTYPE* category of the current locale.
42750 If *pwcs* is a null pointer, *mbstowcs()* shall return the length required to convert the entire array
42751 regardless of the value of *n*, but no values are stored.42752 **RETURN VALUE**42753 CX If an invalid character is encountered, *mbstowcs()* shall return (size_t)-1 and shall set *errno* to
42754 indicate the error.42755 XSI Otherwise, *mbstowcs()* shall return the number of the array elements modified (or required if
42756 *pwcs* is null), not including a terminating 0 code, if any. The array shall not be zero-terminated if
42757 the value returned is *n*.42758 **ERRORS**42759 The *mbstowcs()* function shall fail if:

42760 XSI [EILSEQ] An invalid byte sequence is detected.

42761 **EXAMPLES**

42762 None.

42763 **APPLICATION USAGE**

42764 None.

42765 **RATIONALE**

42766 None.

42767 **FUTURE DIRECTIONS**

42768 None.

42769 **SEE ALSO**42770 *mblen()*, *mbtowc()*, *wctomb()*, *wcstombs()*

42771 XBD <stdlib.h>

42772 **CHANGE HISTORY**

42773 First released in Issue 4. Aligned with the ISO C standard.

42774 **Issue 6**42775 The *mbstowcs()* prototype is updated for alignment with the ISO/IEC 9899:1999 standard.

42776 Extensions beyond the ISO C standard are marked.

42777 **Issue 7**

42778 Austin Group Interpretation 1003.1-2001 #170 is applied.

42779 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0371 [195] is applied.

+

42780 **NAME**

42781 mbtowc — convert a character to a wide-character code

42782 **SYNOPSIS**

42783 #include <stdlib.h>

42784 int mbtowc(wchar_t *restrict *pwc*, const char *restrict *s*, size_t *n*);42785 **DESCRIPTION**

42786 CX The functionality described on this reference page is aligned with the ISO C standard. Any
 42787 conflict between the requirements described here and the ISO C standard is unintentional. This
 42788 volume of POSIX.1-2008 defers to the ISO C standard.

42789 If *s* is not a null pointer, *mbtowc()* shall determine the number of bytes that constitute the
 42790 character pointed to by *s*. It shall then determine the wide-character code for the value of type
 42791 **wchar_t** that corresponds to that character. (The value of the wide-character code corresponding
 42792 to the null byte is 0.) If the character is valid and *pwc* is not a null pointer, *mbtowc()* shall store
 42793 the wide-character code in the object pointed to by *pwc*.

42794 The behavior of this function is affected by the *LC_CTYPE* category of the current locale. For a
 42795 state-dependent encoding, this function is placed into its initial state by a call for which its
 42796 character pointer argument, *s*, is a null pointer. Subsequent calls with *s* as other than a null
 42797 pointer shall cause the internal state of the function to be altered as necessary. A call with *s* as a
 42798 null pointer shall cause this function to return a non-zero value if encodings have state
 42799 dependency, and 0 otherwise. If the implementation employs special bytes to change the shift
 42800 state, these bytes shall not produce separate wide-character codes, but shall be grouped with an
 42801 adjacent character. Changing the *LC_CTYPE* category causes the shift state of this function to be
 42802 unspecified. At most *n* bytes of the array pointed to by *s* shall be examined.

42803 The implementation shall behave as if no function defined in this volume of POSIX.1-2008 calls
 42804 *mbtowc()*.

42805 CX The *mbtowc()* function need not be thread-safe.

+

42806 **RETURN VALUE**

42807 If *s* is a null pointer, *mbtowc()* shall return a non-zero or 0 value, if character encodings,
 42808 respectively, do or do not have state-dependent encodings. If *s* is not a null pointer, *mbtowc()*
 42809 shall either return 0 (if *s* points to the null byte), or return the number of bytes that constitute the
 42810 CX converted character (if the next *n* or fewer bytes form a valid character), or return -1 and shall
 42811 set *errno* to indicate the error (if they do not form a valid character).

42812 In no case shall the value returned be greater than *n* or the value of the {MB_CUR_MAX} macro.

42813 **ERRORS**

42814 The *mbtowc()* function shall fail if:

42815 XSI [EILSEQ] An invalid character sequence is detected.

42816 **EXAMPLES**

42817 None.

42818 **APPLICATION USAGE**

42819 None.

42820 **RATIONALE**

42821 None.

42822 **FUTURE DIRECTIONS**

42823 None.

42824 **SEE ALSO**42825 *mblen()*, *mbstowcs()*, *wctomb()*, *wcstombs()*42826 XBD **<stdlib.h>**42827 **CHANGE HISTORY**

42828 First released in Issue 4. Aligned with the ISO C standard.

42829 **Issue 6**42830 The *mbtowc()* prototype is updated for alignment with the ISO/IEC 9899:1999 standard.

42831 Extensions beyond the ISO C standard are marked.

42832 **Issue 7**

42833 Austin Group Interpretation 1003.1-2001 #170 is applied.

42834 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0372 [109] and XSH/TC1-2008/0373 +
42835 [195] are applied.

42836 **NAME**

42837 memccpy — copy bytes in memory

42838 **SYNOPSIS**

```
42839 XSI      #include <string.h>
42840          void *memccpy(void *restrict s1, const void *restrict s2,
42841                        int c, size_t n);
```

42842 **DESCRIPTION**

42843 The *memccpy()* function shall copy bytes from memory area *s2* into *s1*, stopping after the first
42844 occurrence of byte *c* (converted to an **unsigned char**) is copied, or after *n* bytes are copied,
42845 whichever comes first. If copying takes place between objects that overlap, the behavior is
42846 undefined.

42847 **RETURN VALUE**

42848 The *memccpy()* function shall return a pointer to the byte after the copy of *c* in *s1*, or a null
42849 pointer if *c* was not found in the first *n* bytes of *s2*.

42850 **ERRORS**

42851 No errors are defined.

42852 **EXAMPLES**

42853 None.

42854 **APPLICATION USAGE**

42855 The *memccpy()* function does not check for the overflow of the receiving memory area.

42856 **RATIONALE**

42857 None.

42858 **FUTURE DIRECTIONS**

42859 None.

42860 **SEE ALSO**

42861 XBD [<string.h>](#)

42862 **CHANGE HISTORY**

42863 First released in Issue 1. Derived from Issue 1 of the SVID.

42864 **Issue 6**

42865 The **restrict** keyword is added to the *memccpy()* prototype for alignment with the
42866 ISO/IEC 9899:1999 standard.

42867 **NAME**

42868 memchr — find byte in memory

42869 **SYNOPSIS**

42870 #include <string.h>

42871 void *memchr(const void *s, int c, size_t n);

42872 **DESCRIPTION**

42873 CX The functionality described on this reference page is aligned with the ISO C standard. Any
 42874 conflict between the requirements described here and the ISO C standard is unintentional. This
 42875 volume of POSIX.1-2008 defers to the ISO C standard.

42876 The *memchr()* function shall locate the first occurrence of *c* (converted to an **unsigned char**) in
 42877 the initial *n* bytes (each interpreted as **unsigned char**) pointed to by *s*. |

42878 Implementations shall behave as if they read the memory byte by byte from the beginning of the +
 42879 bytes pointed to by *s* and stop at the first occurrence of *c* (if it is found in the initial *n* bytes).

42880 **RETURN VALUE**

42881 The *memchr()* function shall return a pointer to the located byte, or a null pointer if the byte is |
 42882 not found.

42883 **ERRORS**

42884 No errors are defined.

42885 **EXAMPLES**

42886 None.

42887 **APPLICATION USAGE**

42888 None.

42889 **RATIONALE**

42890 None.

42891 **FUTURE DIRECTIONS**

42892 None.

42893 **SEE ALSO**42894 XBD [<string.h>](#)42895 **CHANGE HISTORY**

42896 First released in Issue 1. Derived from Issue 1 of the SVID.

42897 **Issue 7**

42898 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0374 [110] is applied. +

42899 **NAME**

42900 memcmp — compare bytes in memory

42901 **SYNOPSIS**

42902 #include <string.h>

42903 int memcmp(const void *s1, const void *s2, size_t n);

42904 **DESCRIPTION**

42905 **CX** The functionality described on this reference page is aligned with the ISO C standard. Any
42906 conflict between the requirements described here and the ISO C standard is unintentional. This
42907 volume of POSIX.1-2008 defers to the ISO C standard.

42908 The *memcmp()* function shall compare the first *n* bytes (each interpreted as **unsigned char**) of the
42909 object pointed to by *s1* to the first *n* bytes of the object pointed to by *s2*.

42910 The sign of a non-zero return value shall be determined by the sign of the difference between the
42911 values of the first pair of bytes (both interpreted as type **unsigned char**) that differ in the objects
42912 being compared.

42913 **RETURN VALUE**

42914 The *memcmp()* function shall return an integer greater than, equal to, or less than 0, if the object
42915 pointed to by *s1* is greater than, equal to, or less than the object pointed to by *s2*, respectively.

42916 **ERRORS**

42917 No errors are defined.

42918 **EXAMPLES**

42919 None.

42920 **APPLICATION USAGE**

42921 None.

42922 **RATIONALE**

42923 None.

42924 **FUTURE DIRECTIONS**

42925 None.

42926 **SEE ALSO**

42927 XBD <string.h>

42928 **CHANGE HISTORY**

42929 First released in Issue 1. Derived from Issue 1 of the SVID.

42930 **NAME**

42931 memcpy — copy bytes in memory

42932 **SYNOPSIS**

42933 #include <string.h>

42934 void *memcpy(void *restrict s1, const void *restrict s2, size_t n);

42935 **DESCRIPTION**

42936 CX The functionality described on this reference page is aligned with the ISO C standard. Any
42937 conflict between the requirements described here and the ISO C standard is unintentional. This
42938 volume of POSIX.1-2008 defers to the ISO C standard.

42939 The *memcpy()* function shall copy *n* bytes from the object pointed to by *s2* into the object pointed
42940 to by *s1*. If copying takes place between objects that overlap, the behavior is undefined.

42941 **RETURN VALUE**42942 The *memcpy()* function shall return *s1*; no return value is reserved to indicate an error.42943 **ERRORS**

42944 No errors are defined.

42945 **EXAMPLES**

42946 None.

42947 **APPLICATION USAGE**42948 The *memcpy()* function does not check for the overflow of the receiving memory area.42949 **RATIONALE**

42950 None.

42951 **FUTURE DIRECTIONS**

42952 None.

42953 **SEE ALSO**

42954 XBD <string.h>

42955 **CHANGE HISTORY**

42956 First released in Issue 1. Derived from Issue 1 of the SVID.

42957 **Issue 6**42958 The *memcpy()* prototype is updated for alignment with the ISO/IEC 9899:1999 standard.

42959 **NAME**

42960 memmove — copy bytes in memory with overlapping areas

42961 **SYNOPSIS**

42962 #include <string.h>

42963 void *memmove(void *s1, const void *s2, size_t n);

42964 **DESCRIPTION**

42965 CX The functionality described on this reference page is aligned with the ISO C standard. Any
42966 conflict between the requirements described here and the ISO C standard is unintentional. This
42967 volume of POSIX.1-2008 defers to the ISO C standard.

42968 The *memmove()* function shall copy *n* bytes from the object pointed to by *s2* into the object
42969 pointed to by *s1*. Copying takes place as if the *n* bytes from the object pointed to by *s2* are first
42970 copied into a temporary array of *n* bytes that does not overlap the objects pointed to by *s1* and
42971 *s2*, and then the *n* bytes from the temporary array are copied into the object pointed to by *s1*.

42972 **RETURN VALUE**42973 The *memmove()* function shall return *s1*; no return value is reserved to indicate an error.42974 **ERRORS**

42975 No errors are defined.

42976 **EXAMPLES**

42977 None.

42978 **APPLICATION USAGE**

42979 None.

42980 **RATIONALE**

42981 None.

42982 **FUTURE DIRECTIONS**

42983 None.

42984 **SEE ALSO**

42985 XBD <string.h>

42986 **CHANGE HISTORY**

42987 First released in Issue 4. Derived from the ANSI C standard.

42988 **NAME**

42989 memset — set bytes in memory

42990 **SYNOPSIS**

42991 #include <string.h>

42992 void *memset(void *s, int c, size_t n);

42993 **DESCRIPTION**

42994 CX The functionality described on this reference page is aligned with the ISO C standard. Any
42995 conflict between the requirements described here and the ISO C standard is unintentional. This
42996 volume of POSIX.1-2008 defers to the ISO C standard.

42997 The *memset()* function shall copy *c* (converted to an **unsigned char**) into each of the first *n* bytes
42998 of the object pointed to by *s*.

42999 **RETURN VALUE**43000 The *memset()* function shall return *s*; no return value is reserved to indicate an error.43001 **ERRORS**

43002 No errors are defined.

43003 **EXAMPLES**

43004 None.

43005 **APPLICATION USAGE**

43006 None.

43007 **RATIONALE**

43008 None.

43009 **FUTURE DIRECTIONS**

43010 None.

43011 **SEE ALSO**

43012 XBD <string.h>

43013 **CHANGE HISTORY**

43014 First released in Issue 1. Derived from Issue 1 of the SVID.

NAME

mkdir, mkdirat — make a directory relative to directory file descriptor

SYNOPSIS

```
#include <sys/stat.h>
```

```
int mkdir(const char *path, mode_t mode);
```

```
int mkdirat(int fd, const char *path, mode_t mode);
```

DESCRIPTION

The *mkdir()* function shall create a new directory with name *path*. The file permission bits of the new directory shall be initialized from *mode*. These file permission bits of the *mode* argument shall be modified by the process' file creation mask.

When bits in *mode* other than the file permission bits are set, the meaning of these additional bits is implementation-defined.

The directory's user ID shall be set to the process' effective user ID. The directory's group ID shall be set to the group ID of the parent directory or to the effective group ID of the process. Implementations shall provide a way to initialize the directory's group ID to the group ID of the parent directory. Implementations may, but need not, provide an implementation-defined way to initialize the directory's group ID to the effective group ID of the calling process.

The newly created directory shall be an empty directory.

If *path* names a symbolic link, *mkdir()* shall fail and set *errno* to [EEXIST].

Upon successful completion, *mkdir()* shall mark for update the last data access, last data modification, and last file status change timestamps of the directory. Also, the last data modification and last file status change timestamps of the directory that contains the new entry shall be marked for update.

The *mkdirat()* function shall be equivalent to the *mkdir()* function except in the case where *path* specifies a relative path. In this case the newly created directory is created relative to the directory associated with the file descriptor *fd* instead of the current working directory. If the file descriptor was opened without O_SEARCH, the function shall check whether directory searches are permitted using the current permissions of the directory underlying the file descriptor. If the file descriptor was opened with O_SEARCH, the function shall not perform the check.

If *mkdirat()* is passed the special value AT_FDCWD in the *fd* parameter, the current working directory shall be used and the behavior shall be identical to a call to *mkdir()*.

RETURN VALUE

Upon successful completion, these functions shall return 0. Otherwise, these functions shall return -1 and set *errno* to indicate the error. If -1 is returned, no directory shall be created.

ERRORS

These functions shall fail if:

[EACCES] Search permission is denied on a component of the path prefix, or write permission is denied on the parent directory of the directory to be created.

[EEXIST] The named file exists.

[ELOOP] A loop exists in symbolic links encountered during resolution of the *path* argument.

[EMLINK] The link count of the parent directory would exceed {LINK_MAX}.

43057	[ENAMETOOLONG]	
43058		The length of a component of a pathname is longer than {NAME_MAX}.
43059	[ENOENT]	A component of the path prefix specified by <i>path</i> does not name an existing directory or <i>path</i> is an empty string.
43060		
43061	[ENOSPC]	The file system does not contain enough space to hold the contents of the new directory or to extend the parent directory of the new directory.
43062		
43063	[ENOTDIR]	A component of the path prefix names an existing file that is neither a directory nor a symbolic link to a directory.
43064		
43065	[EROFS]	The parent directory resides on a read-only file system.
43066	In addition, the <i>mkdirat</i> () function shall fail if:	
43067	[EBADF]	The <i>path</i> argument does not specify an absolute path and the <i>fd</i> argument is neither AT_FDCWD nor a valid file descriptor open for reading or searching.
43068		
43069	[ENOTDIR]	The <i>path</i> argument is not an absolute path and <i>fd</i> is a file descriptor associated with a non-directory file.
43070		
43071	These functions may fail if:	
43072	[ELOOP]	More than {SYMLOOP_MAX} symbolic links were encountered during resolution of the <i>path</i> argument.
43073		
43074	[ENAMETOOLONG]	
43075		The length of a pathname exceeds {PATH_MAX}, or pathname resolution of a symbolic link produced an intermediate result with a length that exceeds {PATH_MAX}.
43076		
43077		

43078 EXAMPLES

43079 Creating a Directory

43080 The following example shows how to create a directory named **/home/cnd/mod1**, with
 43081 read/write/search permissions for owner and group, and with read/search permissions for
 43082 others.

```

43083 #include <sys/types.h>
43084 #include <sys/stat.h>

43085 int status;
43086 ...
43087 status = mkdir("/home/cnd/mod1", S_IRWXU | S_IRWXG | S_IROTH | S_IXOTH);
  
```

43088 APPLICATION USAGE

43089 None.

43090 RATIONALE

43091 The *mkdir*() function originated in 4.2 BSD and was added to System V in Release 3.0.

43092 4.3 BSD detects [ENAMETOOLONG].

43093 The POSIX.1-1990 standard required that the group ID of a newly created directory be set to the
 43094 group ID of its parent directory or to the effective group ID of the creating process. FIPS 151-2
 43095 required that implementations provide a way to have the group ID be set to the group ID of the
 43096 containing directory, but did not prohibit implementations also supporting a way to set the
 43097 group ID to the effective group ID of the creating process. Conforming applications should not
 43098 assume which group ID will be used. If it matters, an application can use *chown*() to set the

group ID after the directory is created, or determine under what conditions the implementation will set the desired group ID.

The purpose of the *mkdirat()* function is to create a directory in directories other than the current working directory without exposure to race conditions. Any part of the path of a file could be changed in parallel to the call to *mkdir()*, resulting in unspecified behavior. By opening a file descriptor for the target directory and using the *mkdirat()* function it can be guaranteed that the newly created directory is located relative to the desired directory.

FUTURE DIRECTIONS

None.

SEE ALSO

chmod(), *mkdtemp()*, *mknod()*, *umask()*

XBD [<sys/stat.h>](#), [<sys/types.h>](#)

CHANGE HISTORY

First released in Issue 3. Included for alignment with the POSIX.1-1988 standard.

Issue 6

In the SYNOPSIS, the optional include of the [<sys/types.h>](#) header is removed.

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The requirement to include [<sys/types.h>](#) has been removed. Although [<sys/types.h>](#) was required for conforming implementations of previous POSIX specifications, it was not required for UNIX applications.
- The [ELOOP] mandatory error condition is added.
- A second [ENAMETOOLONG] is added as an optional error condition.

The following changes were made to align with the IEEE P1003.1a draft standard:

- The [ELOOP] optional error condition is added.

Issue 7

Austin Group Interpretation 1003.1-2001 #143 is applied.

The *mkdirat()* function is added from The Open Group Technical Standard, 2006, Extended API Set Part 2.

Changes are made related to support for finegrained timestamps.

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0375 [461], XSH/TC1-2008/0376 [324], + XSH/TC1-2008/0377 [277], XSH/TC1-2008/0378 [278], and XSH/TC1-2008/0379 [278] are + applied.

NAME

mkdtemp, mkstemp — create a unique directory or file

SYNOPSIS

```
CX      #include <stdlib.h>
char *mkdtemp(char *template);
int mkstemp(char *template);
```

DESCRIPTION

The *mkdtemp()* function uses the contents of *template* to construct a unique directory name. The string provided in *template* shall be a pathname ending with six trailing 'X' s. The *mkdtemp()* function shall replace each 'X' with a character from the portable filename character set. The characters are chosen such that the resulting name does not duplicate the name of an existing file at the time of a call to *mkdtemp()*. The unique directory name is used to attempt to create the directory using mode 0700 as modified by the file creation mask.

The *mkstemp()* function shall replace the contents of the string pointed to by *template* by a unique pathname, and return a file descriptor for the file open for reading and writing. The *mkstemp()* function shall create the file, and obtain a file descriptor for it, as if by a call to:

```
open(pathname, O_RDWR|O_CREAT|O_EXCL, S_IRUSR|S_IWUSR)
```

The function thus prevents any possible race condition between testing whether the file exists and opening it for use. The string in *template* should look like a pathname with six trailing 'X' s; *mkstemp()* replaces each 'X' with a character from the portable filename character set. The characters are chosen such that the resulting name does not duplicate the name of an existing file at the time of a call to *mkstemp()*.

RETURN VALUE

Upon successful completion, the *mkdtemp()* function shall return a pointer to the string containing the directory name if it was created. Otherwise, it shall return a null pointer and shall set *errno* to indicate the error.

Upon successful completion, the *mkstemp()* function shall return an open file descriptor. Otherwise, it shall return -1 if no suitable file could be created.

ERRORS

The *mkdtemp()* function shall fail if:

[EACCES]	Search permission is denied on a component of the path prefix, or write permission is denied on the parent directory of the directory to be created.
[EINVAL]	The string pointed to by <i>template</i> does not end in "XXXXXX".
[ELOOP]	A loop exists in symbolic links encountered during resolution of the path of the directory to be created.
[EMLINK]	The link count of the parent directory would exceed {LINK_MAX}.
[ENAMETOOLONG]	The length of a component of a pathname is longer than {NAME_MAX}.
[ENOENT]	A component of the path prefix specified by the <i>template</i> argument does not name an existing directory.
[ENOSPC]	The file system does not contain enough space to hold the contents of the new directory or to extend the parent directory of the new directory.

43174 [ENOTDIR] A component of the path prefix names an existing file that is neither a |
 43175 directory nor a symbolic link to a directory.

43176 [EROFS] The parent directory resides on a read-only file system.

43177 The *mkdtemp()* function may fail if:

43178 [ELOOP] More than {SYMLOOP_MAX} symbolic links were encountered during
 43179 resolution of the path of the directory to be created.

43180 [ENAMETOOLONG]
 43181 The length of a pathname exceeds {PATH_MAX}, or pathname resolution of a
 43182 symbolic link produced an intermediate result with a length that exceeds
 43183 {PATH_MAX}.

43184 The error conditions for the *mkstemp()* function are defined in *open()*.

43185 EXAMPLES

43186 Generating a Pathname

43187 The following example creates a file with a 10-character name beginning with the characters
 43188 "file" and opens the file for reading and writing. The value returned as the value of *fd* is a file
 43189 descriptor that identifies the file.

```
43190 #include <stdlib.h>
43191 ...
43192 char template[] = "/tmp/fileXXXXXX";
43193 int fd;
43194 fd = mkstemp(template);
```

43195 APPLICATION USAGE

43196 It is possible to run out of letters.

43197 The *mkdtemp()* and *mkstemp()* functions need not check to determine whether the filename part
 43198 of *template* exceeds the maximum allowable filename length.

43199 RATIONALE

43200 None.

43201 FUTURE DIRECTIONS

43202 None.

43203 SEE ALSO

43204 *getpid()*, *mkdir()*, *open()*, *tmpfile()*, *tmpnam()*

43205 XBD *<stdlib.h>*

43206 CHANGE HISTORY

43207 First released in Issue 4, Version 2.

43208 Issue 5

43209 Moved from X/OPEN UNIX extension to BASE.

Issue 7

- 43210
- 43211 Austin Group Interpretation 1003.1-2001 #143 is applied.
- 43212 SD5-XSH-ERN-168 is applied, clarifying file permissions upon creation.
- 43213 The *mkstemp()* function is moved from the XSI option to the Base.
- 43214 The *mkdtemp()* function is added from The Open Group Technical Standard, 2006, Extended API
- 43215 Set Part 1.
- 43216 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0380 [291], XSH/TC1-2008/0381 [324], +
- 43217 and XSH/TC1-2008/0382 [291] are applied.

NAME

mkfifo, mkfifoat — make a FIFO special file relative to directory file descriptor

SYNOPSIS

```
#include <sys/stat.h>

int mkfifo(const char *path, mode_t mode);
int mkfifoat(int fd, const char *path, mode_t mode);
```

DESCRIPTION

The *mkfifo()* function shall create a new FIFO special file named by the pathname pointed to by *path*. The file permission bits of the new FIFO shall be initialized from *mode*. The file permission bits of the *mode* argument shall be modified by the process' file creation mask.

When bits in *mode* other than the file permission bits are set, the effect is implementation-defined.

If *path* names a symbolic link, *mkfifo()* shall fail and set *errno* to [EEXIST].

The FIFO's user ID shall be set to the process' effective user ID. The FIFO's group ID shall be set to the group ID of the parent directory or to the effective group ID of the process. Implementations shall provide a way to initialize the FIFO's group ID to the group ID of the parent directory. Implementations may, but need not, provide an implementation-defined way to initialize the FIFO's group ID to the effective group ID of the calling process.

Upon successful completion, *mkfifo()* shall mark for update the last data access, last data modification, and last file status change timestamps of the file. Also, the last data modification and last file status change timestamps of the directory that contains the new entry shall be marked for update.

The *mkfifoat()* function shall be equivalent to the *mkfifo()* function except in the case where *path* specifies a relative path. In this case the newly created FIFO is created relative to the directory associated with the file descriptor *fd* instead of the current working directory. If the file descriptor was opened without O_SEARCH, the function shall check whether directory searches are permitted using the current permissions of the directory underlying the file descriptor. If the file descriptor was opened with O_SEARCH, the function shall not perform the check.

If *mkfifoat()* is passed the special value AT_FDCWD in the *fd* parameter, the current working directory shall be used and the behavior shall be identical to a call to *mkfifo()*.

RETURN VALUE

Upon successful completion, these functions shall return 0. Otherwise, these functions shall return -1 and set *errno* to indicate the error. If -1 is returned, no FIFO shall be created.

ERRORS

These functions shall fail if:

[EACCES] A component of the path prefix denies search permission, or write permission is denied on the parent directory of the FIFO to be created.

[EEXIST] The named file already exists.

[ELOOP] A loop exists in symbolic links encountered during resolution of the *path* argument.

[ENAMETOOLONG]

The length of a component of a pathname is longer than {NAME_MAX}.

43260	[ENOENT]	A component of the path prefix of <i>path</i> does not name an existing file or <i>path</i> is an empty string.	
43261			
43262	[ENOENT] or [ENOTDIR]		
43263		The <i>path</i> argument contains at least one non- <code><slash></code> character and ends with one or more trailing <code><slash></code> characters. If <i>path</i> names an existing file, an [ENOENT] error shall not occur.	+
43264			+
43265			
43266	[ENOSPC]	The directory that would contain the new file cannot be extended or the file system is out of file-allocation resources.	
43267			
43268	[ENOTDIR]	A component of the path prefix names an existing file that is neither a directory nor a symbolic link to a directory.	
43269			
43270	[EROFS]	The named file resides on a read-only file system.	
43271		The <i>mkfifoat()</i> function shall fail if:	
43272	[EACCES]	<i>fd</i> was not opened with <code>O_SEARCH</code> and the permissions of the directory underlying <i>fd</i> do not permit directory searches.	
43273			
43274	[EBADF]	The <i>path</i> argument does not specify an absolute path and the <i>fd</i> argument is neither <code>AT_FDCWD</code> nor a valid file descriptor open for reading or searching.	
43275			
43276	[ENOTDIR]	The <i>path</i> argument is not an absolute path and <i>fd</i> is a file descriptor associated with a non-directory file.	+
43277			
43278		These functions may fail if:	
43279	[ELOOP]	More than <code>{SYMLOOP_MAX}</code> symbolic links were encountered during resolution of the <i>path</i> argument.	
43280			
43281	[ENAMETOOLONG]		
43282		The length of a pathname exceeds <code>{PATH_MAX}</code> , or pathname resolution of a symbolic link produced an intermediate result with a length that exceeds <code>{PATH_MAX}</code> .	
43283			
43284			

EXAMPLES

Creating a FIFO File

The following example shows how to create a FIFO file named `/home/cnd/mod_done`, with read/write permissions for owner, and with read permissions for group and others.

```
#include <sys/types.h>
#include <sys/stat.h>

int status;
...
status = mkfifo("/home/cnd/mod_done", S_IWUSR | S_IRUSR |
    S_IRGRP | S_IROTH);
```

APPLICATION USAGE

None.

RATIONALE

The syntax of this function is intended to maintain compatibility with historical implementations of *mknod()*. The latter function was included in the 1984 `/usr/group` standard but only for use in creating FIFO special files. The *mknod()* function was originally excluded from the POSIX.1-1988 standard as implementation-defined and replaced by *mkdir()* and

mkfifo(). The *mknod()* function is now included for alignment with the Single UNIX Specification.

The POSIX.1-1990 standard required that the group ID of a newly created FIFO be set to the group ID of its parent directory or to the effective group ID of the creating process. FIPS 151-2 required that implementations provide a way to have the group ID be set to the group ID of the containing directory, but did not prohibit implementations also supporting a way to set the group ID to the effective group ID of the creating process. Conforming applications should not assume which group ID will be used. If it matters, an application can use *chown()* to set the group ID after the FIFO is created, or determine under what conditions the implementation will set the desired group ID.

The purpose of the *mkfifoat()* function is to create a FIFO special file in directories other than the current working directory without exposure to race conditions. Any part of the path of a file could be changed in parallel to a call to *mkfifo()*, resulting in unspecified behavior. By opening a file descriptor for the target directory and using the *mkfifoat()* function it can be guaranteed that the newly created FIFO is located relative to the desired directory.

FUTURE DIRECTIONS

None.

SEE ALSO

chmod(), *mknod()*, *umask()*

XBD [*<sys/stat.h>*](#), [*<sys/types.h>*](#)

CHANGE HISTORY

First released in Issue 3. Included for alignment with the POSIX.1-1988 standard.

Issue 6

In the SYNOPSIS, the optional include of the [*<sys/types.h>*](#) header is removed.

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The requirement to include [*<sys/types.h>*](#) has been removed. Although [*<sys/types.h>*](#) was required for conforming implementations of previous POSIX specifications, it was not required for UNIX applications.
- The [ELOOP] mandatory error condition is added.
- A second [ENAMETOOLONG] is added as an optional error condition.

The following changes were made to align with the IEEE P1003.1a draft standard:

- The [ELOOP] optional error condition is added.

Issue 7

Austin Group Interpretation 1003.1-2001 #143 is applied.

The *mkfifoat()* function is added from The Open Group Technical Standard, 2006, Extended API Set Part 2.

Changes are made related to support for finegrained timestamps.

Changes are made to allow a directory to be opened for searching.

43341 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0383 [461], XSH/TC1-2008/0384 +
43342 [146,435], XSH/TC1-2008/0385 [324], XSH/TC1-2008/0386 [278], and XSH/TC1-2008/0387 +
43343 [278] are applied.

NAME

mknod, mknodat — make directory, special file, or regular file

SYNOPSIS

```
#include <sys/stat.h>

int mknod(const char *path, mode_t mode, dev_t dev);
int mknodat(int fd, const char *path, mode_t mode, dev_t dev);
```

DESCRIPTION

The *mknod()* function shall create a new file named by the pathname to which the argument *path* points.

The file type for *path* is OR'ed into the *mode* argument, and the application shall select one of the following symbolic constants:

Name	Description
S_IFIFO	FIFO-special
S_IFCHR	Character-special (non-portable)
S_IFDIR	Directory (non-portable)
S_IFBLK	Block-special (non-portable)
S_IFREG	Regular (non-portable)

The only portable use of *mknod()* is to create a FIFO-special file. If *mode* is not S_IFIFO or *dev* is not 0, the behavior of *mknod()* is unspecified.

The permissions for the new file are OR'ed into the *mode* argument, and may be selected from any combination of the following symbolic constants:

Name	Description
S_ISUID	Set user ID on execution.
S_ISGID	Set group ID on execution.
S_IRWXU	Read, write, or execute (search) by owner.
S_IRUSR	Read by owner.
S_IWUSR	Write by owner.
S_IXUSR	Execute (search) by owner.
S_IRWXG	Read, write, or execute (search) by group.
S_IRGRP	Read by group.
S_IWGRP	Write by group.
S_IXGRP	Execute (search) by group.
S_IRWXO	Read, write, or execute (search) by others.
S_IROTH	Read by others.
S_IWOTH	Write by others.
S_IXOTH	Execute (search) by others.
S_ISVTX	On directories, restricted deletion flag.

The user ID of the file shall be initialized to the effective user ID of the process. The group ID of the file shall be initialized to either the effective group ID of the process or the group ID of the parent directory. Implementations shall provide a way to initialize the file's group ID to the group ID of the parent directory. Implementations may, but need not, provide an implementation-defined way to initialize the file's group ID to the effective group ID of the calling process. The owner, group, and other permission bits of *mode* shall be modified by the file mode creation mask of the process. The *mknod()* function shall clear each bit whose corresponding bit in the file mode creation mask of the process is set.

43389	If <i>path</i> names a symbolic link, <i>mknod()</i> shall fail and set <i>errno</i> to [EEXIST].	
43390	Upon successful completion, <i>mknod()</i> shall mark for update the last data access, last data	
43391	modification, and last file status change timestamps of the file. Also, the last data modification	
43392	and last file status change timestamps of the directory that contains the new entry shall be	
43393	marked for update.	
43394	Only a process with appropriate privileges may invoke <i>mknod()</i> for file types other than FIFO-	
43395	special.	
43396	The <i>mknodat()</i> function shall be equivalent to the <i>mknod()</i> function except in the case where <i>path</i>	
43397	specifies a relative path. In this case the newly created directory, special file, or regular file is	
43398	located relative to the directory associated with the file descriptor <i>fd</i> instead of the current	
43399	working directory. If the file descriptor was opened without O_SEARCH, the function shall	
43400	check whether directory searches are permitted using the current permissions of the directory	
43401	underlying the file descriptor. If the file descriptor was opened with O_SEARCH, the function	
43402	shall not perform the check.	
43403	If <i>mknodat()</i> is passed the special value AT_FDCWD in the <i>fd</i> parameter, the current working	
43404	directory shall be used and the behavior shall be identical to a call to <i>mknod()</i> .	
43405	RETURN VALUE	
43406	Upon successful completion, these functions shall return 0. Otherwise, these functions shall	
43407	return -1 and set <i>errno</i> to indicate the error. If -1 is returned, the new file shall not be created.	
43408	ERRORS	
43409	These functions shall fail if:	
43410	[EACCES]	A component of the path prefix denies search permission, or write permission
43411		is denied on the parent directory.
43412	[EEXIST]	The named file exists.
43413	[EINVAL]	An invalid argument exists.
43414	[EIO]	An I/O error occurred while accessing the file system.
43415	[ELOOP]	A loop exists in symbolic links encountered during resolution of the <i>path</i>
43416		argument.
43417	[ENAMETOOLONG]	
43418		The length of a component of a pathname is longer than {NAME_MAX}.
43419	[ENOENT]	A component of the path prefix of <i>path</i> does not name an existing file or <i>path</i> is
43420		an empty string.
43421	[ENOENT] or [ENOTDIR]	
43422		The <i>path</i> argument contains at least one non-<slash> character and ends with
43423		one or more trailing <slash> characters. If <i>path</i> names an existing file, an
43424		[ENOENT] error shall not occur.
43425	[ENOSPC]	The directory that would contain the new file cannot be extended or the file
43426		system is out of file allocation resources.
43427	[ENOTDIR]	A component of the path prefix names an existing file that is neither a
43428		directory nor a symbolic link to a directory.
43429	[EPERM]	The invoking process does not have appropriate privileges and the file type is
43430		not FIFO-special.

43431	[EROFS]	The directory in which the file is to be created is located on a read-only file system.
43432		
43433	The <i>mknodat()</i> function shall fail if:	
43434	[EACCES]	<i>fd</i> was not opened with O_SEARCH and the permissions of the directory underlying <i>fd</i> do not permit directory searches.
43435		
43436	[EBADF]	The <i>path</i> argument does not specify an absolute path and the <i>fd</i> argument is neither AT_FDCWD nor a valid file descriptor open for reading or searching.
43437		
43438	[ENOTDIR]	The <i>path</i> argument is not an absolute path and <i>fd</i> is a file descriptor associated with a non-directory file.
43439		
43440	These functions may fail if:	
43441	[ELOOP]	More than {SYMLOOP_MAX} symbolic links were encountered during resolution of the <i>path</i> argument.
43442		
43443	[ENAMETOOLONG]	The length of a pathname exceeds {PATH_MAX}, or pathname resolution of a symbolic link produced an intermediate result with a length that exceeds {PATH_MAX}.
43444		
43445		
43446		

43447 EXAMPLES

43448 Creating a FIFO Special File

43449 The following example shows how to create a FIFO special file named */home/cnd/mod_done*,
 43450 with read/write permissions for owner, and with read permissions for group and others.

```

43451 #include <sys/types.h>
43452 #include <sys/stat.h>

43453 dev_t dev;
43454 int status;
43455 ...
43456 status = mknod("/home/cnd/mod_done", S_IFIFO | S_IWUSR |
43457             S_IRUSR | S_IRGRP | S_IROTH, dev);

```

43458 APPLICATION USAGE

43459 The *mkfifo()* function is preferred over this function for making FIFO special files.

43460 RATIONALE

43461 The POSIX.1-1990 standard required that the group ID of a newly created file be set to the group
 43462 ID of its parent directory or to the effective group ID of the creating process. FIPS 151-2 required
 43463 that implementations provide a way to have the group ID be set to the group ID of the
 43464 containing directory, but did not prohibit implementations also supporting a way to set the
 43465 group ID to the effective group ID of the creating process. Conforming applications should not
 43466 assume which group ID will be used. If it matters, an application can use *chown()* to set the
 43467 group ID after the file is created, or determine under what conditions the implementation will
 43468 set the desired group ID.

43469 The purpose of the *mknodat()* function is to create directories, special files, or regular files in
 43470 directories other than the current working directory without exposure to race conditions. Any
 43471 part of the path of a file could be changed in parallel to a call to *mknod()*, resulting in unspecified
 43472 behavior. By opening a file descriptor for the target directory and using the *mknodat()* function it
 43473 can be guaranteed that the newly created directory, special file, or regular file is located relative
 43474 to the desired directory.

43475 **FUTURE DIRECTIONS**

43476 None.

43477 **SEE ALSO**43478 *chmod()*, *creat()*, *exec*, *fstatat()*, *mkdir()*, *mkfifo()*, *open()*, *umask()*

43479 XBD <sys/stat.h>

43480 **CHANGE HISTORY**

43481 First released in Issue 4, Version 2.

43482 **Issue 5**

43483 Moved from X/OPEN UNIX extension to BASE.

43484 **Issue 6**

43485 The normative text is updated to avoid use of the term “must” for application requirements.

43486 The wording of the mandatory [ELOOP] error condition is updated, and a second optional
43487 [ELOOP] error condition is added.43488 **Issue 7**

43489 Austin Group Interpretation 1003.1-2001 #143 is applied.

43490 The *mknodat()* function is added from The Open Group Technical Standard, 2006, Extended API
43491 Set Part 2.

43492 Changes are made related to support for finegrained timestamps.

43493 Changes are made to allow a directory to be opened for searching.

43494 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0388 [324], XSH/TC1-2008/0389 [461], +
43495 XSH/TC1-2008/0390 [146,435], XSH/TC1-2008/0391 [278], and XSH/TC1-2008/0392 [278] are +
43496 applied.

43497 **NAME**

43498 mkstemp — create a unique directory

43499 **SYNOPSIS**43500 CX `#include <stdlib.h>`43501 `int mkstemp(char *template);`43502 **DESCRIPTION**43503 Refer to *mkdtemp()*.

43504 **NAME**

43505 mktime — convert broken-down time into time since the Epoch

43506 **SYNOPSIS**

43507 #include <time.h>

43508 time_t mktime(struct tm *timeptr);

43509 **DESCRIPTION**

43510 CX The functionality described on this reference page is aligned with the ISO C standard. Any
 43511 conflict between the requirements described here and the ISO C standard is unintentional. This
 43512 volume of POSIX.1-2008 defers to the ISO C standard.

43513 The *mktime()* function shall convert the broken-down time, expressed as local time, in the
 43514 structure pointed to by *timeptr*, into a time since the Epoch value with the same encoding as that
 43515 of the values returned by *time()*. The original values of the *tm_wday* and *tm_yday* components of
 43516 the structure are ignored, and the original values of the other components are not restricted to
 43517 the ranges described in <time.h>.

43518 CX A positive or 0 value for *tm_isdst* shall cause *mktime()* to presume initially that Daylight Savings
 43519 Time, respectively, is or is not in effect for the specified time. A negative value for *tm_isdst* shall
 43520 cause *mktime()* to attempt to determine whether Daylight Savings Time is in effect for the
 43521 specified time.

43522 Local timezone information shall be set as though *mktime()* called *tzset()*.

43523 The relationship between the **tm** structure (defined in the <time.h> header) and the time in
 43524 seconds since the Epoch is that the result shall be as specified in the expression given in the
 43525 definition of seconds since the Epoch (see XBD [Section 4.15](#), on page 113) corrected for timezone
 43526 and any seasonal time adjustments, where the names in the structure and in the expression
 43527 correspond.

43528 Upon successful completion, the values of the *tm_wday* and *tm_yday* components of the structure
 43529 shall be set appropriately, and the other components are set to represent the specified time since
 43530 the Epoch, but with their values forced to the ranges indicated in the <time.h> entry; the final
 43531 value of *tm_mday* shall not be set until *tm_mon* and *tm_year* are determined.

43532 **RETURN VALUE**

43533 The *mktime()* function shall return the specified time since the Epoch encoded as a value of type
 43534 **time_t**. If the time since the Epoch cannot be represented, the function shall return the value
 43535 CX (**time_t**)-1 and set *errno* to indicate the error.

43536 **ERRORS**43537 The *mktime()* function shall fail if:

43538 CX [EOVERFLOW] The result cannot be represented.

43539 **EXAMPLES**

43540 What day of the week is July 4, 2001?

43541 #include <stdio.h>

43542 #include <time.h>

43543 struct tm time_str;

43544 char daybuf[20];

43545 int main(void)

43546 {

43547 time_str.tm_year = 2001 - 1900;

```

43548     time_str.tm_mon = 7 - 1;
43549     time_str.tm_mday = 4;
43550     time_str.tm_hour = 0;
43551     time_str.tm_min = 0;
43552     time_str.tm_sec = 1;
43553     time_str.tm_isdst = -1;
43554     if (mktime(&time_str) == -1)
43555         (void)puts("-unknown-");
43556     else {
43557         (void)strftime(daybuf, sizeof(daybuf), "%A", &time_str);
43558         (void)puts(daybuf);
43559     }
43560     return 0;
43561 }

```

43562 APPLICATION USAGE

43563 None.

43564 RATIONALE

43565 None.

43566 FUTURE DIRECTIONS

43567 None.

43568 SEE ALSO

43569 [asctime\(\)](#), [clock\(\)](#), [ctime\(\)](#), [difftime\(\)](#), [gmtime\(\)](#), [localtime\(\)](#), [strftime\(\)](#), [strptime\(\)](#), [time\(\)](#), [tzset\(\)](#),
43570 [utime\(\)](#)

43571 XBD [Section 4.15](#) (on page 113), [<time.h>](#)

43572 CHANGE HISTORY

43573 First released in Issue 3. Included for alignment with the POSIX.1-1988 standard and the ANSI C
43574 standard.

43575 Issue 6

43576 Extensions beyond the ISO C standard are marked.

43577 IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/58 is applied, updating the RETURN
43578 VALUE and ERRORS sections to add the optional [EOVERFLOW] error as a CX extension.

43579 IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/59 is applied, adding the [tzset\(\)](#) function
43580 to the SEE ALSO section.

43581 Issue 7

43582 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0393 [104] is applied.

+

NAME

mlock, munlock — lock or unlock a range of process address space (**REALTIME**)

SYNOPSIS

```
#include <sys/mman.h>

int mlock(const void *addr, size_t len);
int munlock(const void *addr, size_t len);
```

DESCRIPTION

The *mlock()* function shall cause those whole pages containing any part of the address space of the process starting at address *addr* and continuing for *len* bytes to be memory-resident until unlocked or until the process exits or *execs* another process image. The implementation may require that *addr* be a multiple of {PAGESIZE}.

The *munlock()* function shall unlock those whole pages containing any part of the address space of the process starting at address *addr* and continuing for *len* bytes, regardless of how many times *mlock()* has been called by the process for any of the pages in the specified range. The implementation may require that *addr* be a multiple of {PAGESIZE}.

If any of the pages in the range specified to a call to *munlock()* are also mapped into the address spaces of other processes, any locks established on those pages by another process are unaffected by the call of this process to *munlock()*. If any of the pages in the range specified by a call to *munlock()* are also mapped into other portions of the address space of the calling process outside the range specified, any locks established on those pages via the other mappings are also unaffected by this call.

Upon successful return from *mlock()*, pages in the specified range shall be locked and memory-resident. Upon successful return from *munlock()*, pages in the specified range shall be unlocked with respect to the address space of the process. Memory residency of unlocked pages is unspecified.

Appropriate privileges are required to lock process memory with *mlock()*.

RETURN VALUE

Upon successful completion, the *mlock()* and *munlock()* functions shall return a value of zero. Otherwise, no change is made to any locks in the address space of the process, and the function shall return a value of -1 and set *errno* to indicate the error.

ERRORS

The *mlock()* and *munlock()* functions shall fail if:

[ENOMEM] Some or all of the address range specified by the *addr* and *len* arguments does not correspond to valid mapped pages in the address space of the process.

The *mlock()* function shall fail if:

[EAGAIN] Some or all of the memory identified by the operation could not be locked when the call was made.

The *mlock()* and *munlock()* functions may fail if:

[EINVAL] The *addr* argument is not a multiple of {PAGESIZE}.

The *mlock()* function may fail if:

[ENOMEM] Locking the pages mapped by the specified range would exceed an implementation-defined limit on the amount of memory that the process may lock.

43626 [EPERM] The calling process does not have appropriate privileges to perform the
43627 requested operation.

43628 **EXAMPLES**

43629 None.

43630 **APPLICATION USAGE**

43631 None.

43632 **RATIONALE**

43633 None.

43634 **FUTURE DIRECTIONS**

43635 None.

43636 **SEE ALSO**

43637 *exec, exit(), fork(), mlockall(), munmap()*

43638 XBD <sys/mman.h>

43639 **CHANGE HISTORY**

43640 First released in Issue 5. Included for alignment with the POSIX Realtime Extension.

43641 **Issue 6**

43642 The *mlock()* and *munlock()* functions are marked as part of the Range Memory Locking option.

43643 The [ENOSYS] error condition has been removed as stubs need not be provided if an
43644 implementation does not support the Range Memory Locking option.

NAME

mlockall, munlockall — lock/unlock the address space of a process (**REALTIME**)

SYNOPSIS

```
ML      #include <sys/mman.h>

int mlockall(int flags);
int munlockall(void);
```

DESCRIPTION

The *mlockall()* function shall cause all of the pages mapped by the address space of a process to be memory-resident until unlocked or until the process exits or *execs* another process image. The *flags* argument determines whether the pages to be locked are those currently mapped by the address space of the process, those that are mapped in the future, or both. The *flags* argument is constructed from the bitwise-inclusive OR of one or more of the following symbolic constants, defined in *<sys/mman.h>*:

MCL_CURRENT Lock all of the pages currently mapped into the address space of the process.

MCL_FUTURE Lock all of the pages that become mapped into the address space of the process in the future, when those mappings are established.

If **MCL_FUTURE** is specified, and the automatic locking of future mappings eventually causes the amount of locked memory to exceed the amount of available physical memory or any other implementation-defined limit, the behavior is implementation-defined. The manner in which the implementation informs the application of these situations is also implementation-defined.

The *munlockall()* function shall unlock all currently mapped pages of the address space of the process. Any pages that become mapped into the address space of the process after a call to *munlockall()* shall not be locked, unless there is an intervening call to *mlockall()* specifying **MCL_FUTURE** or a subsequent call to *mlockall()* specifying **MCL_CURRENT**. If pages mapped into the address space of the process are also mapped into the address spaces of other processes and are locked by those processes, the locks established by the other processes shall be unaffected by a call by this process to *munlockall()*.

Upon successful return from the *mlockall()* function that specifies **MCL_CURRENT**, all currently mapped pages of the address space of the process shall be memory-resident and locked. Upon return from the *munlockall()* function, all currently mapped pages of the address space of the process shall be unlocked with respect to the address space of the process. The memory residency of unlocked pages is unspecified.

Appropriate privileges are required to lock process memory with *mlockall()*.

RETURN VALUE

Upon successful completion, the *mlockall()* function shall return a value of zero. Otherwise, no additional memory shall be locked, and the function shall return a value of *-1* and set *errno* to indicate the error. The effect of failure of *mlockall()* on previously existing locks in the address space is unspecified.

If it is supported by the implementation, the *munlockall()* function shall always return a value of zero. Otherwise, the function shall return a value of *-1* and set *errno* to indicate the error.

ERRORS

The *mlockall()* function shall fail if:

[EAGAIN] Some or all of the memory identified by the operation could not be locked when the call was made.

43689	[EINVAL]	The <i>flags</i> argument is zero, or includes unimplemented flags.
43690		The <i>mlockall()</i> function may fail if:
43691	[ENOMEM]	Locking all of the pages currently mapped into the address space of the process would exceed an implementation-defined limit on the amount of memory that the process may lock.
43692		
43693		
43694	[EPERM]	The calling process does not have appropriate privileges to perform the requested operation.
43695		
43696	EXAMPLES	
43697	None.	
43698	APPLICATION USAGE	
43699	None.	
43700	RATIONALE	
43701	None.	
43702	FUTURE DIRECTIONS	
43703	None.	
43704	SEE ALSO	
43705	<i>exec</i> , <i>exit()</i> , <i>fork()</i> , <i>mlock()</i> , <i>munmap()</i>	
43706	XBD < <i>sys/mman.h</i> >	
43707	CHANGE HISTORY	
43708	First released in Issue 5. Included for alignment with the POSIX Realtime Extension.	
43709	Issue 6	
43710	The <i>mlockall()</i> and <i>munlockall()</i> functions are marked as part of the Process Memory Locking option.	
43711		
43712	The [ENOSYS] error condition has been removed as stubs need not be provided if an implementation does not support the Process Memory Locking option.	
43713		

43714 **NAME**43715 `mmap` — map pages of memory43716 **SYNOPSIS**43717 `#include <sys/mman.h>`

```
43718 void *mmap(void *addr, size_t len, int prot, int flags,
43719           int fildes, off_t off);
```

43720 **DESCRIPTION**

43721 The `mmap()` function shall establish a mapping between an address space of a process and a
 43722 memory object.

43723 The `mmap()` function shall be supported for the following memory objects:

- 43724 • Regular files
- 43725 SHM • Shared memory objects
- 43726 TYM • Typed memory objects

43727 Support for any other type of file is unspecified.

43728 The format of the call is as follows:

```
43729 pa=mmap(addr, len, prot, flags, fildes, off);
```

43730 The `mmap()` function shall establish a mapping between the address space of the process at an
 43731 address *pa* for *len* bytes to the memory object represented by the file descriptor *fildes* at offset *off*
 43732 for *len* bytes. The value of *pa* is an implementation-defined function of the parameter *addr* and
 43733 the values of *flags*, further described below. A successful `mmap()` call shall return *pa* as its result.
 43734 The address range starting at *pa* and continuing for *len* bytes shall be legitimate for the possible
 43735 (not necessarily current) address space of the process. The range of bytes starting at *off* and
 43736 continuing for *len* bytes shall be legitimate for the possible (not necessarily current) offsets in the
 43737 memory object represented by *fildes*.

43738 TYM If *fildes* represents a typed memory object opened with either the
 43739 POSIX_TYPED_MEM_ALLOCATE flag or the POSIX_TYPED_MEM_ALLOCATE_CONTIG
 43740 flag, the memory object to be mapped shall be that portion of the typed memory object allocated
 43741 by the implementation as specified below. In this case, if *off* is non-zero, the behavior of `mmap()`
 43742 is undefined. If *fildes* refers to a valid typed memory object that is not accessible from the calling
 43743 process, `mmap()` shall fail.

43744 The mapping established by `mmap()` shall replace any previous mappings for those whole pages
 43745 containing any part of the address space of the process starting at *pa* and continuing for *len*
 43746 bytes.

43747 If the size of the mapped file changes after the call to `mmap()` as a result of some other operation
 43748 on the mapped file, the effect of references to portions of the mapped region that correspond to
 43749 added or removed portions of the file is unspecified.

43750 If *len* is zero, `mmap()` shall fail and no mapping shall be established.

43751 The parameter *prot* determines whether read, write, execute, or some combination of accesses
 43752 are permitted to the data being mapped. The *prot* shall be either PROT_NONE or the bitwise-
 43753 inclusive OR of one or more of the other flags in the following table, defined in the
 43754 `<sys/mman.h>` header.

Symbolic Constant	Description
PROT_READ	Data can be read.
PROT_WRITE	Data can be written.
PROT_EXEC	Data can be executed.
PROT_NONE	Data cannot be accessed.

If an implementation cannot support the combination of access types specified by *prot*, the call to *mmap()* shall fail.

An implementation may permit accesses other than those specified by *prot*; however, the implementation shall not permit a write to succeed where PROT_WRITE has not been set and shall not permit any access where PROT_NONE alone has been set. The implementation shall support at least the following values of *prot*: PROT_NONE, PROT_READ, PROT_WRITE, and the bitwise-inclusive OR of PROT_READ and PROT_WRITE. The file descriptor *fildes* shall have been opened with read permission, regardless of the protection options specified. If PROT_WRITE is specified, the application shall ensure that it has opened the file descriptor *fildes* with write permission unless MAP_PRIVATE is specified in the *flags* parameter as described below.

The parameter *flags* provides other information about the handling of the mapped data. The value of *flags* is the bitwise-inclusive OR of these options, defined in <sys/mman.h>:

Symbolic Constant	Description
MAP_SHARED	Changes are shared.
MAP_PRIVATE	Changes are private.
MAP_FIXED	Interpret <i>addr</i> exactly.

It is implementation-defined whether MAP_FIXED shall be supported. MAP_FIXED shall be supported on XSI-conformant systems.

MAP_SHARED and MAP_PRIVATE describe the disposition of write references to the memory object. If MAP_SHARED is specified, write references shall change the underlying object. If MAP_PRIVATE is specified, modifications to the mapped data by the calling process shall be visible only to the calling process and shall not change the underlying object. It is unspecified whether modifications to the underlying object done after the MAP_PRIVATE mapping is established are visible through the MAP_PRIVATE mapping. Either MAP_SHARED or MAP_PRIVATE can be specified, but not both. The mapping type is retained across *fork()*.

The state of synchronization objects such as mutexes, semaphores, barriers, and conditional variables placed in shared memory mapped with MAP_SHARED becomes undefined when the last region in any process containing the synchronization object is unmapped.

When *fildes* represents a typed memory object opened with either the POSIX_TYPED_MEM_ALLOCATE flag or the POSIX_TYPED_MEM_ALLOCATE_CONTIG flag, *mmap()* shall, if there are enough resources available, map *len* bytes allocated from the corresponding typed memory object which were not previously allocated to any process in any processor that may access that typed memory object. If there are not enough resources available, the function shall fail. If *fildes* represents a typed memory object opened with the POSIX_TYPED_MEM_ALLOCATE_CONTIG flag, these allocated bytes shall be contiguous within the typed memory object. If *fildes* represents a typed memory object opened with the POSIX_TYPED_MEM_ALLOCATE flag, these allocated bytes may be composed of non-contiguous fragments within the typed memory object. If *fildes* represents a typed memory object opened with neither the POSIX_TYPED_MEM_ALLOCATE_CONTIG flag nor the POSIX_TYPED_MEM_ALLOCATE flag, *len* bytes starting at offset *off* within the typed memory object are mapped, exactly as when mapping a file or shared memory object. In this case, if two

processes map an area of typed memory using the same *off* and *len* values and using file descriptors that refer to the same memory pool (either from the same port or from a different port), both processes shall map the same region of storage.

When MAP_FIXED is set in the *flags* argument, the implementation is informed that the value of *pa* shall be *addr*, exactly. If MAP_FIXED is set, *mmap()* may return MAP_FAILED and set *errno* to [EINVAL]. If a MAP_FIXED request is successful, the mapping established by *mmap()* replaces any previous mappings for the pages in the range [*pa*,*pa+len*) of the process.

When MAP_FIXED is not set, the implementation uses *addr* in an implementation-defined manner to arrive at *pa*. The *pa* so chosen shall be an area of the address space that the implementation deems suitable for a mapping of *len* bytes to the file. All implementations interpret an *addr* value of 0 as granting the implementation complete freedom in selecting *pa*, subject to constraints described below. A non-zero value of *addr* is taken to be a suggestion of a process address near which the mapping should be placed. When the implementation selects a value for *pa*, it never places a mapping at address 0, nor does it replace any extant mapping.

If MAP_FIXED is specified and *addr* is non-zero, it shall have the same remainder as the *off* parameter, modulo the page size as returned by *sysconf()* when passed _SC_PAGESIZE or _SC_PAGE_SIZE. The implementation may require that *off* is a multiple of the page size. If MAP_FIXED is specified, the implementation may require that *addr* is a multiple of the page size. The system performs mapping operations over whole pages. Thus, while the parameter *len* need not meet a size or alignment constraint, the system shall include, in any mapping operation, any partial page specified by the address range starting at *pa* and continuing for *len* bytes.

The system shall always zero-fill any partial page at the end of an object. Further, the system shall never write out any modified portions of the last page of an object which are beyond its end. References within the address range starting at *pa* and continuing for *len* bytes to whole pages following the end of an object shall result in delivery of a SIGBUS signal.

An implementation may generate SIGBUS signals when a reference would cause an error in the mapped object, such as out-of-space condition.

The *mmap()* function shall add an extra reference to the file associated with the file descriptor *fd* which is not removed by a subsequent *close()* on that file descriptor. This reference shall be removed when there are no more mappings to the file.

The last data access timestamp of the mapped file may be marked for update at any time between the *mmap()* call and the corresponding *munmap()* call. The initial read or write reference to a mapped region shall cause the file's last data access timestamp to be marked for update if it has not already been marked for update.

The last data modification and last file status change timestamps of a file that is mapped with MAP_SHARED and PROT_WRITE shall be marked for update at some point in the interval between a write reference to the mapped region and the next call to *msync()* with MS_ASYNC or MS_SYNC for that portion of the file by any process. If there is no such call and if the underlying file is modified as a result of a write reference, then these timestamps shall be marked for update at some time after the write reference.

There may be implementation-defined limits on the number of memory regions that can be mapped (per process or per system).

If such a limit is imposed, whether the number of memory regions that can be mapped by a process is decreased by the use of *shmat()* is implementation-defined.

If *mmap()* fails for reasons other than [EBADF], [EINVAL], or [ENOTSUP], some of the

43848 mappings in the address range starting at *addr* and continuing for *len* bytes may have been
 43849 unmapped.

43850 RETURN VALUE

43851 Upon successful completion, the *mmap()* function shall return the address at which the mapping
 43852 was placed (*pa*); otherwise, it shall return a value of MAP_FAILED and set *errno* to indicate the
 43853 error. The symbol MAP_FAILED is defined in the **<sys/mman.h>** header. No successful return
 43854 from *mmap()* shall return the value MAP_FAILED.

43855 ERRORS

43856 The *mmap()* function shall fail if:

43857		[EACCES]	The <i>fildev</i> argument is not open for read, regardless of the protection specified, 43858 or <i>fildev</i> is not open for write and PROT_WRITE was specified for a 43859 MAP_SHARED type mapping.
43860	ML	[EAGAIN]	The mapping could not be locked in memory, if required by <i>mlockall()</i> , due to 43861 a lack of resources.
43862		[EBADF]	The <i>fildev</i> argument is not a valid open file descriptor.
43863		[EINVAL]	The value of <i>len</i> is zero.
43864		[EINVAL]	The value of <i>flags</i> is invalid (neither MAP_PRIVATE nor MAP_SHARED is 43865 set).
43866		[EMFILE]	The number of mapped regions would exceed an implementation-defined 43867 limit (per process or per system).
43868		[ENODEV]	The <i>fildev</i> argument refers to a file whose type is not supported by <i>mmap()</i> .
43869		[ENOMEM]	MAP_FIXED was specified, and the range [<i>addr</i> , <i>addr</i> + <i>len</i>) exceeds that allowed 43870 for the address space of a process; or, if MAP_FIXED was not specified and 43871 there is insufficient room in the address space to effect the mapping.
43872	ML	[ENOMEM]	The mapping could not be locked in memory, if required by <i>mlockall()</i> , 43873 because it would require more space than the system is able to supply.
43874	TYM	[ENOMEM]	Not enough unallocated memory resources remain in the typed memory 43875 object designated by <i>fildev</i> to allocate <i>len</i> bytes.
43876		[ENOTSUP]	MAP_FIXED or MAP_PRIVATE was specified in the <i>flags</i> argument and the 43877 implementation does not support this functionality.
43878			The implementation does not support the combination of accesses requested 43879 in the <i>prot</i> argument.
43880		[ENXIO]	Addresses in the range [<i>off</i> , <i>off</i> + <i>len</i>) are invalid for the object specified by <i>fildev</i> .
43881		[ENXIO]	MAP_FIXED was specified in <i>flags</i> and the combination of <i>addr</i> , <i>len</i> , and <i>off</i> is 43882 invalid for the object specified by <i>fildev</i> .
43883	TYM	[ENXIO]	The <i>fildev</i> argument refers to a typed memory object that is not accessible from 43884 the calling process.
43885		[EOVERFLOW]	The file is a regular file and the value of <i>off</i> plus <i>len</i> exceeds the offset 43886 maximum established in the open file description associated with <i>fildev</i> .

43887 The *mmap()* function may fail if:

43888 [EINVAL] The *addr* argument (if MAP_FIXED was specified) or *off* is not a multiple of the
 43889 page size as returned by *sysconf()*, or is considered invalid by the
 43890 implementation.

43891 EXAMPLES

43892 None.

43893 APPLICATION USAGE

43894 Use of *mmap()* may reduce the amount of memory available to other memory allocation
 43895 functions.

43896 Use of MAP_FIXED may result in unspecified behavior in further use of *malloc()* and *shmat()*.
 43897 The use of MAP_FIXED is discouraged, as it may prevent an implementation from making the
 43898 most effective use of resources. Most implementations require that *off* and *addr* are multiples of
 43899 the page size as returned by *sysconf()*.

43900 The application must ensure correct synchronization when using *mmap()* in conjunction with
 43901 any other file access method, such as *read()* and *write()*, standard input/output, and *shmat()*.

43902 The *mmap()* function allows access to resources via address space manipulations, instead of
 43903 *read()/write()*. Once a file is mapped, all a process has to do to access it is use the data at the
 43904 address to which the file was mapped. So, using pseudo-code to illustrate the way in which an
 43905 existing program might be changed to use *mmap()*, the following:

```
43906 fildes = open(...)
43907 lseek(fildes, some_offset)
43908 read(fildes, buf, len)
43909 /* Use data in buf. */
```

43910 becomes:

```
43911 fildes = open(...)
43912 address = mmap(0, len, PROT_READ, MAP_PRIVATE, fildes, some_offset)
43913 /* Use data at address. */
```

43914 RATIONALE

43915 After considering several other alternatives, it was decided to adopt the *mmap()* definition
 43916 found in SVR4 for mapping memory objects into process address spaces. The SVR4 definition is
 43917 minimal, in that it describes only what has been built, and what appears to be necessary for a
 43918 general and portable mapping facility.

43919 Note that while *mmap()* was first designed for mapping files, it is actually a general-purpose
 43920 mapping facility. It can be used to map any appropriate object, such as memory, files, devices,
 43921 and so on, into the address space of a process.

43922 When a mapping is established, it is possible that the implementation may need to map more
 43923 than is requested into the address space of the process because of hardware requirements. An
 43924 application, however, cannot count on this behavior. Implementations that do not use a paged
 43925 architecture may simply allocate a common memory region and return the address of it; such
 43926 implementations probably do not allocate any more than is necessary. References past the end of
 43927 the requested area are unspecified.

43928 If an application requests a mapping that would overlay existing mappings in the process, it
 43929 might be desirable that an implementation detect this and inform the application. However, the
 43930 default, portable (not MAP_FIXED) operation does not overlay existing mappings. On the other
 43931 hand, if the program specifies a fixed address mapping (which requires some implementation

knowledge to determine a suitable address, if the function is supported at all), then the program is presumed to be successfully managing its own address space and should be trusted when it asks to map over existing data structures. Furthermore, it is also desirable to make as few system calls as possible, and it might be considered onerous to require an *munmap()* before an *mmap()* to the same address range. This volume of POSIX.1-2008 specifies that the new mappings replace any existing mappings, following existing practice in this regard.

It is not expected that all hardware implementations are able to support all combinations of permissions at all addresses. Implementations are required to disallow write access to mappings without write permission and to disallow access to mappings without any access permission. Other than these restrictions, implementations may allow access types other than those requested by the application. For example, if the application requests only *PROT_WRITE*, the implementation may also allow read access. A call to *mmap()* fails if the implementation cannot support allowing all the access requested by the application. For example, some implementations cannot support a request for both write access and execute access simultaneously. All implementations must support requests for no access, read access, write access, and both read and write access. Strictly conforming code must only rely on the required checks. These restrictions allow for portability across a wide range of hardware.

The *MAP_FIXED* address treatment is likely to fail for non-page-aligned values and for certain architecture-dependent address ranges. Conforming implementations cannot count on being able to choose address values for *MAP_FIXED* without utilizing non-portable, implementation-defined knowledge. Nonetheless, *MAP_FIXED* is provided as a standard interface conforming to existing practice for utilizing such knowledge when it is available.

Similarly, in order to allow implementations that do not support virtual addresses, support for directly specifying any mapping addresses via *MAP_FIXED* is not required and thus a conforming application may not count on it.

The *MAP_PRIVATE* function can be implemented efficiently when memory protection hardware is available. When such hardware is not available, implementations can implement such “mappings” by simply making a real copy of the relevant data into process private memory, though this tends to behave similarly to *read()*.

The function has been defined to allow for many different models of using shared memory. However, all uses are not equally portable across all machine architectures. In particular, the *mmap()* function allows the system as well as the application to specify the address at which to map a specific region of a memory object. The most portable way to use the function is always to let the system choose the address, specifying *NULL* as the value for the argument *addr* and not to specify *MAP_FIXED*.

If it is intended that a particular region of a memory object be mapped at the same address in a group of processes (on machines where this is even possible), then *MAP_FIXED* can be used to pass in the desired mapping address. The system can still be used to choose the desired address if the first such mapping is made without specifying *MAP_FIXED*, and then the resulting mapping address can be passed to subsequent processes for them to pass in via *MAP_FIXED*. The availability of a specific address range cannot be guaranteed, in general.

The *mmap()* function can be used to map a region of memory that is larger than the current size of the object. Memory access within the mapping but beyond the current end of the underlying objects may result in *SIGBUS* signals being sent to the process. The reason for this is that the size of the object can be manipulated by other processes and can change at any moment. The implementation should tell the application that a memory reference is outside the object where this can be detected; otherwise, written data may be lost and read data may not reflect actual data in the object.

Note that references beyond the end of the object do not extend the object as the new end cannot be determined precisely by most virtual memory hardware. Instead, the size can be directly manipulated by *ftruncate()*.

Process memory locking does apply to shared memory regions, and the MEMLOCK_FUTURE argument to *mlockall()* can be relied upon to cause new shared memory regions to be automatically locked.

Existing implementations of *mmap()* return the value `-1` when unsuccessful. Since the casting of this value to type `void *` cannot be guaranteed by the ISO C standard to be distinct from a successful value, this volume of POSIX.1-2008 defines the symbol `MAP_FAILED`, which a conforming implementation does not return as the result of a successful call.

FUTURE DIRECTIONS

None.

SEE ALSO

exec, *fcntl()*, *fork()*, *lockf()*, *msync()*, *munmap()*, *mprotect()*, *posix_typed_mem_open()*, *shmat()*, *sysconf()*

XBD [`<sys/mman.h>`](#)

CHANGE HISTORY

First released in Issue 4, Version 2.

Issue 5

Moved from X/OPEN UNIX extension to BASE.

Aligned with *mmap()* in the POSIX Realtime Extension as follows:

- The DESCRIPTION is extensively reworded.
- The [EAGAIN] and [ENOTSUP] mandatory error conditions are added.
- New cases of [ENOMEM] and [ENXIO] are added as mandatory error conditions.
- The value returned on failure is the value of the constant `MAP_FAILED`; this was previously defined as `-1`.

Large File Summit extensions are added.

Issue 6

The *mmap()* function is marked as part of the Memory Mapped Files option.

The Open Group Corrigendum U028/6 is applied, changing `(void *)-1` to `MAP_FAILED`.

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The DESCRIPTION is updated to describe the use of `MAP_FIXED`.
- The DESCRIPTION is updated to describe the addition of an extra reference to the file associated with the file descriptor passed to *mmap()*.
- The DESCRIPTION is updated to state that there may be implementation-defined limits on the number of memory regions that can be mapped.
- The DESCRIPTION is updated to describe constraints on the alignment and size of the *off* argument.

- The [EINVAL] and [EMFILE] error conditions are added.

- The [EOVERFLOW] error condition is added. This change is to support large files.

The following changes are made for alignment with the ISO POSIX-1: 1996 standard:

- The DESCRIPTION is updated to describe the cases when MAP_PRIVATE and MAP_FIXED need not be supported.

The following changes are made for alignment with IEEE Std 1003.1j-2000:

- Semantics for typed memory objects are added to the DESCRIPTION.
- New [ENOMEM] and [ENXIO] errors are added to the ERRORS section.
- The *posix_typed_mem_open()* function is added to the SEE ALSO section.

The normative text is updated to avoid use of the term “must” for application requirements.

IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/34 is applied, changing the margin code in the SYNOPSIS from MF|SHM to MC3 (notation for MF|SHM|TYM).

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/60 is applied, updating the DESCRIPTION and ERRORS sections to add the [EINVAL] error when *len* is zero.

Issue 7

Austin Group Interpretations 1003.1-2001 #078 and #079 are applied, clarifying page alignment requirements and adding a note about the state of synchronization objects becoming undefined when a shared region is unmapped.

Functionality relating to the Memory Protection and Memory Mapped Files options is moved to the Base.

Changes are made related to support for finegrained timestamps.

44040 NAME

44041 modf, modff, modfl — decompose a floating-point number

44042 SYNOPSIS

44043 #include <math.h>

44044 double modf(double *x*, double **iptr*);44045 float modff(float *value*, float **iptr*);44046 long double modfl(long double *value*, long double **iptr*);

44047 DESCRIPTION

44048 CX The functionality described on this reference page is aligned with the ISO C standard. Any
 44049 conflict between the requirements described here and the ISO C standard is unintentional. This
 44050 volume of POSIX.1-2008 defers to the ISO C standard.

44051 These functions shall break the argument *x* into integral and fractional parts, each of which has
 44052 the same sign as the argument. It stores the integral part as a **double** (for the *modf()* function), a
 44053 **float** (for the *modff()* function), or a **long double** (for the *modfl()* function), in the object pointed
 44054 to by *iptr*.

44055 RETURN VALUE

44056 Upon successful completion, these functions shall return the signed fractional part of *x*.44057 MX If *x* is NaN, a NaN shall be returned, and **iptr* shall be set to a NaN.44058 If *x* is $\pm\text{Inf}$, ± 0 shall be returned, and **iptr* shall be set to $\pm\text{Inf}$.

44059 ERRORS

44060 No errors are defined.

44061 EXAMPLES

44062 None.

44063 APPLICATION USAGE

44064 The *modf()* function computes the function result and **iptr* such that:44065 *a* = modf(*x*, *iptr*) ;44066 *x* == *a* + **iptr* ;

44067 allowing for the usual floating-point inaccuracies.

44068 RATIONALE

44069 None.

44070 FUTURE DIRECTIONS

44071 None.

44072 SEE ALSO

44073 *frexp()*, *isnan()*, *ldexp()*

44074 XBD <math.h>

44075 CHANGE HISTORY

44076 First released in Issue 1. Derived from Issue 1 of the SVID.

44077 Issue 5

44078 The DESCRIPTION is updated to indicate how an application should check for an error. This
 44079 text was previously published in the APPLICATION USAGE section.

44080 **Issue 6**

44081 The *modff()* and *modfl()* functions are added for alignment with the ISO/IEC 9899:1999
44082 standard.

44083 The DESCRIPTION, RETURN VALUE, ERRORS, and APPLICATION USAGE sections are
44084 revised to align with the ISO/IEC 9899:1999 standard.

44085 IEC 60559:1989 standard floating-point extensions over the ISO/IEC 9899:1999 standard are
44086 marked.

44087 IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/35 is applied, correcting the code example
44088 in the APPLICATION USAGE section.

44089 **NAME**44090 `mprotect` — set protection of memory mapping44091 **SYNOPSIS**44092 `#include <sys/mman.h>`44093 `int mprotect(void *addr, size_t len, int prot);`44094 **DESCRIPTION**

44095 The `mprotect()` function shall change the access protections to be that specified by *prot* for those
 44096 whole pages containing any part of the address space of the process starting at address *addr* and
 44097 continuing for *len* bytes. The parameter *prot* determines whether read, write, execute, or some
 44098 combination of accesses are permitted to the data being mapped. The *prot* argument should be
 44099 either `PROT_NONE` or the bitwise-inclusive OR of one or more of `PROT_READ`, `PROT_WRITE`,
 44100 and `PROT_EXEC`.

44101 If an implementation cannot support the combination of access types specified by *prot*, the call to
 44102 `mprotect()` shall fail.

44103 An implementation may permit accesses other than those specified by *prot*; however, no
 44104 implementation shall permit a write to succeed where `PROT_WRITE` has not been set or shall
 44105 permit any access where `PROT_NONE` alone has been set. Implementations shall support at
 44106 least the following values of *prot*: `PROT_NONE`, `PROT_READ`, `PROT_WRITE`, and the bitwise-
 44107 inclusive OR of `PROT_READ` and `PROT_WRITE`. If `PROT_WRITE` is specified, the application
 44108 shall ensure that it has opened the mapped objects in the specified address range with write
 44109 permission, unless `MAP_PRIVATE` was specified in the original mapping, regardless of whether
 44110 the file descriptors used to map the objects have since been closed.

44111 The implementation may require that *addr* be a multiple of the page size as returned by
 44112 `sysconf()`.

44113 The behavior of this function is unspecified if the mapping was not established by a call to
 44114 `mmap()`.

44115 When `mprotect()` fails for reasons other than `[EINVAL]`, the protections on some of the pages in
 44116 the range `[addr,addr+len)` may have been changed.

44117 **RETURN VALUE**

44118 Upon successful completion, `mprotect()` shall return 0; otherwise, it shall return `-1` and set *errno*
 44119 to indicate the error.

44120 **ERRORS**

44121 The `mprotect()` function shall fail if:

44122 `[EACCES]` The *prot* argument specifies a protection that violates the access permission the
 44123 process has to the underlying memory object.

44124 `[EAGAIN]` The *prot* argument specifies `PROT_WRITE` over a `MAP_PRIVATE` mapping
 44125 and there are insufficient memory resources to reserve for locking the private
 44126 page.

44127 `[ENOMEM]` Addresses in the range `[addr,addr+len)` are invalid for the address space of a
 44128 process, or specify one or more pages which are not mapped.

44129 `[ENOMEM]` The *prot* argument specifies `PROT_WRITE` on a `MAP_PRIVATE` mapping, and
 44130 it would require more space than the system is able to supply for locking the
 44131 private pages, if required.

44132 [ENOTSUP] The implementation does not support the combination of accesses requested
 44133 in the *prot* argument.

44134 The *mprotect()* function may fail if:

44135 [EINVAL] The *addr* argument is not a multiple of the page size as returned by *sysconf()*.

44136 EXAMPLES

44137 None.

44138 APPLICATION USAGE

44139 Most implementations require that *addr* is a multiple of the page size as returned by *sysconf()*.

44140 RATIONALE

44141 None.

44142 FUTURE DIRECTIONS

44143 None.

44144 SEE ALSO

44145 *mmap()*, *sysconf()*

44146 XBD <sys/mman.h>

44147 CHANGE HISTORY

44148 First released in Issue 4, Version 2.

44149 Issue 5

44150 Moved from X/OPEN UNIX extension to BASE.

44151 Aligned with *mprotect()* in the POSIX Realtime Extension as follows:

- 44152 • The DESCRIPTION is largely reworded.
- 44153 • [ENOTSUP] and a second form of [ENOMEM] are added as mandatory error conditions.
- 44154 • [EAGAIN] is moved from the optional to the mandatory error conditions.

44155 Issue 6

44156 The *mprotect()* function is marked as part of the Memory Protection option.

44157 The following new requirements on POSIX implementations derive from alignment with the
 44158 Single UNIX Specification:

- 44159 • The DESCRIPTION is updated to state that implementations require *addr* to be a multiple
 44160 of the page size as returned by *sysconf()*.
- 44161 • The [EINVAL] error condition is added.

44162 The normative text is updated to avoid use of the term “must” for application requirements.

44163 Issue 7

44164 SD5-XSH-ERN-22 is applied, deleting erroneous APPLICATION USAGE.

44165 Austin Group Interpretation 1003.1-2001 #078 is applied, clarifying page alignment
 44166 requirements.

44167 The *mprotect()* function is moved from the Memory Protection option to the Base.

44168 **NAME**44169 mq_close — close a message queue (**REALTIME**)44170 **SYNOPSIS**

```
44171 MSG    #include <mqueue.h>
44172        int mq_close(mqd_t mqdes);
```

44173 **DESCRIPTION**

44174 The *mq_close()* function shall remove the association between the message queue descriptor,
 44175 *mqdes*, and its message queue. The results of using this message queue descriptor after successful
 44176 return from this *mq_close()*, and until the return of this message queue descriptor from a
 44177 subsequent *mq_open()*, are undefined.

44178 If the process has successfully attached a notification request to the message queue via this
 44179 *mqdes*, this attachment shall be removed, and the message queue is available for another process
 44180 to attach for notification.

44181 **RETURN VALUE**

44182 Upon successful completion, the *mq_close()* function shall return a value of zero; otherwise, the
 44183 function shall return a value of -1 and set *errno* to indicate the error.

44184 **ERRORS**

44185 The *mq_close()* function shall fail if:

44186 [EBADF] The *mqdes* argument is not a valid message queue descriptor.

44187 **EXAMPLES**

44188 None.

44189 **APPLICATION USAGE**

44190 None.

44191 **RATIONALE**

44192 None.

44193 **FUTURE DIRECTIONS**

44194 None.

44195 **SEE ALSO**

44196 *mq_open()*, *mq_unlink()*, *msgctl()*, *msgget()*, *msgrcv()*, *msgsnd()*

44197 XBD [<mqueue.h>](#)

44198 **CHANGE HISTORY**

44199 First released in Issue 5. Included for alignment with the POSIX Realtime Extension.

44200 **Issue 6**

44201 The *mq_close()* function is marked as part of the Message Passing option.

44202 The [ENOSYS] error condition has been removed as stubs need not be provided if an
 44203 implementation does not support the Message Passing option.

NAME

mq_getattr — get message queue attributes (**REALTIME**)

SYNOPSIS

```
#include <mqueue.h>

int mq_getattr(mqd_t mqdes, struct mq_attr *mqstat);
```

DESCRIPTION

The *mq_getattr()* function shall obtain status information and attributes of the message queue and the open message queue description associated with the message queue descriptor.

The *mqdes* argument specifies a message queue descriptor.

The results shall be returned in the **mq_attr** structure referenced by the *mqstat* argument.

Upon return, the following members shall have the values associated with the open message queue description as set when the message queue was opened and as modified by subsequent *mq_setattr()* calls: *mq_flags*.

The following attributes of the message queue shall be returned as set at message queue creation: *mq_maxmsg*, *mq_msgsize*.

Upon return, the following members within the **mq_attr** structure referenced by the *mqstat* argument shall be set to the current state of the message queue:

mq_curmsgs The number of messages currently on the queue.

RETURN VALUE

Upon successful completion, the *mq_getattr()* function shall return zero. Otherwise, the function shall return -1 and set *errno* to indicate the error.

ERRORS

The *mq_getattr()* function may fail if:

[EBADF] The *mqdes* argument is not a valid message queue descriptor.

EXAMPLES

See *mq_notify()*.

APPLICATION USAGE

None.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

mq_notify(), *mq_open()*, *mq_send()*, *mq_setattr()*, *msgctl()*, *msgget()*, *msgrcv()*, *msgsnd()*

XBD **<mqueue.h>**

CHANGE HISTORY

First released in Issue 5. Included for alignment with the POSIX Realtime Extension.

Issue 6

The *mq_getattr()* function is marked as part of the Message Passing option.

The [ENOSYS] error condition has been removed as stubs need not be provided if an implementation does not support the Message Passing option.

44245	The <i>mq_timedsend()</i> function is added to the SEE ALSO section for alignment with IEEE Std
44246	1003.1d-1999.
44247	IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/61 is applied, updating the ERRORS
44248	section to change the [EBADF] error from mandatory to optional.

44249 **NAME**44250 mq_notify — notify process that a message is available (**REALTIME**)44251 **SYNOPSIS**

```
44252 MSG    #include <mqueue.h>
44253        int mq_notify(mqd_t mqdes, const struct sigevent *notification);
```

44254 **DESCRIPTION**

44255 If the argument *notification* is not NULL, this function shall register the calling process to be
 44256 notified of message arrival at an empty message queue associated with the specified message
 44257 queue descriptor, *mqdes*. The notification specified by the *notification* argument shall be sent to
 44258 the process when the message queue transitions from empty to non-empty. At any time, only
 44259 one process may be registered for notification by a message queue. If the calling process or any
 44260 other process has already registered for notification of message arrival at the specified message
 44261 queue, subsequent attempts to register for that message queue shall fail.

44262 If *notification* is NULL and the process is currently registered for notification by the specified
 44263 message queue, the existing registration shall be removed.

44264 When the notification is sent to the registered process, its registration shall be removed. The
 44265 message queue shall then be available for registration.

44266 If a process has registered for notification of message arrival at a message queue and some
 44267 thread is blocked in *mq_receive()* or *mq_timedreceive()* waiting to receive a message when a
 44268 message arrives at the queue, the arriving message shall satisfy the appropriate *mq_receive()* or
 44269 *mq_timedreceive()*, respectively. The resulting behavior is as if the message queue remains empty,
 44270 and no notification shall be sent.

44271 **RETURN VALUE**

44272 Upon successful completion, the *mq_notify()* function shall return a value of zero; otherwise, the
 44273 function shall return a value of -1 and set *errno* to indicate the error.

44274 **ERRORS**

44275 The *mq_notify()* function shall fail if:

44276 [EBADF] The *mqdes* argument is not a valid message queue descriptor.

44277 [EBUSY] A process is already registered for notification by the message queue.

44278 The *mq_notify()* function may fail if:

44279 [EINVAL] The *notification* argument is NULL and the process is currently not registered.

44280 **EXAMPLES**

44281 The following program registers a notification request for the message queue named in its
 44282 command-line argument. Notification is performed by creating a thread. The thread executes a
 44283 function which reads one message from the queue and then terminates the process.

```
44284 #include <pthread.h>
44285 #include <mqueue.h>
44286 #include <assert.h>
44287 #include <stdio.h>
44288 #include <stdlib.h>
44289 #include <unistd.h>

44290 static void                                /* Thread start function */
44291 tfunc(union sigval sv)
44292 {
```

```

44293     struct mq_attr attr;
44294     ssize_t nr;
44295     void *buf;
44296     mqd_t mqdes = *((mqd_t *) sv.sival_ptr);
44297     /* Determine maximum msg size; allocate buffer to receive msg */
44298     if (mq_getattr(mqdes, &attr) == -1) {
44299         perror("mq_getattr");
44300         exit(EXIT_FAILURE);
44301     }
44302     buf = malloc(attr.mq_msgsize);
44303     if (buf == NULL) {
44304         perror("malloc");
44305         exit(EXIT_FAILURE);
44306     }
44307     nr = mq_receive(mqdes, buf, attr.mq_msgsize, NULL);
44308     if (nr == -1) {
44309         perror("mq_receive");
44310         exit(EXIT_FAILURE);
44311     }
44312     printf("Read %ld bytes from message queue\n", (long) nr);
44313     free(buf);
44314     exit(EXIT_SUCCESS);          /* Terminate the process */
44315 }
44316
44317 int
44318 main(int argc, char *argv[])
44319 {
44320     mqd_t mqdes;
44321     struct sigevent not;
44322     assert(argc == 2);
44323     mqdes = mq_open(argv[1], O_RDONLY);
44324     if (mqdes == (mqd_t) -1) {
44325         perror("mq_open");
44326         exit(EXIT_FAILURE);
44327     }
44328     not.sigev_notify = SIGEV_THREAD;
44329     not.sigev_notify_function = tfunc;
44330     not.sigev_notify_attributes = NULL;
44331     not.sigev_value.sival_ptr = &mqdes;    /* Arg. to thread func. */
44332     if (mq_notify(mqdes, &not) == -1) {
44333         perror("mq_notify");
44334         exit(EXIT_FAILURE);
44335     }
44336     pause();    /* Process will be terminated by thread function */
44337 }

```


44337 **APPLICATION USAGE**

44338 None.

44339 **RATIONALE**

44340 None.

44341 **FUTURE DIRECTIONS**

44342 None.

44343 **SEE ALSO**44344 *mq_open()*, *mq_send()*, *mq_receive()*, *msgctl()*, *msgget()*, *msgrcv()*, *msgsnd()*

44345 XBD <mqqueue.h>

44346 **CHANGE HISTORY**

44347 First released in Issue 5. Included for alignment with the POSIX Realtime Extension.

44348 **Issue 6**44349 The *mq_notify()* function is marked as part of the Message Passing option.44350 The [ENOSYS] error condition has been removed as stubs need not be provided if an
44351 implementation does not support the Message Passing option.44352 The *mq_timedsend()* function is added to the SEE ALSO section for alignment with IEEE Std
44353 1003.1d-1999.44354 **Issue 7**44355 SD5-XSH-ERN-38 is applied, adding the *mq_timedreceive()* function to the DESCRIPTION.

44356 Austin Group Interpretation 1003.1-2001 #032 is applied, adding the [EINVAL] error.

44357 An example is added.

NAME

mq_open — open a message queue (**REALTIME**)

SYNOPSIS

```
#include <mqueue.h>

mqd_t mq_open(const char *name, int oflag, ...);
```

DESCRIPTION

The *mq_open()* function shall establish the connection between a process and a message queue with a message queue descriptor. It shall create an open message queue description that refers to the message queue, and a message queue descriptor that refers to that open message queue description. The message queue descriptor is used by other functions to refer to that message queue. The *name* argument points to a string naming a message queue. It is unspecified whether the name appears in the file system and is visible to other functions that take pathnames as arguments. The *name* argument conforms to the construction rules for a pathname, except that the interpretation of <slash> characters other than the leading <slash> character in *name* is implementation-defined, and that the length limits for the *name* argument are implementation-defined and need not be the same as the pathname limits {PATH_MAX} and {NAME_MAX}. If *name* begins with the <slash> character, then processes calling *mq_open()* with the same value of *name* shall refer to the same message queue object, as long as that name has not been removed. If *name* does not begin with the <slash> character, the effect is implementation-defined. If the *name* argument is not the name of an existing message queue and creation is not requested, *mq_open()* shall fail and return an error.

A message queue descriptor may be implemented using a file descriptor, in which case applications can open up to at least {OPEN_MAX} file and message queues.

The *oflag* argument requests the desired receive and/or send access to the message queue. The requested access permission to receive messages or send messages shall be granted if the calling process would be granted read or write access, respectively, to an equivalently protected file.

The value of *oflag* is the bitwise-inclusive OR of values from the following list. Applications shall specify exactly one of the first three values (access modes) below in the value of *oflag*:

O_RDONLY Open the message queue for receiving messages. The process can use the returned message queue descriptor with *mq_receive()*, but not *mq_send()*. A message queue may be open multiple times in the same or different processes for receiving messages.

O_WRONLY Open the queue for sending messages. The process can use the returned message queue descriptor with *mq_send()* but not *mq_receive()*. A message queue may be open multiple times in the same or different processes for sending messages.

O_RDWR Open the queue for both receiving and sending messages. The process can use any of the functions allowed for **O_RDONLY** and **O_WRONLY**. A message queue may be open multiple times in the same or different processes for sending messages.

Any combination of the remaining flags may be specified in the value of *oflag*:

O_CREAT Create a message queue. It requires two additional arguments: *mode*, which shall be of type **mode_t**, and *attr*, which shall be a pointer to an **mq_attr** structure. If the pathname *name* has already been used to create a message queue that still exists, then this flag shall have no effect, except as noted under **O_EXCL**. Otherwise, a message queue shall be created without any messages

44404		in it. The user ID of the message queue shall be set to the effective user ID of
44405		the process. The group ID of the message queue shall be set to the effective
44406		group ID of the process; however, if the <i>name</i> argument is visible in the file
44407		system, the group ID may be set to the group ID of the containing directory.
44408		When bits in <i>mode</i> other than the file permission bits are specified, the effect is
44409		unspecified. If <i>attr</i> is NULL, the message queue shall be created with
44410		implementation-defined default message queue attributes. If <i>attr</i> is non-NULL
44411		and the calling process has appropriate privileges on <i>name</i> , the message queue
44412		<i>mq_maxmsg</i> and <i>mq_msgsize</i> attributes shall be set to the values of the
44413		corresponding members in the mq_attr structure referred to by <i>attr</i> . The
44414		values of the <i>mq_flags</i> and <i>mq_curmsgs</i> members of the mq_attr structure shall
44415		be ignored. If <i>attr</i> is non-NULL, but the calling process does not have
44416		appropriate privileges on <i>name</i> , the <i>mq_open()</i> function shall fail and return an
44417		error without creating the message queue.
44418	O_EXCL	If O_EXCL and O_CREAT are set, <i>mq_open()</i> shall fail if the message queue
44419		<i>name</i> exists. The check for the existence of the message queue and the creation
44420		of the message queue if it does not exist shall be atomic with respect to other
44421		threads executing <i>mq_open()</i> naming the same <i>name</i> with O_EXCL and
44422		O_CREAT set. If O_EXCL is set and O_CREAT is not set, the result is
44423		undefined.
44424	O_NONBLOCK	Determines whether an <i>mq_send()</i> or <i>mq_receive()</i> waits for resources or
44425		messages that are not currently available, or fails with <i>errno</i> set to [EAGAIN];
44426		see <i>mq_send()</i> and <i>mq_receive()</i> for details.
44427	The <i>mq_open()</i> function does not add or remove messages from the queue.	
44428	RETURN VALUE	
44429	Upon successful completion, the function shall return a message queue descriptor; otherwise,	
44430	the function shall return (mqd_t)−1 and set <i>errno</i> to indicate the error.	
44431	ERRORS	
44432	The <i>mq_open()</i> function shall fail if:	
44433	[EACCES]	The message queue exists and the permissions specified by <i>oflag</i> are denied, or
44434		the message queue does not exist and permission to create the message queue
44435		is denied.
44436	[EEXIST]	O_CREAT and O_EXCL are set and the named message queue already exists.
44437	[EINTR]	The <i>mq_open()</i> function was interrupted by a signal.
44438	[EINVAL]	The <i>mq_open()</i> function is not supported for the given name.
44439	[EINVAL]	O_CREAT was specified in <i>oflag</i> , the value of <i>attr</i> is not NULL, and either
44440		<i>mq_maxmsg</i> or <i>mq_msgsize</i> was less than or equal to zero.
44441	[EMFILE]	Too many message queue descriptors or file descriptors are currently in use by
44442		this process.
44443	[ENFILE]	Too many message queues are currently open in the system.
44444	[ENOENT]	O_CREAT is not set and the named message queue does not exist.
44445	[ENOSPC]	There is insufficient space for the creation of the new message queue.

44446 If any of the following conditions occur, the *mq_open()* function may return (**mqd_t**)–1 and set
 44447 *errno* to the corresponding value.

44448 [ENAMETOOLONG]

44449 The length of the *name* argument exceeds {_POSIX_PATH_MAX} on systems
 44450 XSI that do not support the XSI option or exceeds {_XOPEN_PATH_MAX} on XSI
 44451 systems, or has a pathname component that is longer than
 44452 XSI {_POSIX_NAME_MAX} on systems that do not support the XSI option or
 44453 longer than {_XOPEN_NAME_MAX} on XSI systems.

44454 EXAMPLES

44455 None.

44456 APPLICATION USAGE

44457 None.

44458 RATIONALE

44459 None.

44460 FUTURE DIRECTIONS

44461 A future version might require the *mq_open()* and *mq_unlink()* functions to have semantics
 44462 similar to normal file system operations.

44463 SEE ALSO

44464 *mq_close()*, *mq_getattr()*, *mq_receive()*, *mq_send()*, *mq_setattr()*, *mq_unlink()*, *msgctl()*, *msgget()*,
 44465 *msgrcv()*, *msgsnd()*

44466 XBD <mqqueue.h>

44467 CHANGE HISTORY

44468 First released in Issue 5. Included for alignment with the POSIX Realtime Extension.

44469 Issue 6

44470 The *mq_open()* function is marked as part of the Message Passing option.

44471 The [ENOSYS] error condition has been removed as stubs need not be provided if an
 44472 implementation does not support the Message Passing option.

44473 The *mq_timedreceive()* and *mq_timedsend()* functions are added to the SEE ALSO section for
 44474 alignment with IEEE Std 1003.1d-1999.

44475 The DESCRIPTION of O_EXCL is updated in response to IEEE PASC Interpretation 1003.1c #48.

44476 IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/62 is applied, updating the description of
 44477 the permission bits in the DESCRIPTION. The change is made for consistency with the
 44478 *shm_open()* and *sem_open()* functions.

44479 Issue 7

44480 Austin Group Interpretation 1003.1-2001 #077 is applied, clarifying the *name* argument and
 44481 changing [ENAMETOOLONG] from a “shall fail” to a “may fail” error.

44482 Austin Group Interpretation 1003.1-2001 #141 is applied, adding FUTURE DIRECTIONS.

44483 SD5-XSH-ERN-170 is applied, updating the DESCRIPTION to clarify the wording for setting the
 44484 user ID and group ID of the message queue.

44485 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0394 [259] is applied.

+

NAME

mq_receive, mq_timedreceive — receive a message from a message queue (**REALTIME**)

SYNOPSIS

```
MSG    #include <mqueue.h>

ssize_t mq_receive(mqd_t mqdes, char *msg_ptr, size_t msg_len,
    unsigned *msg_prio);

#include <mqueue.h>
#include <time.h>

ssize_t mq_timedreceive(mqd_t mqdes, char *restrict msg_ptr,
    size_t msg_len, unsigned *restrict msg_prio,
    const struct timespec *restrict abstime);
```

DESCRIPTION

The *mq_receive()* function shall receive the oldest of the highest priority message(s) from the message queue specified by *mqdes*. If the size of the buffer in bytes, specified by the *msg_len* argument, is less than the *mq_msgsize* attribute of the message queue, the function shall fail and return an error. Otherwise, the selected message shall be removed from the queue and copied to the buffer pointed to by the *msg_ptr* argument.

If the value of *msg_len* is greater than {SSIZE_MAX}, the result is implementation-defined.

If the argument *msg_prio* is not NULL, the priority of the selected message shall be stored in the location referenced by *msg_prio*.

If the specified message queue is empty and O_NONBLOCK is not set in the message queue description associated with *mqdes*, *mq_receive()* shall block until a message is enqueued on the message queue or until *mq_receive()* is interrupted by a signal. If more than one thread is waiting to receive a message when a message arrives at an empty queue and the Priority Scheduling option is supported, then the thread of highest priority that has been waiting the longest shall be selected to receive the message. Otherwise, it is unspecified which waiting thread receives the message. If the specified message queue is empty and O_NONBLOCK is set in the message queue description associated with *mqdes*, no message shall be removed from the queue, and *mq_receive()* shall return an error.

The *mq_timedreceive()* function shall receive the oldest of the highest priority messages from the message queue specified by *mqdes* as described for the *mq_receive()* function. However, if O_NONBLOCK was not specified when the message queue was opened via the *mq_open()* function, and no message exists on the queue to satisfy the receive, the wait for such a message shall be terminated when the specified timeout expires. If O_NONBLOCK is set, this function is equivalent to *mq_receive()*.

The timeout expires when the absolute time specified by *abstime* passes, as measured by the clock on which timeouts are based (that is, when the value of that clock equals or exceeds *abstime*), or if the absolute time specified by *abstime* has already been passed at the time of the call.

The timeout shall be based on the CLOCK_REALTIME clock. The resolution of the timeout shall be the resolution of the clock on which it is based. The *timespec* argument is defined in the **<time.h>** header.

Under no circumstance shall the operation fail with a timeout if a message can be removed from the message queue immediately. The validity of the *abstime* parameter need not be checked if a message can be removed from the message queue immediately.

RETURN VALUE

Upon successful completion, the *mq_receive()* and *mq_timedreceive()* functions shall return the length of the selected message in bytes and the message shall be removed from the queue. Otherwise, no message shall be removed from the queue, the functions shall return a value of *-1*, and set *errno* to indicate the error.

ERRORS

These functions shall fail if:

- [EAGAIN] O_NONBLOCK was set in the message description associated with *mqdes*, and the specified message queue is empty.
- [EBADF] The *mqdes* argument is not a valid message queue descriptor open for reading.
- [EMSGSIZE] The specified message buffer size, *msg_len*, is less than the message size attribute of the message queue.
- [EINTR] The *mq_receive()* or *mq_timedreceive()* operation was interrupted by a signal.
- [EINVAL] The process or thread would have blocked, and the *abstime* parameter specified a nanoseconds field value less than zero or greater than or equal to 1 000 million.
- [ETIMEDOUT] The O_NONBLOCK flag was not set when the message queue was opened, but no message arrived on the queue before the specified timeout expired.

These functions may fail if:

- [EBADMSG] The implementation has detected a data corruption problem with the message.

EXAMPLES

None.

APPLICATION USAGE

None.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

mq_open(), *mq_send()*, *msgctl()*, *msgget()*, *msgrcv()*, *msgsnd()*, *time()*

XBD <mqqueue.h>, <time.h>

CHANGE HISTORY

First released in Issue 5. Included for alignment with the POSIX Realtime Extension.

Issue 6

The *mq_receive()* function is marked as part of the Message Passing option.

The Open Group Corrigendum U021/4 is applied. The DESCRIPTION is changed to refer to *msg_len* rather than *maxsize*.

The [ENOSYS] error condition has been removed as stubs need not be provided if an implementation does not support the Message Passing option.

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- 44573 • In this function it is possible for the return value to exceed the range of the type **ssize_t**
44574 (since **size_t** has a larger range of positive values than **ssize_t**). A sentence restricting the
44575 size of the **size_t** object is added to the description to resolve this conflict.

44576 The *mq_timedreceive()* function is added for alignment with IEEE Std 1003.1d-1999.

44577 The **restrict** keyword is added to the *mq_timedreceive()* prototype for alignment with the
44578 ISO/IEC 9899:1999 standard.

44579 IEEE PASC Interpretation 1003.1 #109 is applied, correcting the return type for *mq_timedreceive()*
44580 from **int** to **ssize_t**.

44581 **Issue 7**

44582 The *mq_timedreceive()* function is moved from the Timeouts option to the Base.

44583 Functionality relating to the Timers option is moved to the Base.

NAME

mq_send, mq_timedsend — send a message to a message queue (**REALTIME**)

SYNOPSIS

```
MSG    #include <mqueue.h>

int mq_send(mqd_t mqdes, const char *msg_ptr, size_t msg_len,
            unsigned msg_prio);

#include <mqueue.h>
#include <time.h>

int mq_timedsend(mqd_t mqdes, const char *msg_ptr, size_t msg_len,
                unsigned msg_prio, const struct timespec *abstime);
```

DESCRIPTION

The *mq_send()* function shall add the message pointed to by the argument *msg_ptr* to the message queue specified by *mqdes*. The *msg_len* argument specifies the length of the message, in bytes, pointed to by *msg_ptr*. The value of *msg_len* shall be less than or equal to the *mq_msgsize* attribute of the message queue, or *mq_send()* shall fail.

If the specified message queue is not full, *mq_send()* shall behave as if the message is inserted into the message queue at the position indicated by the *msg_prio* argument. A message with a larger numeric value of *msg_prio* shall be inserted before messages with lower values of *msg_prio*. A message shall be inserted after other messages in the queue, if any, with equal *msg_prio*. The value of *msg_prio* shall be less than {MQ_PRIO_MAX}.

If the specified message queue is full and O_NONBLOCK is not set in the message queue description associated with *mqdes*, *mq_send()* shall block until space becomes available to enqueue the message, or until *mq_send()* is interrupted by a signal. If more than one thread is waiting to send when space becomes available in the message queue and the Priority Scheduling option is supported, then the thread of the highest priority that has been waiting the longest shall be unblocked to send its message. Otherwise, it is unspecified which waiting thread is unblocked. If the specified message queue is full and O_NONBLOCK is set in the message queue description associated with *mqdes*, the message shall not be queued and *mq_send()* shall return an error.

The *mq_timedsend()* function shall add a message to the message queue specified by *mqdes* in the manner defined for the *mq_send()* function. However, if the specified message queue is full and O_NONBLOCK is not set in the message queue description associated with *mqdes*, the wait for sufficient room in the queue shall be terminated when the specified timeout expires. If O_NONBLOCK is set in the message queue description, this function shall be equivalent to *mq_send()*.

The timeout shall expire when the absolute time specified by *abstime* passes, as measured by the clock on which timeouts are based (that is, when the value of that clock equals or exceeds *abstime*), or if the absolute time specified by *abstime* has already been passed at the time of the call.

The timeout shall be based on the CLOCK_REALTIME clock. The resolution of the timeout shall be the resolution of the clock on which it is based. The *timespec* argument is defined in the **<time.h>** header.

Under no circumstance shall the operation fail with a timeout if there is sufficient room in the queue to add the message immediately. The validity of the *abstime* parameter need not be checked when there is sufficient room in the queue.

RETURN VALUE

Upon successful completion, the `mq_send()` and `mq_timedsend()` functions shall return a value of zero. Otherwise, no message shall be enqueued, the functions shall return `-1`, and `errno` shall be set to indicate the error.

ERRORS

The `mq_send()` and `mq_timedsend()` functions shall fail if:

- [EAGAIN] The `O_NONBLOCK` flag is set in the message queue description associated with `mqdes`, and the specified message queue is full.
- [EBADF] The `mqdes` argument is not a valid message queue descriptor open for writing.
- [EINTR] A signal interrupted the call to `mq_send()` or `mq_timedsend()`.
- [EINVAL] The value of `msg_prio` was outside the valid range.
- [EINVAL] The process or thread would have blocked, and the `abstime` parameter specified a nanoseconds field value less than zero or greater than or equal to 1 000 million.
- [EMSGSIZE] The specified message length, `msg_len`, exceeds the message size attribute of the message queue.
- [ETIMEDOUT] The `O_NONBLOCK` flag was not set when the message queue was opened, but the timeout expired before the message could be added to the queue.

EXAMPLES

None.

APPLICATION USAGE

The value of the symbol `{MQ_PRIO_MAX}` limits the number of priority levels supported by the application. Message priorities range from 0 to `{MQ_PRIO_MAX}-1`.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

`mq_open()`, `mq_receive()`, `mq_setattr()`, `time()`

XBD `<mqqueue.h>`, `<time.h>`

CHANGE HISTORY

First released in Issue 5. Included for alignment with the POSIX Realtime Extension.

Issue 6

The `mq_send()` function is marked as part of the Message Passing option.

The `[ENOSYS]` error condition has been removed as stubs need not be provided if an implementation does not support the Message Passing option.

The `mq_timedsend()` function is added for alignment with IEEE Std 1003.1d-1999.

Issue 7

The `mq_timedsend()` function is moved from the Timeouts option to the Base.

Functionality relating to the Timers option is moved to the Base.

NAME

mq_setattr — set message queue attributes (**REALTIME**)

SYNOPSIS

```
#include <mqueue.h>

int mq_setattr(mqd_t mqdes, const struct mq_attr *restrict mqstat,
               struct mq_attr *restrict omqstat);
```

DESCRIPTION

The *mq_setattr()* function shall set attributes associated with the open message queue description referenced by the message queue descriptor specified by *mqdes*.

The message queue attributes corresponding to the following members defined in the **mq_attr** structure shall be set to the specified values upon successful completion of *mq_setattr()*:

mq_flags The value of this member is the bitwise-logical OR of zero or more of **O_NONBLOCK** and any implementation-defined flags.

The values of the *mq_maxmsg*, *mq_msgsize*, and *mq_curmsgs* members of the **mq_attr** structure shall be ignored by *mq_setattr()*.

If *omqstat* is non-NULL, the *mq_setattr()* function shall store, in the location referenced by *omqstat*, the previous message queue attributes and the current queue status. These values shall be the same as would be returned by a call to *mq_getattr()* at that point.

RETURN VALUE

Upon successful completion, the function shall return a value of zero and the attributes of the message queue shall have been changed as specified.

Otherwise, the message queue attributes shall be unchanged, and the function shall return a value of -1 and set *errno* to indicate the error.

ERRORS

The *mq_setattr()* function shall fail if:

[EBADF] The *mqdes* argument is not a valid message queue descriptor.

EXAMPLES

None.

APPLICATION USAGE

None.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

mq_open(), *mq_send()*, *msgctl()*, *msgget()*, *msgrcv()*, *msgsnd()*

XBD **<mqueue.h>**

CHANGE HISTORY

First released in Issue 5. Included for alignment with the POSIX Realtime Extension.

44708 **Issue 6**

44709 The *mq_setattr()* function is marked as part of the Message Passing option.

44710 The [ENOSYS] error condition has been removed as stubs need not be provided if an
44711 implementation does not support the Message Passing option.

44712 The *mq_timedsend()* function is added to the SEE ALSO section for alignment with IEEE Std
44713 1003.1d-1999.

44714 The **restrict** keyword is added to the *mq_setattr()* prototype for alignment with the
44715 ISO/IEC 9899:1999 standard.

44716 **NAME**44717 mq_timedreceive — receive a message from a message queue (**ADVANCED REALTIME**)44718 **SYNOPSIS**

```
44719 MSG      #include <mqueue.h>
44720           #include <time.h>
44721           ssize_t mq_timedreceive(mqd_t mqdes, char *restrict msg_ptr,
44722                                   size_t msg_len, unsigned *restrict msg_prio,
44723                                   const struct timespec *restrict abstime);
```

44724 **DESCRIPTION**44725 Refer to *mq_receive()*.

44726 **NAME**44727 mq_timedsend — send a message to a message queue (**ADVANCED REALTIME**)44728 **SYNOPSIS**

```
44729 MSG      #include <mqueue.h>
44730          #include <time.h>
44731          int mq_timedsend(mqd_t mqdes, const char *msg_ptr, size_t msg_len,
44732                          unsigned msg_prio, const struct timespec *abstime);
```

44733 **DESCRIPTION**44734 Refer to *mq_send()*.

44735 **NAME**44736 mq_unlink — remove a message queue (**REALTIME**)44737 **SYNOPSIS**

```
44738 MSG      #include <mqueue.h>
44739          int mq_unlink(const char *name);
```

44740 **DESCRIPTION**

44741 The *mq_unlink()* function shall remove the message queue named by the string name. If one or
 44742 more processes have the message queue open when *mq_unlink()* is called, destruction of the
 44743 message queue shall be postponed until all references to the message queue have been closed.
 44744 However, the *mq_unlink()* call need not block until all references have been closed; it may return
 44745 immediately.

44746 After a successful call to *mq_unlink()*, reuse of the name shall subsequently cause *mq_open()* to
 44747 behave as if no message queue of this name exists (that is, *mq_open()* will fail if O_CREAT is not
 44748 set, or will create a new message queue if O_CREAT is set).

44749 **RETURN VALUE**

44750 Upon successful completion, the function shall return a value of zero. Otherwise, the named
 44751 message queue shall be unchanged by this function call, and the function shall return a value of
 44752 -1 and set *errno* to indicate the error.

44753 **ERRORS**44754 The *mq_unlink()* function shall fail if:

44755 [EACCES] Permission is denied to unlink the named message queue.

44756 [ENOENT] The named message queue does not exist.

44757 The *mq_unlink()* function may fail if:

44758 [ENAMETOOLONG]

44759 The length of the *name* argument exceeds {_POSIX_PATH_MAX} on systems
 44760 XSI that do not support the XSI option or exceeds {_XOPEN_PATH_MAX} on XSI
 44761 systems, or has a pathname component that is longer than
 44762 XSI {_POSIX_NAME_MAX} on systems that do not support the XSI option or
 44763 longer than {_XOPEN_NAME_MAX} on XSI systems. A call to *mq_unlink()*
 44764 with a *name* argument that contains the same message queue name as was
 44765 previously used in a successful *mq_open()* call shall not give an
 44766 [ENAMETOOLONG] error.

44767 **EXAMPLES**

44768 None.

44769 **APPLICATION USAGE**

44770 None.

44771 **RATIONALE**

44772 None.

44773 **FUTURE DIRECTIONS**

44774 A future version might require the *mq_open()* and *mq_unlink()* functions to have semantics
 44775 similar to normal file system operations.

44776 **SEE ALSO**44777 *mq_close()*, *mq_open()*, *msgctl()*, *msgget()*, *msgrcv()*, *msgsnd()*

44778 XBD <mqqueue.h>

44779 **CHANGE HISTORY**

44780 First released in Issue 5. Included for alignment with the POSIX Realtime Extension.

44781 **Issue 6**44782 The *mq_unlink()* function is marked as part of the Message Passing option.44783 The Open Group Corrigendum U021/5 is applied, clarifying that upon unsuccessful completion,
44784 the named message queue is unchanged by this function.44785 The [ENOSYS] error condition has been removed as stubs need not be provided if an
44786 implementation does not support the Message Passing option.44787 **Issue 7**44788 Austin Group Interpretation 1003.1-2001 #077 is applied, changing [ENAMETOOLONG] from a
44789 “shall fail” to a “may fail” error .

44790 Austin Group Interpretation 1003.1-2001 #141 is applied.

44791 NAME

44792 mrnd48 — generate uniformly distributed pseudo-random signed long integers

44793 SYNOPSIS

```
44794 XSI #include <stdlib.h>  
44795 long mrnd48(void);
```

44796 DESCRIPTION

44797 Refer to *drand48()*.

44798 **NAME**44799 **msgctl** — XSI message control operations44800 **SYNOPSIS**

```
44801 XSI      #include <sys/msg.h>
44802      int msgctl(int msqid, int cmd, struct msqid_ds *buf);
```

44803 **DESCRIPTION**

44804 The *msgctl()* function operates on XSI message queues (see XBD [Section 3.225](#), on page 69). It is
 44805 unspecified whether this function interoperates with the realtime interprocess communication
 44806 facilities defined in [Section 2.8](#) (on page 502).

44807 The *msgctl()* function shall provide message control operations as specified by *cmd*. The
 44808 following values for *cmd*, and the message control operations they specify, are:

44809 **IPC_STAT** Place the current value of each member of the **msqid_ds** data structure
 44810 associated with *msqid* into the structure pointed to by *buf*. The contents of this
 44811 structure are defined in **<sys/msg.h>**.

44812 **IPC_SET** Set the value of the following members of the **msqid_ds** data structure
 44813 associated with *msqid* to the corresponding value found in the structure
 44814 pointed to by *buf*:

```
44815      msg_perm.uid
44816      msg_perm.gid
44817      msg_perm.mode
44818      msg_qbytes
```

44819 Also, the *msg_ctime* timestamp shall be set to the current time, as described in [Section 2.7.1](#) (on page 501). +
 44820 +

44821 IPC_SET can only be executed by a process with appropriate privileges or that
 44822 has an effective user ID equal to the value of **msg_perm.cuid** or
 44823 **msg_perm.uid** in the **msqid_ds** data structure associated with *msqid*. Only a
 44824 process with appropriate privileges can raise the value of **msg_qbytes**.

44825 **IPC_RMID** Remove the message queue identifier specified by *msqid* from the system and
 44826 destroy the message queue and **msqid_ds** data structure associated with it.
 44827 IPC_RMID can only be executed by a process with appropriate privileges or
 44828 one that has an effective user ID equal to the value of **msg_perm.cuid** or
 44829 **msg_perm.uid** in the **msqid_ds** data structure associated with *msqid*.

44830 **RETURN VALUE**

44831 Upon successful completion, *msgctl()* shall return 0; otherwise, it shall return -1 and set *errno* to
 44832 indicate the error.

44833 **ERRORS**

44834 The *msgctl()* function shall fail if:

44835 **[EACCES]** The argument *cmd* is IPC_STAT and the calling process does not have read
 44836 permission; see [Section 2.7](#) (on page 501).

44837 **[EINVAL]** The value of *msqid* is not a valid message queue identifier; or the value of *cmd*
 44838 is not a valid command.

44839 **[EPERM]** The argument *cmd* is IPC_RMID or IPC_SET and the effective user ID of the
 44840 calling process is not equal to that of a process with appropriate privileges and
 44841 it is not equal to the value of **msg_perm.cuid** or **msg_perm.uid** in the data

44842 structure associated with *mqd*.

44843 [EPERM] The argument *cmd* is IPC_SET, an attempt is being made to increase to the

44844 value of **msg_qbytes**, and the effective user ID of the calling process does not

44845 have appropriate privileges.

44846 EXAMPLES

44847 None.

44848 APPLICATION USAGE

44849 The POSIX Realtime Extension defines alternative interfaces for interprocess communication

44850 (IPC). Application developers who need to use IPC should design their applications so that

44851 modules using the IPC routines described in [Section 2.7](#) (on page 501) can be easily modified to

44852 use the alternative interfaces.

44853 RATIONALE

44854 None.

44855 FUTURE DIRECTIONS

44856 None.

44857 SEE ALSO

44858 [Section 2.7](#) (on page 501), [Section 2.8](#) (on page 502), [mq_close\(\)](#), [mq_getattr\(\)](#), [mq_notify\(\)](#),

44859 [mq_open\(\)](#), [mq_receive\(\)](#), [mq_send\(\)](#), [mq_setattr\(\)](#), [mq_unlink\(\)](#), [msgget\(\)](#), [msgrcv\(\)](#), [msgsnd\(\)](#)

44860 XBD [Section 3.225](#) (on page 69), [<sys/msg.h>](#)

44861 CHANGE HISTORY

44862 First released in Issue 2. Derived from Issue 2 of the SVID.

44863 Issue 5

44864 The note about use of POSIX Realtime Extension IPC routines has been moved from FUTURE

44865 DIRECTIONS to a new APPLICATION USAGE section.

44866 Issue 7

44867 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0395 [345] is applied.

+

NAME

msgget — get the XSI message queue identifier

SYNOPSIS

```
XSI    #include <sys/msg.h>
      int msgget(key_t key, int msgflg);
```

DESCRIPTION

The `msgget()` function operates on XSI message queues (see XBD [Section 3.225](#), on page 69). It is unspecified whether this function interoperates with the realtime interprocess communication facilities defined in [Section 2.8](#) (on page 502).

The `msgget()` function shall return the message queue identifier associated with the argument `key`.

A message queue identifier, associated message queue, and data structure (see `<sys/msg.h>`), shall be created for the argument `key` if one of the following is true:

- The argument `key` is equal to `IPC_PRIVATE`.
- The argument `key` does not already have a message queue identifier associated with it, and (`msgflg & IPC_CREAT`) is non-zero.

Upon creation, the data structure associated with the new message queue identifier shall be initialized as follows:

- `msg_perm.cuid`, `msg_perm.uid`, `msg_perm.cgid`, and `msg_perm.gid` shall be set to the effective user ID and effective group ID, respectively, of the calling process.
- The low-order 9 bits of `msg_perm.mode` shall be set to the low-order 9 bits of `msgflg`.
- `msg_qnum`, `msg_lspid`, `msg_lrpid`, `msg_stime`, and `msg_rtime` shall be set to 0.
- `msg_ctime` shall be set to the current time, as described in [Section 2.7.1](#) (on page 501).
- `msg_qbytes` shall be set to the system limit.

RETURN VALUE

Upon successful completion, `msgget()` shall return a non-negative integer, namely a message queue identifier. Otherwise, it shall return `-1` and set `errno` to indicate the error.

ERRORS

The `msgget()` function shall fail if:

[EACCES]	A message queue identifier exists for the argument <code>key</code> , but operation permission as specified by the low-order 9 bits of <code>msgflg</code> would not be granted; see Section 2.7 (on page 501).
[EEXIST]	A message queue identifier exists for the argument <code>key</code> but (<code>(msgflg & IPC_CREAT) && (msgflg & IPC_EXCL)</code>) is non-zero.
[ENOENT]	A message queue identifier does not exist for the argument <code>key</code> and (<code>msgflg & IPC_CREAT</code>) is 0.
[ENOSPC]	A message queue identifier is to be created but the system-imposed limit on the maximum number of allowed message queue identifiers system-wide would be exceeded.

44907 **EXAMPLES**

44908 None.

44909 **APPLICATION USAGE**

44910 The POSIX Realtime Extension defines alternative interfaces for interprocess communication
44911 (IPC). Application developers who need to use IPC should design their applications so that
44912 modules using the IPC routines described in [Section 2.7](#) (on page 501) can be easily modified to
44913 use the alternative interfaces.

44914 **RATIONALE**

44915 None.

44916 **FUTURE DIRECTIONS**

44917 None.

44918 **SEE ALSO**

44919 [Section 2.7](#) (on page 501), [Section 2.8](#) (on page 502), [ftok\(\)](#), [mq_close\(\)](#), [mq_getattr\(\)](#), [mq_notify\(\)](#),
44920 [mq_open\(\)](#), [mq_receive\(\)](#), [mq_send\(\)](#), [mq_setattr\(\)](#), [mq_unlink\(\)](#), [msgctl\(\)](#), [msgrcv\(\)](#), [msgsnd\(\)](#)

44921 XBD [Section 3.225](#) (on page 69), [<sys/msg.h>](#)44922 **CHANGE HISTORY**

44923 First released in Issue 2. Derived from Issue 2 of the SVID.

44924 **Issue 5**

44925 The note about use of POSIX Realtime Extension IPC routines has been moved from FUTURE
44926 DIRECTIONS to a new APPLICATION USAGE section.

44927 **Issue 7**

44928 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0396 [345] and XSH/TC1-2008/0397 +
44929 [344] are applied.

NAME

msgrcv — XSI message receive operation

SYNOPSIS

```
XSI    #include <sys/msg.h>

      ssize_t msgrcv(int msqid, void *msgp, size_t msgsz, long msgtyp,
                    int msgflg);
```

DESCRIPTION

The *msgrcv()* function operates on XSI message queues (see XBD [Section 3.225](#), on page 69). It is unspecified whether this function interoperates with the realtime interprocess communication facilities defined in [Section 2.8](#) (on page 502).

The *msgrcv()* function shall read a message from the queue associated with the message queue identifier specified by *msqid* and place it in the user-defined buffer pointed to by *msgp*.

The application shall ensure that the argument *msgp* points to a user-defined buffer that contains first a field of type **long** specifying the type of the message, and then a data portion that holds the data bytes of the message. The structure below is an example of what this user-defined buffer might look like:

```
struct mymsg {
    long    mtype;      /* Message type. */
    char    mtext[1];   /* Message text. */
}
```

The structure member *mtype* is the received message's type as specified by the sending process.

The structure member *mtext* is the text of the message.

The argument *msgsz* specifies the size in bytes of *mtext*. The received message shall be truncated to *msgsz* bytes if it is larger than *msgsz* and (*msgflg* & MSG_NOERROR) is non-zero. The truncated part of the message shall be lost and no indication of the truncation shall be given to the calling process.

If the value of *msgsz* is greater than {SSIZE_MAX}, the result is implementation-defined.

The argument *msgtyp* specifies the type of message requested as follows:

- If *msgtyp* is 0, the first message on the queue shall be received.
- If *msgtyp* is greater than 0, the first message of type *msgtyp* shall be received.
- If *msgtyp* is less than 0, the first message of the lowest type that is less than or equal to the absolute value of *msgtyp* shall be received.

The argument *msgflg* specifies the action to be taken if a message of the desired type is not on the queue. These are as follows:

- If (*msgflg* & IPC_NOWAIT) is non-zero, the calling thread shall return immediately with a return value of -1 and *errno* set to [ENOMSG].
- If (*msgflg* & IPC_NOWAIT) is 0, the calling thread shall suspend execution until one of the following occurs:
 - A message of the desired type is placed on the queue.
 - The message queue identifier *msqid* is removed from the system; when this occurs, *errno* shall be set to [EIDRM] and -1 shall be returned.

44971 — The calling thread receives a signal that is to be caught; in this case a message is not
 44972 received and the calling thread resumes execution in the manner prescribed in
 44973 *sigaction()*.

44974 Upon successful completion, the following actions are taken with respect to the data structure
 44975 associated with *msqid*:

- 44976 • **msg_qnum** shall be decremented by 1.
- 44977 • **msg_lrpid** shall be set to the process ID of the calling process. |
- 44978 • **msg_rtime** shall be set to the current time, as described in [Section 2.7.1](#) (on page 501). |

44979 RETURN VALUE

44980 Upon successful completion, *msgrcv()* shall return a value equal to the number of bytes actually
 44981 placed into the buffer *mtext*. Otherwise, no message shall be received, *msgrcv()* shall return `-1`,
 44982 and *errno* shall be set to indicate the error. |

44983 ERRORS

44984 The *msgrcv()* function shall fail if:

44985	[E2BIG]	The value of <i>mtext</i> is greater than <i>msgsz</i> and (<i>msgflg</i> & MSG_NOERROR) is 0.
44986	[EACCES]	Operation permission is denied to the calling process; see Section 2.7 (on page 501).
44987		
44988	[EIDRM]	The message queue identifier <i>msqid</i> is removed from the system.
44989	[EINTR]	The <i>msgrcv()</i> function was interrupted by a signal.
44990	[EINVAL]	<i>msqid</i> is not a valid message queue identifier.
44991	[ENOMSG]	The queue does not contain a message of the desired type and (<i>msgflg</i> & IPC_NOWAIT) is non-zero.
44992		

44993 EXAMPLES

44994 Receiving a Message

44995 The following example receives the first message on the queue (based on the value of the *msgtyp*
 44996 argument, 0). The queue is identified by the *msqid* argument (assuming that the value has
 44997 previously been set). This call specifies that an error should be reported if no message is
 44998 available, but not if the message is too large. The message size is calculated directly using the
 44999 *sizeof* operator.

```

45000 #include <sys/msg.h>
45001 ...
45002 int result;
45003 int msqid;
45004 struct message {
45005     long type;
45006     char text[20];
45007 } msg;
45008 long msgtyp = 0;
45009 ...
45010 result = msgrcv(msqid, (void *) &msg, sizeof(msg.text),
45011                 msgtyp, MSG_NOERROR | IPC_NOWAIT);
  
```

45012 APPLICATION USAGE

45013 The POSIX Realtime Extension defines alternative interfaces for interprocess communication
45014 (IPC). Application developers who need to use IPC should design their applications so that
45015 modules using the IPC routines described in [Section 2.7](#) (on page 501) can be easily modified to
45016 use the alternative interfaces.

45017 RATIONALE

45018 None.

45019 FUTURE DIRECTIONS

45020 None.

45021 SEE ALSO

45022 [Section 2.7](#) (on page 501), [Section 2.8](#) (on page 502), [mq_close\(\)](#), [mq_getattr\(\)](#), [mq_notify\(\)](#),
45023 [mq_open\(\)](#), [mq_receive\(\)](#), [mq_send\(\)](#), [mq_setattr\(\)](#), [mq_unlink\(\)](#), [msgctl\(\)](#), [msgget\(\)](#), [msgsnd\(\)](#),
45024 [sigaction\(\)](#)

45025 XBD [Section 3.225](#) (on page 69), [<sys/msg.h>](#)

45026 CHANGE HISTORY

45027 First released in Issue 2. Derived from Issue 2 of the SVID.

45028 Issue 5

45029 The type of the return value is changed from **int** to **ssize_t**, and a warning is added to the
45030 DESCRIPTION about values of *msgsz* larger the {SSIZE_MAX}.

45031 The note about use of POSIX Realtime Extension IPC routines has been moved from FUTURE
45032 DIRECTIONS to the APPLICATION USAGE section.

45033 Issue 6

45034 The normative text is updated to avoid use of the term “must” for application requirements.

45035 Issue 7

45036 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0398 [345] and XSH/TC1-2008/0399 +
45037 [421] are applied.

NAME

msgsnd — XSI message send operation

SYNOPSIS

```
XSI    #include <sys/msg.h>

int msgsnd(int msqid, const void *msgp, size_t msgsz, int msgflg);
```

DESCRIPTION

The *msgsnd()* function operates on XSI message queues (see XBD [Section 3.225](#), on page 69). It is unspecified whether this function interoperates with the realtime interprocess communication facilities defined in [Section 2.8](#) (on page 502).

The *msgsnd()* function shall send a message to the queue associated with the message queue identifier specified by *msqid*.

The application shall ensure that the argument *msgp* points to a user-defined buffer that contains first a field of type **long** specifying the type of the message, and then a data portion that holds the data bytes of the message. The structure below is an example of what this user-defined buffer might look like:

```
struct mymsg {
    long    mtype;        /* Message type. */
    char    mtext[1];     /* Message text. */
}
```

The structure member *mtype* is a non-zero positive type **long** that can be used by the receiving process for message selection.

The structure member *mtext* is any text of length *msgsz* bytes. The argument *msgsz* can range from 0 to a system-imposed maximum.

The argument *msgflg* specifies the action to be taken if one or more of the following is true:

- The number of bytes already on the queue is equal to **msg_qbytes**; see **<sys/msg.h>**.
- The total number of messages on all queues system-wide is equal to the system-imposed limit.

These actions are as follows:

- If (*msgflg* & IPC_NOWAIT) is non-zero, the message shall not be sent and the calling thread shall return immediately.
- If (*msgflg* & IPC_NOWAIT) is 0, the calling thread shall suspend execution until one of the following occurs:
 - The condition responsible for the suspension no longer exists, in which case the message is sent.
 - The message queue identifier *msqid* is removed from the system; when this occurs, *errno* shall be set to [EIDRM] and **-1** shall be returned.
 - The calling thread receives a signal that is to be caught; in this case the message is not sent and the calling thread resumes execution in the manner prescribed in [sigaction\(\)](#).

Upon successful completion, the following actions are taken with respect to the data structure associated with *msqid*; see `<sys/msg.h>`:

- **msg_qnum** shall be incremented by 1.
- **msg_lspid** shall be set to the process ID of the calling process.
- **msg_stime** shall be set to the current time, as described in [Section 2.7.1](#) (on page 501).

RETURN VALUE

Upon successful completion, *msgsnd()* shall return 0; otherwise, no message shall be sent, *msgsnd()* shall return -1, and *errno* shall be set to indicate the error.

ERRORS

The *msgsnd()* function shall fail if:

[EACCES]	Operation permission is denied to the calling process; see Section 2.7 (on page 501).
[EAGAIN]	The message cannot be sent for one of the reasons cited above and (<i>msgflg</i> & <i>IPC_NOWAIT</i>) is non-zero.
[EIDRM]	The message queue identifier <i>msqid</i> is removed from the system.
[EINTR]	The <i>msgsnd()</i> function was interrupted by a signal.
[EINVAL]	The value of <i>msqid</i> is not a valid message queue identifier, or the value of <i>mtyp</i> is less than 1; or the value of <i>msgsz</i> is greater than the system-imposed limit.

EXAMPLES

Sending a Message

The following example sends a message to the queue identified by the *msqid* argument (assuming that value has previously been set). This call specifies that an error should be reported if no message is available. The message size is calculated directly using the *sizeof* operator.

```
#include <sys/msg.h>
...
int result;
int msqid;
struct message {
    long type;
    char text[20];
} msg;

msg.type = 1;
strcpy(msg.text, "This is message 1");
...
result = msgsnd(msqid, (void *) &msg, sizeof(msg.text), IPC_NOWAIT);
```

APPLICATION USAGE

The POSIX Realtime Extension defines alternative interfaces for interprocess communication (IPC). Application developers who need to use IPC should design their applications so that modules using the IPC routines described in [Section 2.7](#) (on page 501) can be easily modified to use the alternative interfaces.

45118 **RATIONALE**

45119 None.

45120 **FUTURE DIRECTIONS**

45121 None.

45122 **SEE ALSO**

45123 [Section 2.7](#) (on page 501), [Section 2.8](#) (on page 502), [mq_close\(\)](#), [mq_getattr\(\)](#), [mq_notify\(\)](#),
45124 [mq_open\(\)](#), [mq_receive\(\)](#), [mq_send\(\)](#), [mq_setattr\(\)](#), [mq_unlink\(\)](#), [msgctl\(\)](#), [msgget\(\)](#), [msgrcv\(\)](#),
45125 [sigaction\(\)](#)

45126 XBD [Section 3.225](#) (on page 69), [<sys/msg.h>](#)45127 **CHANGE HISTORY**

45128 First released in Issue 2. Derived from Issue 2 of the SVID.

45129 **Issue 5**

45130 The note about use of POSIX Realtime Extension IPC routines has been moved from FUTURE
45131 DIRECTIONS to a new APPLICATION USAGE section.

45132 **Issue 6**

45133 The normative text is updated to avoid use of the term “must” for application requirements.

45134 **Issue 7**

45135 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0400 [345] and XSH/TC1-2008/0401 +
45136 [359] are applied.

NAME

msync — synchronize memory with physical storage

SYNOPSIS

```
#include <sys/mman.h>

int msync(void *addr, size_t len, int flags);
```

DESCRIPTION

The *msync()* function shall write all modified data to permanent storage locations, if any, in those whole pages containing any part of the address space of the process starting at address *addr* and continuing for *len* bytes. If no such storage exists, *msync()* need not have any effect. If requested, the *msync()* function shall then invalidate cached copies of data.

The implementation may require that *addr* be a multiple of the page size as returned by *sysconf()*.

For mappings to files, the *msync()* function shall ensure that all write operations are completed as defined for synchronized I/O data integrity completion. It is unspecified whether the implementation also writes out other file attributes. When the *msync()* function is called on MAP_PRIVATE mappings, any modified data shall not be written to the underlying object and shall not cause such data to be made visible to other processes. It is unspecified whether data in MAP_PRIVATE mappings has any permanent storage locations. The effect of *msync()* on a shared memory object or a typed memory object is unspecified. The behavior of this function is unspecified if the mapping was not established by a call to *mmap()*.

The *flags* argument is constructed from the bitwise-inclusive OR of one or more of the following flags defined in the *<sys/mman.h>* header:

Symbolic Constant	Description
MS_ASYNC	Perform asynchronous writes.
MS_SYNC	Perform synchronous writes.
MS_INVALIDATE	Invalidate cached data.

When MS_ASYNC is specified, *msync()* shall return immediately once all the write operations are initiated or queued for servicing; when MS_SYNC is specified, *msync()* shall not return until all write operations are completed as defined for synchronized I/O data integrity completion. Either MS_ASYNC or MS_SYNC shall be specified, but not both.

When MS_INVALIDATE is specified, *msync()* shall invalidate all cached copies of mapped data that are inconsistent with the permanent storage locations such that subsequent references shall obtain data that was consistent with the permanent storage locations sometime between the call to *msync()* and the first subsequent memory reference to the data.

If *msync()* causes any write to a file, the file's last data modification and last file status change timestamps shall be marked for update.

RETURN VALUE

Upon successful completion, *msync()* shall return 0; otherwise, it shall return -1 and set *errno* to indicate the error.

ERRORS

The *msync()* function shall fail if:

[EBUSY] Some or all of the addresses in the range starting at *addr* and continuing for *len* bytes are locked, and MS_INVALIDATE is specified.

45180 [EINVAL] The value of *flags* is invalid.

45181 [ENOMEM] The addresses in the range starting at *addr* and continuing for *len* bytes are

45182 outside the range allowed for the address space of a process or specify one or

45183 more pages that are not mapped.

45184 The *msync()* function may fail if:

45185 [EINVAL] The value of *addr* is not a multiple of the page size as returned by *sysconf()*.

45186 EXAMPLES

45187 None.

45188 APPLICATION USAGE

45189 The *msync()* function is only supported if the Synchronized Input and Output option is

45190 supported, and thus need not be available on all implementations.

45191 The *msync()* function should be used by programs that require a memory object to be in a

45192 known state; for example, in building transaction facilities.

45193 Normal system activity can cause pages to be written to disk. Therefore, there are no guarantees

45194 that *msync()* is the only control over when pages are or are not written to disk.

45195 RATIONALE

45196 The *msync()* function writes out data in a mapped region to the permanent storage for the

45197 underlying object. The call to *msync()* ensures data integrity of the file.

45198 After the data is written out, any cached data may be invalidated if the MS_INVALIDATE flag

45199 was specified. This is useful on systems that do not support read/write consistency.

45200 FUTURE DIRECTIONS

45201 None.

45202 SEE ALSO

45203 *mmap()*, *sysconf()*

45204 XBD <sys/mman.h>

45205 CHANGE HISTORY

45206 First released in Issue 4, Version 2.

45207 Issue 5

45208 Moved from X/OPEN UNIX extension to BASE.

45209 Aligned with *msync()* in the POSIX Realtime Extension as follows:

- 45210 • The DESCRIPTION is extensively reworded.
- 45211 • [EBUSY] and a new form of [EINVAL] are added as mandatory error conditions.

45212 Issue 6

45213 The *msync()* function is marked as part of the Memory Mapped Files and Synchronized Input

45214 and Output options.

45215 The following changes are made for alignment with the ISO POSIX-1:1996 standard:

- 45216 • The [EBUSY] mandatory error condition is added.

45217 The following new requirements on POSIX implementations derive from alignment with the

45218 Single UNIX Specification:

- 45219 • The DESCRIPTION is updated to state that implementations require *addr* to be a multiple
- 45220 of the page size.
- 45221 • The second [EINVAL] error condition is made mandatory.
- 45222 The DESCRIPTION is updated for alignment with IEEE Std 1003.1j-2000 by adding reference to
- 45223 typed memory objects.
- 45224 **Issue 7**
- 45225 Austin Group Interpretation 1003.1-2001 #078 is applied, clarifying page alignment
- 45226 requirements.
- 45227 SD5-XSH-ERN-110 is applied.
- 45228 The *msync()* function is marked as part of the Synchronized Input and Output option or XSI
- 45229 option as the Memory Mapped Files is moved to the Base.
- 45230 Changes are made related to support for finegrained timestamps.

45231 NAME

45232 munlock — unlock a range of process address space

45233 SYNOPSIS

```
45234 MLR       #include <sys/mman.h>  
45235       int munlock(const void *addr, size_t len);
```

45236 DESCRIPTION

45237 Refer to *mlock()*.

45238 **NAME**

45239 munlockall — unlock the address space of a process

45240 **SYNOPSIS**

45241 ML #include <sys/mman.h>

45242 int munlockall(void);

45243 **DESCRIPTION**45244 Refer to *mlckall()*.

45245 **NAME**45246 **munmap** — unmap pages of memory45247 **SYNOPSIS**45248 `#include <sys/mman.h>`45249 `int munmap(void *addr, size_t len);`45250 **DESCRIPTION**

45251 The *munmap()* function shall remove any mappings for those entire pages containing any part of
 45252 the address space of the process starting at *addr* and continuing for *len* bytes. Further references
 45253 to these pages shall result in the generation of a SIGSEGV signal to the process. If there are no
 45254 mappings in the specified address range, then *munmap()* has no effect.

45255 The implementation may require that *addr* be a multiple of the page size as returned by
 45256 *sysconf()*.

45257 If a mapping to be removed was private, any modifications made in this address range shall be
 45258 discarded.

45259 ML|MLR Any memory locks (see *mlock()* and *mlockall()*) associated with this address range shall be
 45260 removed, as if by an appropriate call to *munlock()*.

45261 TYM If a mapping removed from a typed memory object causes the corresponding address range of
 45262 the memory pool to be inaccessible by any process in the system except through allocatable
 45263 mappings (that is, mappings of typed memory objects opened with the
 45264 POSIX_TYPED_MEM_MAP_ALLOCATABLE flag), then that range of the memory pool shall
 45265 become deallocated and may become available to satisfy future typed memory allocation
 45266 requests.

45267 A mapping removed from a typed memory object opened with the
 45268 POSIX_TYPED_MEM_MAP_ALLOCATABLE flag shall not affect in any way the availability of
 45269 that typed memory for allocation.

45270 The behavior of this function is unspecified if the mapping was not established by a call to
 45271 *mmap()*.

45272 **RETURN VALUE**

45273 Upon successful completion, *munmap()* shall return 0; otherwise, it shall return -1 and set *errno*
 45274 to indicate the error.

45275 **ERRORS**

45276 The *munmap()* function shall fail if:

45277 [EINVAL] Addresses in the range [*addr*,*addr*+*len*) are outside the valid range for the
 45278 address space of a process.

45279 [EINVAL] The *len* argument is 0.

45280 The *munmap()* function may fail if:

45281 [EINVAL] The *addr* argument is not a multiple of the page size as returned by *sysconf()*.

EXAMPLES

None.

APPLICATION USAGE

None.

RATIONALE

The *munmap()* function corresponds to SVR4, just as the *mmap()* function does.

It is possible that an application has applied process memory locking to a region that contains shared memory. If this has occurred, the *munmap()* call ignores those locks and, if necessary, causes those locks to be removed.

Most implementations require that *addr* is a multiple of the page size as returned by *sysconf()*.

FUTURE DIRECTIONS

None.

SEE ALSO

mlock(), *mlockall()*, *mmap()*, *posix_typed_mem_open()*, *sysconf()*

XBD <sys/mman.h>

CHANGE HISTORY

First released in Issue 4, Version 2.

Issue 5

Moved from X/OPEN UNIX extension to BASE.

Aligned with *munmap()* in the POSIX Realtime Extension as follows:

- The DESCRIPTION is extensively reworded.
- The SIGBUS error is no longer permitted to be generated.

Issue 6

The *munmap()* function is marked as part of the Memory Mapped Files and Shared Memory Objects option.

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The DESCRIPTION is updated to state that implementations require *addr* to be a multiple of the page size.
- The [EINVAL] error conditions are added.

The following changes are made for alignment with IEEE Std 1003.1j-2000:

- Semantics for typed memory objects are added to the DESCRIPTION.
- The *posix_typed_mem_open()* function is added to the SEE ALSO section.

IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/36 is applied, changing the margin code in the SYNOPSIS from MF|SHM to MC3 (notation for MF|SHM|TYM).

Issue 7

Austin Group Interpretation 1003.1-2001 #078 is applied, clarifying page alignment requirements.

The *munmap()* function is moved from the Memory Mapped Files option to the Base.

45321 **NAME**

45322 nan, nanf, nanl — return quiet NaN

45323 **SYNOPSIS**

```
45324 #include <math.h>
45325 double nan(const char *tagp);
45326 float nanf(const char *tagp);
45327 long double nanl(const char *tagp);
```

45328 **DESCRIPTION**

45329 CX The functionality described on this reference page is aligned with the ISO C standard. Any
45330 conflict between the requirements described here and the ISO C standard is unintentional. This
45331 volume of POSIX.1-2008 defers to the ISO C standard.

45332 The function call *nan("n-char-sequence")* shall be equivalent to:

```
45333 strtod("NAN(n-char-sequence)", (char **) NULL);
```

45334 The function call *nan("")* shall be equivalent to:

```
45335 strtod("NAN()", (char **) NULL)
```

45336 If *tagp* does not point to an *n-char* sequence or an empty string, the function call shall be
45337 equivalent to:

```
45338 strtod("NAN", (char **) NULL)
```

45339 Function calls to *nanf()* and *nanl()* are equivalent to the corresponding function calls to *strtod()*
45340 and *strtold()*.

45341 **RETURN VALUE**

45342 These functions shall return a quiet NaN, if available, with content indicated through *tagp*.

45343 If the implementation does not support quiet NaNs, these functions shall return zero.

45344 **ERRORS**

45345 No errors are defined.

45346 **EXAMPLES**

45347 None.

45348 **APPLICATION USAGE**

45349 None.

45350 **RATIONALE**

45351 None.

45352 **FUTURE DIRECTIONS**

45353 None.

45354 **SEE ALSO**

45355 [strtod\(\)](#)

45356 XBD [<math.h>](#)

45357 **CHANGE HISTORY**

45358 First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

NAME

nanosleep — high resolution sleep

SYNOPSIS

```
CX      #include <time.h>
45363      int nanosleep(const struct timespec *rqtp, struct timespec *rmtp);
```

DESCRIPTION

The *nanosleep()* function shall cause the current thread to be suspended from execution until either the time interval specified by the *rqtp* argument has elapsed or a signal is delivered to the calling thread, and its action is to invoke a signal-catching function or to terminate the process. The suspension time may be longer than requested because the argument value is rounded up to an integer multiple of the sleep resolution or because of the scheduling of other activity by the system. But, except for the case of being interrupted by a signal, the suspension time shall not be less than the time specified by *rqtp*, as measured by the system clock *CLOCK_REALTIME*.

The use of the *nanosleep()* function has no effect on the action or blockage of any signal.

RETURN VALUE

If the *nanosleep()* function returns because the requested time has elapsed, its return value shall be zero.

If the *nanosleep()* function returns because it has been interrupted by a signal, it shall return a value of *-1* and set *errno* to indicate the interruption. If the *rmtp* argument is non-NULL, the **timespec** structure referenced by it is updated to contain the amount of time remaining in the interval (the requested time minus the time actually slept). The *rqtp* and *rmtp* arguments may point to the same object. If the *rmtp* argument is NULL, the remaining time is not returned.

If *nanosleep()* fails, it shall return a value of *-1* and set *errno* to indicate the error.

ERRORS

The *nanosleep()* function shall fail if:

[EINTR]	The <i>nanosleep()</i> function was interrupted by a signal.
[EINVAL]	The <i>rqtp</i> argument specified a nanosecond value less than zero or greater than or equal to 1 000 million.

EXAMPLES

None.

APPLICATION USAGE

None.

RATIONALE

It is common to suspend execution of a thread for an interval in order to poll the status of a non-interrupting function. A large number of actual needs can be met with a simple extension to *sleep()* that provides finer resolution.

In the POSIX.1-1990 standard and SVR4, it is possible to implement such a routine, but the frequency of wakeup is limited by the resolution of the *alarm()* and *sleep()* functions. In 4.3 BSD, it is possible to write such a routine using no static storage and reserving no system facilities. Although it is possible to write a function with similar functionality to *sleep()* using the remainder of the *timer_**() functions, such a function requires the use of signals and the reservation of some signal number. This volume of POSIX.1-2008 requires that *nanosleep()* be non-intrusive of the signals function.

The *nanosleep()* function shall return a value of 0 on success and *-1* on failure or if interrupted.

45403 This latter case is different from *sleep()*. This was done because the remaining time is returned
45404 via an argument structure pointer, *rmtpt*, instead of as the return value.

45405 **FUTURE DIRECTIONS**

45406 None.

45407 **SEE ALSO**

45408 *clock_nanosleep()*, *sleep()*

45409 XBD <time.h>

45410 **CHANGE HISTORY**

45411 First released in Issue 5. Included for alignment with the POSIX Realtime Extension.

45412 **Issue 6**

45413 The *nanosleep()* function is marked as part of the Timers option.

45414 The [ENOSYS] error condition has been removed as stubs need not be provided if an
45415 implementation does not support the Timers option.

45416 IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/37 is applied, updating the SEE ALSO
45417 section to include the *clock_nanosleep()* function.

45418 IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/63 is applied, correcting text in the
45419 RATIONALE section.

45420 **Issue 7**

45421 SD5-XBD-ERN-33 is applied.

45422 The *nanosleep()* function is moved from the Timers option to the Base.

45423 **NAME**

45424 nearbyint, nearbyintf, nearbyintl — floating-point rounding functions

45425 **SYNOPSIS**

```
45426 #include <math.h>
45427 double nearbyint(double x);
45428 float nearbyintf(float x);
45429 long double nearbyintl(long double x);
```

45430 **DESCRIPTION**

45431 CX The functionality described on this reference page is aligned with the ISO C standard. Any
 45432 conflict between the requirements described here and the ISO C standard is unintentional. This
 45433 volume of POSIX.1-2008 defers to the ISO C standard.

45434 These functions shall round their argument to an integer value in floating-point format, using
 45435 the current rounding direction and without raising the inexact floating-point exception. -

45436 **RETURN VALUE**

45437 MX Upon successful completion, these functions shall return the rounded integer value. The result
 45438 shall have the same sign as x . +

45439 MX If x is NaN, a NaN shall be returned.

45440 If x is ± 0 , ± 0 shall be returned.

45441 If x is $\pm \text{Inf}$, x shall be returned. -

45442 **ERRORS**

45443 No errors are defined. |

45444 **EXAMPLES**

45445 None.

45446 **APPLICATION USAGE**

45447 The integral value returned by these functions need not be expressible as an `intmax_t`. The
 45448 return value should be tested before assigning it to an integer type to avoid the undefined
 45449 results of an integer overflow. |

45450 **RATIONALE**

45451 None.

45452 **FUTURE DIRECTIONS**

45453 None.

45454 **SEE ALSO**

45455 [*feclearexcept\(\)*](#), [*fetestexcept\(\)*](#)

45456 XBD [Section 4.19](#) (on page 116), [`<math.h>`](#)

45457 **CHANGE HISTORY**

45458 First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

45459 **Issue 7**

45460 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0402 [346,428] is applied. +

NAME

newlocale — create or modify a locale object

SYNOPSIS

```
CX    #include <locale.h>

    locale_t newlocale(int category_mask, const char *locale,
    locale_t base);
```

DESCRIPTION

The *newlocale()* function shall create a new locale object or modify an existing one. If the *base* argument is **(locale_t)0**, a new locale object shall be created. It is unspecified whether the locale object pointed to by *base* shall be modified, or freed and a new locale object created.

The *category_mask* argument specifies the locale categories to be set or modified. Values for *category_mask* shall be constructed by a bitwise-inclusive OR of the symbolic constants *LC_CTYPE_MASK*, *LC_NUMERIC_MASK*, *LC_TIME_MASK*, *LC_COLLATE_MASK*, *LC_MONETARY_MASK*, and *LC_MESSAGES_MASK*, or any of the other implementation-defined *LC_*_MASK* values defined in **<locale.h>**.

For each category with the corresponding bit set in *category_mask* the data from the locale named by *locale* shall be used. In the case of modifying an existing locale object, the data from the locale named by *locale* shall replace the existing data within the locale object. If a completely new locale object is created, the data for all sections not requested by *category_mask* shall be taken from the default locale.

The following preset values of *locale* are defined for all settings of *category_mask*:

"POSIX"	Specifies the minimal environment for C-language translation called the POSIX locale.
"C"	Equivalent to "POSIX".
" "	Specifies an implementation-defined native environment. This corresponds to the value of the associated environment variables, <i>LC_*</i> and <i>LANG</i> ; see XBD Chapter 7 (on page 135) and Chapter 8 (on page 173).

If the *base* argument is not **(locale_t)0** and the *newlocale()* function call succeeds, the contents of *base* are unspecified. Applications shall ensure that they stop using *base* as a locale object before calling *newlocale()*. If the function call fails and the *base* argument is not **(locale_t)0**, the contents of *base* shall remain valid and unchanged.

The behavior is undefined if the *base* argument is the special locale object *LC_GLOBAL_LOCALE*, or is not a valid locale object handle and is not **(locale_t)0**.

RETURN VALUE

Upon successful completion, the *newlocale()* function shall return a handle which the caller may use on subsequent calls to *duplocale()*, *freelocale()*, and other functions taking a **locale_t** argument.

Upon failure, the *newlocale()* function shall return **(locale_t)0** and set *errno* to indicate the error.

ERRORS

The *newlocale()* function shall fail if:

[ENOMEM]	There is not enough memory available to create the locale object or load the locale data.

45503 [EINVAL] The *category_mask* contains a bit that does not correspond to a valid category.

45504 [ENOENT] For any of the categories in *category_mask*, the locale data is not available.

45505 The *newlocale()* function may fail if:

45506 [EINVAL] The *locale* argument is not a valid string pointer.

45507 EXAMPLES

45508 Constructing a Locale Object from Different Locales

45509 The following example shows the construction of a locale where the *LC_CTYPE* category data |
 45510 comes from a locale *loc1* and the *LC_TIME* category data from a locale *tok2*:

```
45511 #include <locale.h>
45512 ...
45513 locale_t loc, new_loc;
45514 /* Get the "loc1" data. */
45515 loc = newlocale (LC_CTYPE_MASK, "loc1", (locale_t)0);
45516 if (loc == (locale_t) 0)
45517     abort ();
45518 /* Get the "loc2" data. */
45519 new_loc = newlocale (LC_TIME_MASK, "loc2", loc);
45520 if (new_loc != (locale_t) 0)
45521     /* We don't abort if this fails. In this case this
45522        simply used to unchanged locale object. */
45523     loc = new_loc;
45524 ...
```

45525 Freeing up a Locale Object

45526 The following example shows a code fragment to free a locale object created by *newlocale()*:

```
45527 #include <locale.h>
45528 ...
45529 /* Every locale object allocated with newlocale() should be
45530    * freed using freelocale():
45531    */
45532 locale_t loc;
45533 /* Get the locale. */
45534 loc = newlocale (LC_CTYPE_MASK | LC_TIME_MASK, "locname", (locale_t)0); |
45535 /* ... Use the locale object ... */
45536 ...
45537 /* Free the locale object resources. */
45538 freelocale (loc);
```

APPLICATION USAGE

Handles for locale objects created by the *newlocale()* function should either be released by a corresponding call to *freelocale()*, or be used as a base locale to another *newlocale()* call.

The special locale object LC_GLOBAL_LOCALE must not be passed for the *base* argument, even when returned by the *uselocale()* function.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

duplocale(), *freelocale()*, *uselocale()*

XBD [Chapter 7](#) (on page 135), [Chapter 8](#) (on page 173), [<locale.h>](#)

CHANGE HISTORY

First released in Issue 7.

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0403 [227], XSH/TC1-2008/0404 [283], + XSH/TC1-2008/0405 [295], and XSH/TC1-2008/0406 [227] are applied.

45555 **NAME**

45556 nextafter, nextafterf, nextafterl, nexttoward, nexttowardf, nexttowardl — next representable
 45557 floating-point number

45558 **SYNOPSIS**

```
45559       #include <math.h>

45560       double nextafter(double x, double y);
45561       float nextafterf(float x, float y);
45562       long double nextafterl(long double x, long double y);
45563       double nexttoward(double x, long double y);
45564       float nexttowardf(float x, long double y);
45565       long double nexttowardl(long double x, long double y);
```

45566 **DESCRIPTION**

45567 **CX** The functionality described on this reference page is aligned with the ISO C standard. Any
 45568 conflict between the requirements described here and the ISO C standard is unintentional. This
 45569 volume of POSIX.1-2008 defers to the ISO C standard.

45570 The *nextafter()*, *nextafterf()*, and *nextafterl()* functions shall compute the next representable
 45571 floating-point value following *x* in the direction of *y*. Thus, if *y* is less than *x*, *nextafter()* shall
 45572 return the largest representable floating-point number less than *x*. The *nextafter()*, *nextafterf()*,
 45573 and *nextafterl()* functions shall return *y* if *x* equals *y*.

45574 The *nexttoward()*, *nexttowardf()*, and *nexttowardl()* functions shall be equivalent to the
 45575 corresponding *nextafter()* functions, except that the second parameter shall have type **long**
 45576 **double** and the functions shall return *y* converted to the type of the function if *x* equals *y*.

45577 An application wishing to check for error situations should set *errno* to zero and call
 45578 *feclearexcept*(FE_ALL_EXCEPT) before calling these functions. On return, if *errno* is non-zero or
 45579 *fetestexcept*(FE_INVALID | FE_DIVBYZERO | FE_OVERFLOW | FE_UNDERFLOW) is non-
 45580 zero, an error has occurred.

45581 **RETURN VALUE**

45582 Upon successful completion, these functions shall return the next representable floating-point
 45583 value following *x* in the direction of *y*.

45584 If *x*=*y*, *y* (of the type *x*) shall be returned.

45585 If *x* is finite and the correct function value would overflow, a range error shall occur and
 45586 ±HUGE_VAL, ±HUGE_VALF, and ±HUGE_VALL (with the same sign as *x*) shall be returned as
 45587 appropriate for the return type of the function.

45588 **MX** If *x* or *y* is NaN, a NaN shall be returned.

45589 **MX** If *x*!=*y* and the correct function value is subnormal, zero, or underflows, a range error shall
 45590 occur, and

45591 **MXX** the correct function value (if representable) or

45592 **MX** 0.0 shall be returned.

45593 **ERRORS**

45594 These functions shall fail if:

45595 Range Error The correct value overflows.

45596 If the integer expression (*math_errhandling* & MATH_ERRNO) is non-zero,
 45597 then *errno* shall be set to [ERANGE]. If the integer expression
 45598 (*math_errhandling* & MATH_ERREXCEPT) is non-zero, then the overflow
 45599 floating-point exception shall be raised.

45600 MX **Range Error** The correct value is subnormal or underflows.
 45601 If the integer expression *(math_errhandling & MATH_ERRNO)* is non-zero,
 45602 then *errno* shall be set to [ERANGE]. If the integer expression
 45603 *(math_errhandling & MATH_ERREXCEPT)* is non-zero, then the underflow
 45604 floating-point exception shall be raised.

45605 EXAMPLES

45606 None.

45607 APPLICATION USAGE

45608 On error, the expressions *(math_errhandling & MATH_ERRNO)* and *(math_errhandling &*
 45609 *MATH_ERREXCEPT)* are independent of each other, but at least one of them must be non-zero.

45610 When `<tgmath.h>` is included, note that the return type of *nextafter()* depends on the generic +
 45611 typing deduced from both arguments, while the return type of *nexttoward()* depends only on the +
 45612 generic typing of the first argument.

45613 RATIONALE

45614 None.

45615 FUTURE DIRECTIONS

45616 None.

45617 SEE ALSO

45618 *feclearexcept()*, *fetestexcept()*

45619 XBD Section 4.19 (on page 116), `<math.h>`, `<tgmath.h>`

45620 CHANGE HISTORY

45621 First released in Issue 4, Version 2.

45622 Issue 5

45623 Moved from X/OPEN UNIX extension to BASE.

45624 Issue 6

45625 The *nextafter()* function is no longer marked as an extension. -

45626 The *nextafterf()*, *nextafterl()*, *nexttoward()*, *nexttowardf()*, and *nexttowardl()* functions are added
 45627 for alignment with the ISO/IEC 9899:1999 standard.

45628 The DESCRIPTION, RETURN VALUE, ERRORS, and APPLICATION USAGE sections are
 45629 revised to align with the ISO/IEC 9899:1999 standard.

45630 IEC 60559:1989 standard floating-point extensions over the ISO/IEC 9899:1999 standard are
 45631 marked.

45632 Issue 7

45633 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0407 [68] and XSH/TC1-2008/0408 +
 45634 [357] are applied.

NAME

nftw — walk a file tree

SYNOPSIS

```
XSI      #include <ftw.h>

int nftw(const char *path, int (*fn)(const char *,
    const struct stat *, int, struct FTW *), int fd_limit, int flags);
```

DESCRIPTION

The *nftw()* function shall recursively descend the directory hierarchy rooted in *path*. The *nftw()* function has a similar effect to *ftw()* except that it takes an additional argument *flags*, which is a bitwise-inclusive OR of zero or more of the following flags:

FTW_CHDIR If set, *nftw()* shall change the current working directory to each directory as it reports files in that directory. If clear, *nftw()* shall not change the current working directory.

FTW_DEPTH If set, *nftw()* shall report all files in a directory before reporting the directory itself. If clear, *nftw()* shall report any directory before reporting the files in that directory.

FTW_MOUNT If set, *nftw()* shall only report files in the same file system as *path*. If clear, *nftw()* shall report all files encountered during the walk.

FTW_PHYS If set, *nftw()* shall perform a physical walk and shall not follow symbolic links.

If **FTW_PHYS** is clear and **FTW_DEPTH** is set, *nftw()* shall follow links instead of reporting them, but shall not report any directory that would be a descendant of itself. If **FTW_PHYS** is clear and **FTW_DEPTH** is clear, *nftw()* shall follow links instead of reporting them, but shall not report the contents of any directory that would be a descendant of itself.

At each file it encounters, *nftw()* shall call the user-supplied function *fn* with four arguments:

- The first argument is the pathname of the object.
- The second argument is a pointer to the **stat** buffer containing information on the object, filled in as if *fstatat()*, *stat()*, or *lstat()* had been called to retrieve the information.
- The third argument is an integer giving additional information. Its value is one of the following:

FTW_D The object is a directory.

FTW_DNR The object is a directory that cannot be read. The *fn* function shall not be called for any of its descendants.

FTW_DP The object is a directory and subdirectories have been visited. (This condition shall only occur if the **FTW_DEPTH** flag is included in *flags*.)

FTW_F The object is a non-directory file.

FTW_NS The *stat()* function failed on the object because of lack of appropriate permission. The **stat** buffer passed to *fn* is undefined. Failure of *stat()* for any other reason is considered an error and *nftw()* shall return -1 .

FTW_SL The object is a symbolic link. (This condition shall only occur if the **FTW_PHYS** flag is included in *flags*.)

45675 FTW_SLN The object is a symbolic link that does not name an existing file. (This
45676 condition shall only occur if the FTW_PHYS flag is not included in *flags*.)

- 45677 • The fourth argument is a pointer to an **FTW** structure. The value of **base** is the offset of the
45678 object's filename in the pathname passed as the first argument to *fn*. The value of **level**
45679 indicates depth relative to the root of the walk, where the root level is 0.

45680 The results are unspecified if the application-supplied *fn* function does not preserve the current
45681 working directory.

45682 The argument *fd_limit* sets the maximum number of file descriptors that shall be used by *nftw()*
45683 while traversing the file tree. At most one file descriptor shall be used for each directory level.

45684 The *nftw()* function need not be thread-safe.

45685 RETURN VALUE

45686 The *nftw()* function shall continue until the first of the following conditions occurs:

- 45687 • An invocation of *fn* shall return a non-zero value, in which case *nftw()* shall return that
45688 value.
- 45689 • The *nftw()* function detects an error other than [EACCES] (see FTW_DNR and FTW_NS
45690 above), in which case *nftw()* shall return -1 and set *errno* to indicate the error.
- 45691 • The tree is exhausted, in which case *nftw()* shall return 0.

45692 ERRORS

45693 The *nftw()* function shall fail if:

45694 [EACCES] Search permission is denied for any component of *path* or read permission is
45695 denied for *path*, or *fn* returns -1 and does not reset *errno*.

45696 [ELOOP] A loop exists in symbolic links encountered during resolution of the *path*
45697 argument.

45698 [ENAMETOOLONG]
45699 The length of a component of a pathname is longer than {NAME_MAX}.

45700 [ENOENT] A component of *path* does not name an existing file or *path* is an empty string.

45701 [ENOTDIR] A component of *path* names an existing file that is neither a directory nor a
45702 symbolic link to a directory.

45703 [EOVERFLOW] A field in the **stat** structure cannot be represented correctly in the current
45704 programming environment for one or more files found in the file hierarchy.

45705 The *nftw()* function may fail if:

45706 [ELOOP] More than {SYMLOOP_MAX} symbolic links were encountered during
45707 resolution of the *path* argument.

45708 [EMFILE] All file descriptors available to the process are currently open.

45709 [ENAMETOOLONG]
45710 The length of a pathname exceeds {PATH_MAX}, or pathname resolution of a
45711 symbolic link produced an intermediate result with a length that exceeds
45712 {PATH_MAX}.

45713 [ENFILE] Too many files are currently open in the system.

45714 In addition, *errno* may be set if the function pointed to by *fn* causes *errno* to be set.

EXAMPLES

The following program traverses the directory tree under the path named in its first command-line argument, or under the current directory if no argument is supplied. It displays various information about each file. The second command-line argument can be used to specify characters that control the value assigned to the *flags* argument when calling *nftw()*.

```
#include <ftw.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stdint.h>

static int
display_info(const char *fpath, const struct stat *sb,
             int tflag, struct FTW *ftwbuf)
{
    printf("%-3s %2d %7jd    %-40s %d %s\n",
           (tflag == FTW_D) ? "d" : (tflag == FTW_DNR) ? "dnr" :
           (tflag == FTW_DP) ? "dp" : (tflag == FTW_F) ?
           (S_ISBLK(sb->st_mode) ? "f b" :
            S_ISCHR(sb->st_mode) ? "f c" :
            S_ISFIFO(sb->st_mode) ? "f p" :
            S_ISREG(sb->st_mode) ? "f r" :
            S_ISSOCK(sb->st_mode) ? "f s" : "f ?") :
           (tflag == FTW_NS) ? "ns" : (tflag == FTW_SL) ? "sl" :
           (tflag == FTW_SLN) ? "sln" : "?",
           ftwbuf->level, (intmax_t) sb->st_size,
           fpath, ftwbuf->base, fpath + ftwbuf->base);
    return 0; /* To tell nftw() to continue */
}

int
main(int argc, char *argv[])
{
    int flags = 0;

    if (argc > 2 && strchr(argv[2], 'd') != NULL)
        flags |= FTW_DEPTH;
    if (argc > 2 && strchr(argv[2], 'p') != NULL)
        flags |= FTW_PHYS;

    if (nftw((argc < 2) ? "." : argv[1], display_info, 20, flags) == -1)
    {
        perror("nftw");
        exit(EXIT_FAILURE);
    }

    exit(EXIT_SUCCESS);
}
```

APPLICATION USAGE

The *nftw()* function may allocate dynamic storage during its operation. If *nftw()* is forcibly terminated, such as by *longjmp()* or *siglongjmp()* being executed by the function pointed to by *fn* or an interrupt routine, *nftw()* does not have a chance to free that storage, so it remains permanently allocated. A safe way to handle interrupts is to store the fact that an interrupt has

45763 occurred, and arrange to have the function pointed to by *fn* return a non-zero value at its next
 45764 invocation.

45765 **RATIONALE**

45766 None.

45767 **FUTURE DIRECTIONS**

45768 None.

45769 **SEE ALSO**

45770 *fdopendir()*, *fstatat()*, *readdir()*

45771 XBD <ftw.h>

45772 **CHANGE HISTORY**

45773 First released in Issue 4, Version 2.

45774 **Issue 5**

45775 Moved from X/OPEN UNIX extension to BASE.

45776 In the DESCRIPTION, the definition of the *depth* argument is clarified.

45777 **Issue 6**

45778 The Open Group Base Resolution bwg97-003 is applied.

45779 The ERRORS section is updated as follows:

- 45780 • The wording of the mandatory [ELOOP] error condition is updated.
- 45781 • A second optional [ELOOP] error condition is added.
- 45782 • The [EOVERFLOW] mandatory error condition is added.

45783 Text is added to the DESCRIPTION to say that the *nftw()* function need not be reentrant and that
 45784 the results are unspecified if the application-supplied *fn* function does not preserve the current
 45785 working directory.

45786 IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/64 is applied, changing the argument
 45787 *depth* to *fd_limit* throughout and changing “to a maximum of 5 levels deep” to “using a
 45788 maximum of 5 file descriptors” in the EXAMPLES section.

45789 **Issue 7**

45790 Austin Group Interpretations 1003.1-2001 #143 and #156 are applied.

45791 SD5-XBD-ERN-4 is applied, changing the definition of the [EMFILE] error.

45792 SD5-XBD-ERN-61 is applied.

45793 APPLICATION USAGE is added and the EXAMPLES section is replaced with a new example.

45794 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0409 [403], XSH/TC1-2008/0410 [324], +
 45795 and XSH/TC1-2008/0411 [403] are applied.

NAME

nice — change the nice value of a process

SYNOPSIS

```
#include <unistd.h>

int nice(int incr);
```

DESCRIPTION

The *nice()* function shall add the value of *incr* to the nice value of the calling process. A nice value of a process is a non-negative number for which a more positive value shall result in less favorable scheduling.

A maximum nice value of $2 \times \{\text{NZERO}\} - 1$ and a minimum nice value of 0 shall be imposed by the system. Requests for values above or below these limits shall result in the nice value being set to the corresponding limit. Only a process with appropriate privileges can lower the nice value.

PS|TPS Calling the *nice()* function has no effect on the priority of processes or threads with policy SCHED_FIFO or SCHED_RR. The effect on processes or threads with other scheduling policies is implementation-defined.

The nice value set with *nice()* shall be applied to the process. If the process is multi-threaded, the nice value shall affect all system scope threads in the process.

As -1 is a permissible return value in a successful situation, an application wishing to check for error situations should set *errno* to 0, then call *nice()*, and if it returns -1 , check to see whether *errno* is non-zero.

RETURN VALUE

Upon successful completion, *nice()* shall return the new nice value $-\{\text{NZERO}\}$. Otherwise, -1 shall be returned, the nice value of the process shall not be changed, and *errno* shall be set to indicate the error.

ERRORS

The *nice()* function shall fail if:

[EPERM] The *incr* argument is negative and the calling process does not have appropriate privileges.

EXAMPLES**Changing the Nice Value**

The following example adds the value of the *incr* argument, -20 , to the nice value of the calling process.

```
#include <unistd.h>
...
int incr = -20;
int ret;

ret = nice(incr);
```

APPLICATION USAGE

None.

45835 **RATIONALE**

45836 None.

45837 **FUTURE DIRECTIONS**

45838 None.

45839 **SEE ALSO**

45840 *exec*, *getpriority()*

45841 XBD **<limits.h>**, **<unistd.h>**

45842 **CHANGE HISTORY**

45843 First released in Issue 1. Derived from Issue 1 of the SVID.

45844 **Issue 5**

45845 A statement is added to the description indicating the effects of this function on the different
45846 scheduling policies and multi-threaded processes.

NAME

nl_langinfo, nl_langinfo_l — language information

SYNOPSIS

```
#include <langinfo.h>
```

```
char *nl_langinfo(nl_item item);
```

```
char *nl_langinfo_l(nl_item item, locale_t locale);
```

DESCRIPTION

The *nl_langinfo()* and *nl_langinfo_l()* functions shall return a pointer to a string containing information relevant to the particular language or cultural area defined in the current locale, or in the locale represented by *locale*, respectively (see <langinfo.h>). The manifest constant names and values of *item* are defined in <langinfo.h>. For example:

```
nl_langinfo(ABDAY_1)
```

would return a pointer to the string "Dom" if the identified language was Portuguese, and "Sun" if the identified language was English.

```
nl_langinfo_l(ABDAY_1, loc)
```

would return a pointer to the string "Dom" if the identified language of the locale represented by *loc* was Portuguese, and "Sun" if the identified language of the locale represented by *loc* was English.

The *nl_langinfo()* function need not be thread-safe.

The behavior is undefined if the *locale* argument to *nl_langinfo_l()* is the special locale object LC_GLOBAL_LOCALE or is not a valid locale object handle.

RETURN VALUE

In a locale where *langinfo* data is not defined, these functions shall return a pointer to the corresponding string in the POSIX locale. In all locales, these functions shall return a pointer to an empty string if *item* contains an invalid setting.

The application shall not modify the string returned. The pointer returned by *nl_langinfo()* might be invalidated or the string content might be overwritten by a subsequent call to *nl_langinfo()* in any thread or to *nl_langinfo_l()* in the same thread or the initial thread, by subsequent calls to *setlocale()* with a category corresponding to the category of *item* (see <langinfo.h>) or the category LC_ALL, or by subsequent calls to *uselocale()* which change the category corresponding to the category of *item*. The pointer returned by *nl_langinfo_l()* might be invalidated or the string content might be overwritten by a subsequent call to *nl_langinfo_l()* in the same thread or to *nl_langinfo()* in any thread, or by subsequent calls to *freelocale()* or *newlocale()* which free or modify the locale object that was passed to *nl_langinfo_l()*.

ERRORS

No errors are defined.

EXAMPLES**Getting Date and Time Formatting Information**

The following example returns a pointer to a string containing date and time formatting information, as defined in the *LC_TIME* category of the current locale.

```
#include <time.h>
#include <langinfo.h>
...
strftime(datestring, sizeof(datestring), nl_langinfo(D_T_FMT), tm);
...
```

APPLICATION USAGE

The array pointed to by the return value should not be modified by the program, but may be modified by further calls to these functions.

RATIONALE

The possible interactions between internal data used by *nl_langinfo()* and *nl_langinfo_l()* are complicated by the fact that *nl_langinfo_l()* must be thread-safe but *nl_langinfo()* need not be. The various implementation choices are:

1. *nl_langinfo_l()* and *nl_langinfo()* use separate buffers, or at least one of them does not use an internal string buffer. In this case there are no interactions.
2. *nl_langinfo_l()* and *nl_langinfo()* share an internal per-thread buffer. There can be interactions, but only in the same thread.
3. *nl_langinfo_l()* uses an internal per-thread buffer, and *nl_langinfo()* uses (in all threads) the same buffer that *nl_langinfo_l()* uses in the initial thread. There can be interactions, but only when *nl_langinfo_l()* is called in the initial thread.

FUTURE DIRECTIONS

None.

SEE ALSO

setlocale(), *uselocale()*

XBD [Chapter 7](#) (on page 135), [<langinfo.h>](#), [<locale.h>](#), [<nl_types.h>](#)

CHANGE HISTORY

First released in Issue 2.

Issue 5

The last paragraph of the DESCRIPTION is moved from the APPLICATION USAGE section.

A note indicating that this function need not be reentrant is added to the DESCRIPTION.

Issue 7

Austin Group Interpretation 1003.1-2001 #156 is applied.

The *nl_langinfo()* function is moved from the XSI option to the Base.

The *nl_langinfo_l()* function is added from The Open Group Technical Standard, 2006, Extended API Set Part 4.

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0412 [302], XSH/TC1-2008/0413 [75], XSH/TC1-2008/0414 [283], XSH/TC1-2008/0415 [75,402], XSH/TC1-2008/0416 [283], and XSH/TC1-2008/0417 [402] are applied.

45924 **NAME**

45925 nrnd48 — generate uniformly distributed pseudo-random non-negative long integers

45926 **SYNOPSIS**

45927 XSI #include <stdlib.h>

45928 long nrnd48(unsigned short xsubi[3]);

45929 **DESCRIPTION**45930 Refer to *drnd48()*.

45931 **NAME**

45932 ntohl, ntohs — convert values between host and network byte order

45933 **SYNOPSIS**

45934 #include <arpa/inet.h>

45935 uint32_t ntohl(uint32_t *netlong*);45936 uint16_t ntohs(uint16_t *netshort*);45937 **DESCRIPTION**45938 Refer to *htonl()*.

NAME

open, openat — open file relative to directory file descriptor

SYNOPSIS

```
#include <sys/stat.h>
#include <fcntl.h>

int open(const char *path, int oflag, ...);
int openat(int fd, const char *path, int oflag, ...);
```

DESCRIPTION

The *open()* function shall establish the connection between a file and a file descriptor. It shall create an open file description that refers to a file and a file descriptor that refers to that open file description. The file descriptor is used by other I/O functions to refer to that file. The *path* argument points to a pathname naming the file.

The *open()* function shall return a file descriptor for the named file that is the lowest file descriptor not currently open for that process. The open file description is new, and therefore the file descriptor shall not share it with any other process in the system. The FD_CLOEXEC file descriptor flag associated with the new file descriptor shall be cleared unless the O_CLOEXEC flag is set in *oflag*.

The file offset used to mark the current position within the file shall be set to the beginning of the file.

The file status flags and file access modes of the open file description shall be set according to the value of *oflag*.

Values for *oflag* are constructed by a bitwise-inclusive OR of flags from the following list, defined in **<fcntl.h>**. Applications shall specify exactly one of the first five values (file access modes) below in the value of *oflag*:

O_EXEC Open for execute only (non-directory files). The result is unspecified if this flag is applied to a directory.

O_RDONLY Open for reading only.

O_RDWR Open for reading and writing. The result is undefined if this flag is applied to a FIFO.

O_SEARCH Open directory for search only. The result is unspecified if this flag is applied to a non-directory file.

O_WRONLY Open for writing only.

Any combination of the following may be used:

O_APPEND If set, the file offset shall be set to the end of the file prior to each write.

O_CLOEXEC If set, the FD_CLOEXEC flag for the new file descriptor shall be set.

O_CREAT If the file exists, this flag has no effect except as noted under **O_EXCL** below. Otherwise, the file shall be created; the user ID of the file shall be set to the effective user ID of the process; the group ID of the file shall be set to the group ID of the file's parent directory or to the effective group ID of the process; and the access permission bits (see **<sys/stat.h>**) of the file mode shall be set to the value of the argument following the *oflag* argument taken as type **mode_t** modified as follows: a bitwise AND is performed on the file-mode bits and the corresponding bits in the complement of the process' file mode creation mask. Thus, all bits in the

45983		file mode whose corresponding bit in the file mode creation mask is set	
45984		are cleared. When bits other than the file permission bits are set, the effect	
45985		is unspecified. The argument following the <i>oflag</i> argument does not affect	
45986		whether the file is open for reading, writing, or for both. Implementations	
45987		shall provide a way to initialize the file's group ID to the group ID of the	
45988		parent directory. Implementations may, but need not, provide an	
45989		implementation-defined way to initialize the file's group ID to the	
45990		effective group ID of the calling process.	
45991	O_DIRECTORY	If <i>path</i> resolves to a non-directory file, fail and set <i>errno</i> to [ENOTDIR].	
45992	SIO O_DSYNC	Write I/O operations on the file descriptor shall complete as defined by	
45993		synchronized I/O data integrity completion.	
45994	O_EXCL	If O_CREAT and O_EXCL are set, <i>open()</i> shall fail if the file exists. The	
45995		check for the existence of the file and the creation of the file if it does not	
45996		exist shall be atomic with respect to other threads executing <i>open()</i>	
45997		naming the same filename in the same directory with O_EXCL and	
45998		O_CREAT set. If O_EXCL and O_CREAT are set, and <i>path</i> names a	
45999		symbolic link, <i>open()</i> shall fail and set <i>errno</i> to [EEXIST], regardless of the	
46000		contents of the symbolic link. If O_EXCL is set and O_CREAT is not set,	
46001		the result is undefined.	
46002	O_NOCTTY	If set and <i>path</i> identifies a terminal device, <i>open()</i> shall not cause the	
46003		terminal device to become the controlling terminal for the process. If <i>path</i>	
46004		does not identify a terminal device, O_NOCTTY shall be ignored.	
46005	O_NOFOLLOW	If <i>path</i> names a symbolic link, fail and set <i>errno</i> to [ELOOP].	
46006	O_NONBLOCK	When opening a FIFO with O_RDONLY or O_WRONLY set:	
46007		• If O_NONBLOCK is set, an <i>open()</i> for reading-only shall return	
46008		without delay. An <i>open()</i> for writing-only shall return an error if no	
46009		process currently has the file open for reading.	
46010		• If O_NONBLOCK is clear, an <i>open()</i> for reading-only shall block the	
46011		calling thread until a thread opens the file for writing. An <i>open()</i> for	
46012		writing-only shall block the calling thread until a thread opens the	
46013		file for reading.	
46014		When opening a block special or character special file that supports non-	
46015		blocking opens:	
46016		• If O_NONBLOCK is set, the <i>open()</i> function shall return without	
46017		blocking for the device to be ready or available. Subsequent	
46018		behavior of the device is device-specific.	
46019		• If O_NONBLOCK is clear, the <i>open()</i> function shall block the calling	
46020		thread until the device is ready or available before returning.	
46021		Otherwise, the O_NONBLOCK flag shall not cause an error, but it is	
46022		unspecified whether the file status flags will include the O_NONBLOCK	
46023		flag.	
46024	SIO O_RSYNC	Read I/O operations on the file descriptor shall complete at the same	
46025		level of integrity as specified by the O_DSYNC and O_SYNC flags. If both	
46026		O_DSYNC and O_RSYNC are set in <i>oflag</i> , all I/O operations on the file	
46027		descriptor shall complete as defined by synchronized I/O data integrity	

46028			completion. If both O_SYNC and O_RSYNC are set in flags, all I/O	
46029			operations on the file descriptor shall complete as defined by	
46030			synchronized I/O file integrity completion.	
46031	XSI SIO	O_SYNC	Write I/O operations on the file descriptor shall complete as defined by	
46032			synchronized I/O file integrity completion.	
46033	XSI		The O_SYNC flag shall be supported for regular files, even if the	
46034			Synchronized Input and Output option is not supported.	
46035		O_TRUNC	If the file exists and is a regular file, and the file is successfully opened	
46036			O_RDWR or O_WRONLY, its length shall be truncated to 0, and the mode	
46037			and owner shall be unchanged. It shall have no effect on FIFO special files	
46038			or terminal device files. Its effect on other file types is implementation-	
46039			defined. The result of using O_TRUNC without either O_RDWR or	
46040			O_WRONLY is undefined.	
46041		O_TTY_INIT	If <i>path</i> identifies a terminal device other than a pseudo-terminal, the	
46042			device is not already open in any process, and either O_TTY_INIT is set in	
46043			<i>oflag</i> or O_TTY_INIT has the value zero, <i>open()</i> shall set any non-standard	
46044			termios structure terminal parameters to a state that provides conforming	
46045			behavior; see XBD Section 11.2 (on page 205). It is unspecified whether	
46046			O_TTY_INIT has any effect if the device is already open in any process. If	
46047			<i>path</i> identifies the slave side of a pseudo-terminal that is not already open	
46048			in any process, <i>open()</i> shall set any non-standard termios structure	
46049			terminal parameters to a state that provides conforming behavior,	
46050			regardless of whether O_TTY_INIT is set. If <i>path</i> does not identify a	
46051			terminal device, O_TTY_INIT shall be ignored.	
46052			If O_CREAT is set and the file did not previously exist, upon successful completion, <i>open()</i> shall	
46053			mark for update the last data access, last data modification, and last file status change	
46054			timestamps of the file and the last data modification and last file status change timestamps of	
46055			the parent directory.	
46056			If O_TRUNC is set and the file did previously exist, upon successful completion, <i>open()</i> shall	
46057			mark for update the last data modification and last file status change timestamps of the file.	
46058	SIO		If both the O_SYNC and O_DSYNC flags are set, the effect is as if only the O_SYNC flag was set.	
46059	OB XSR		If <i>path</i> refers to a STREAMS file, <i>oflag</i> may be constructed from O_NONBLOCK OR'ed with	
46060			either O_RDONLY, O_WRONLY, or O_RDWR. Other flag values are not applicable to STREAMS	
46061			devices and shall have no effect on them. The value O_NONBLOCK affects the operation of	
46062			STREAMS drivers and certain functions applied to file descriptors associated with STREAMS	
46063			files. For STREAMS drivers, the implementation of O_NONBLOCK is device-specific.	
46064			The application shall ensure that it specifies the O_TTY_INIT flag on the first open of a terminal	
46065			device since system boot or since the device was closed by the process that last had it open. The	
46066	XSI		application need not specify the O_TTY_INIT flag when opening pseudo-terminals. If <i>path</i>	
46067			names the master side of a pseudo-terminal device, then it is unspecified whether <i>open()</i> locks	
46068			the slave side so that it cannot be opened. Conforming applications shall call <i>unlockpt()</i> before	
46069			opening the slave side.	
46070			The largest value that can be represented correctly in an object of type off_t shall be established	
46071			as the offset maximum in the open file description.	
46072			The <i>openat()</i> function shall be equivalent to the <i>open()</i> function except in the case where <i>path</i>	

46073 specifies a relative path. In this case the file to be opened is determined relative to the directory
 46074 associated with the file descriptor *fd* instead of the current working directory. If the file
 46075 descriptor was opened without O_SEARCH, the function shall check whether directory searches
 46076 are permitted using the current permissions of the directory underlying the file descriptor. If the
 46077 file descriptor was opened with O_SEARCH, the function shall not perform the check.

46078 The *oflag* parameter and the optional fourth parameter correspond exactly to the parameters of
 46079 *open()*.

46080 If *openat()* is passed the special value AT_FDCWD in the *fd* parameter, the current working
 46081 directory shall be used and the behavior shall be identical to a call to *open()*.

46082 RETURN VALUE

46083 Upon successful completion, these functions shall open the file and return a non-negative
 46084 integer representing the lowest numbered unused file descriptor. Otherwise, these functions
 46085 shall return -1 and set *errno* to indicate the error. If -1 is returned, no files shall be created or
 46086 modified.

46087 ERRORS

46088 These functions shall fail if:

46089 [EACCES] Search permission is denied on a component of the path prefix, or the file
 46090 exists and the permissions specified by *oflag* are denied, or the file does not
 46091 exist and write permission is denied for the parent directory of the file to be
 46092 created, or O_TRUNC is specified and write permission is denied.

46093 [EEXIST] O_CREAT and O_EXCL are set, and the named file exists.

46094 [EINTR] A signal was caught during *open()*.

46095 SIO [EINVAL] The implementation does not support synchronized I/O for this file.

46096 OB XSR [EIO] The *path* argument names a STREAMS file and a hangup or error occurred
 46097 during the *open()*.

46098 [EISDIR] The named file is a directory and *oflag* includes O_WRONLY or O_RDWR.

46099 [ELOOP] A loop exists in symbolic links encountered during resolution of the *path*
 46100 argument, or O_NOFOLLOW was specified and the *path* argument names a
 46101 symbolic link.

46102 [EMFILE] All file descriptors available to the process are currently open.

46103 [ENAMETOOLONG]

46104 The length of a component of a pathname is longer than {NAME_MAX}.

46105 [ENFILE] The maximum allowable number of files is currently open in the system.

46106 [ENOENT] O_CREAT is not set and a component of *path* does not name an existing file, or
 46107 O_CREAT is set and a component of the path prefix of *path* does not name an
 46108 existing file, or *path* points to an empty string.

46109 [ENOENT] or [ENOTDIR]

46110 O_CREAT is set, and the *path* argument contains at least one non-*<slash>*
 46111 character and ends with one or more trailing *<slash>* characters. If *path* names
 46112 an existing file, an [ENOENT] error shall not occur.

46113 OB XSR [ENOSR] The *path* argument names a STREAMS-based file and the system is unable to
 46114 allocate a STREAM.

46115		[ENOSPC]	The directory or file system that would contain the new file cannot be expanded, the file does not exist, and O_CREAT is specified.	
46116				
46117		[ENOTDIR]	A component of the path prefix names an existing file that is neither a directory nor a symbolic link to a directory; or O_CREAT and O_EXCL are not specified, the <i>path</i> argument contains at least one non- <code><slash></code> character and ends with one or more trailing <code><slash></code> characters, and the last pathname component names an existing file that is neither a directory nor a symbolic link to a directory; or O_DIRECTORY was specified and the <i>path</i> argument resolves to a non-directory file.	
46118				
46119				
46120				
46121				
46122				
46123				
46124		[ENXIO]	O_NONBLOCK is set, the named file is a FIFO, O_WRONLY is set, and no process has the file open for reading.	
46125				
46126		[ENXIO]	The named file is a character special or block special file, and the device associated with this special file does not exist.	
46127				
46128		[EOVERFLOW]	The named file is a regular file and the size of the file cannot be represented correctly in an object of type <code>off_t</code> .	
46129				
46130		[EROFS]	The named file resides on a read-only file system and either O_WRONLY, O_RDWR, O_CREAT (if the file does not exist), or O_TRUNC is set in the <i>oflag</i> argument.	
46131				
46132				
46133		The <i>openat()</i> function shall fail if:		
46134		[EACCES]	<i>fd</i> was not opened with O_SEARCH and the permissions of the directory underlying <i>fd</i> do not permit directory searches.	
46135				
46136		[EBADF]	The <i>path</i> argument does not specify an absolute path and the <i>fd</i> argument is neither AT_FDCWD nor a valid file descriptor open for reading or searching.	
46137				
46138		[ENOTDIR]	The <i>path</i> argument is not an absolute path and <i>fd</i> is a file descriptor associated with a non-directory file.	+
46139				
46140		These functions may fail if:		
46141	XSI	[EAGAIN]	The <i>path</i> argument names the slave side of a pseudo-terminal device that is locked.	
46142				
46143		[EINVAL]	The value of the <i>oflag</i> argument is not valid.	
46144		[ELOOP]	More than {SYMLOOP_MAX} symbolic links were encountered during resolution of the <i>path</i> argument.	
46145				
46146		[ENAMETOOLONG]	The length of a pathname exceeds {PATH_MAX}, or pathname resolution of a symbolic link produced an intermediate result with a length that exceeds {PATH_MAX}.	
46147				
46148				
46149				
46150	OB XSR	[ENOMEM]	The <i>path</i> argument names a STREAMS file and the system is unable to allocate resources.	
46151				
46152		[ETXTBSY]	The file is a pure procedure (shared text) file that is being executed and <i>oflag</i> is O_WRONLY or O_RDWR.	
46153				

46154 **EXAMPLES**46155 **Opening a File for Writing by the Owner**

46156 The following example opens the file `/tmp/file`, either by creating it (if it does not already exist),
 46157 or by truncating its length to 0 (if it does exist). In the former case, if the call creates a new file,
 46158 the access permission bits in the file mode of the file are set to permit reading and writing by the
 46159 owner, and to permit reading only by group members and others.

46160 If the call to `open()` is successful, the file is opened for writing.

```
46161 #include <fcntl.h>
46162 ...
46163 int fd;
46164 mode_t mode = S_IRUSR | S_IWUSR | S_IRGRP | S_IROTH;
46165 char *pathname = "/tmp/file";
46166 ...
46167 fd = open(pathname, O_WRONLY | O_CREAT | O_TRUNC, mode);
46168 ...
```

46169 **Opening a File Using an Existence Check**

46170 The following example uses the `open()` function to try to create the **LOCKFILE** file and open it
 46171 for writing. Since the `open()` function specifies the `O_EXCL` flag, the call fails if the file already
 46172 exists. In that case, the program assumes that someone else is updating the password file and
 46173 exits.

```
46174 #include <fcntl.h>
46175 #include <stdio.h>
46176 #include <stdlib.h>
46177 #define LOCKFILE "/etc/ptmp"
46178 ...
46179 int pfd; /* Integer for file descriptor returned by open() call. */
46180 ...
46181 if ((pfd = open(LOCKFILE, O_WRONLY | O_CREAT | O_EXCL,
46182               S_IRUSR | S_IWUSR | S_IRGRP | S_IROTH)) == -1)
46183 {
46184     fprintf(stderr, "Cannot open /etc/ptmp. Try again later.\n");
46185     exit(1);
46186 }
46187 ...
```

46188 **Opening a File for Writing**

46189 The following example opens a file for writing, creating the file if it does not already exist. If the
 46190 file does exist, the system truncates the file to zero bytes.

```
46191 #include <fcntl.h>
46192 #include <stdio.h>
46193 #include <stdlib.h>
46194 #define LOCKFILE "/etc/ptmp"
46195 ...
46196 int pfd;
```

```

46197 char pathname[PATH_MAX+1];
46198 ...
46199 if ((pfd = open(pathname, O_WRONLY | O_CREAT | O_TRUNC,
46200 S_IRUSR | S_IWUSR | S_IRGRP | S_IROTH)) == -1)
46201 {
46202     perror("Cannot open output file\n"); exit(1);
46203 }
46204 ...

```

APPLICATION USAGE

POSIX.1-2008 does not require that terminal parameters be automatically set to any state on first open, nor that they be reset after the last close. It is possible for a non-conforming application to leave a terminal device in a state where the next process to use that device finds it in a non-conforming state, but has no way of determining this. To ensure that the device is set to a conforming initial state, applications which perform a first open of a terminal (other than a pseudo-terminal) should do so using the `O_TTY_INIT` flag to set the parameters associated with the terminal to a conforming state.

Except as specified in this volume of POSIX.1-2008, the flags allowed in *oflag* are not mutually-exclusive and any number of them may be used simultaneously. Not all combinations of flags make sense. For example, using `O_SEARCH | O_CREAT` will successfully open a pre-existing directory for searching, but if there is no existing file by that name, then it is unspecified whether a regular file will be created. Likewise, if a non-directory file descriptor is successfully returned, it is unspecified whether that descriptor will have execute permissions as if by `O_EXEC` (note that it is unspecified whether `O_EXEC` and `O_SEARCH` have the same value).

RATIONALE

Some implementations permit opening FIFOs with `O_RDWR`. Since FIFOs could be implemented in other ways, and since two file descriptors can be used to the same effect, this possibility is left as undefined.

See [getgroups\(\)](#) about the group of a newly created file.

The use of `open()` to create a regular file is preferable to the use of `creat()`, because the latter is redundant and included only for historical reasons.

The use of the `O_TRUNC` flag on FIFOs and directories (pipes cannot be `open()`-ed) must be permissible without unexpected side-effects (for example, `creat()` on a FIFO must not remove data). Since terminal special files might have type-ahead data stored in the buffer, `O_TRUNC` should not affect their content, particularly if a program that normally opens a regular file should open the current controlling terminal instead. Other file types, particularly implementation-defined ones, are left implementation-defined.

POSIX.1-2008 permits `[EACCES]` to be returned for conditions other than those explicitly listed.

The `O_NOCTTY` flag was added to allow applications to avoid unintentionally acquiring a controlling terminal as a side-effect of opening a terminal file. This volume of POSIX.1-2008 does not specify how a controlling terminal is acquired, but it allows an implementation to provide this on `open()` if the `O_NOCTTY` flag is not set and other conditions specified in XBD [Chapter 11](#) (on page 199) are met.

In historical implementations the value of `O_RDONLY` is zero. Because of that, it is not possible to detect the presence of `O_RDONLY` and another option. Future implementations should encode `O_RDONLY` and `O_WRONLY` as bit flags so that:

```
O_RDONLY | O_WRONLY == O_RDWR
```

O_EXEC and O_SEARCH are specified as two of the five file access modes. Since O_EXEC does not apply to directories, and O_SEARCH only applies to directories, their values need not be distinct. Since O_RDONLY has historically had the value zero, implementations are not able to distinguish between O_SEARCH and O_SEARCH | O_RDONLY, and similarly for O_EXEC.

In general, the *open()* function follows the symbolic link if *path* names a symbolic link. However, the *open()* function, when called with O_CREAT and O_EXCL, is required to fail with [EEXIST] if *path* names an existing symbolic link, even if the symbolic link refers to a nonexistent file. This behavior is required so that privileged applications can create a new file in a known location without the possibility that a symbolic link might cause the file to be created in a different location.

For example, a privileged application that must create a file with a predictable name in a user-writable directory, such as the user's home directory, could be compromised if the user creates a symbolic link with that name that refers to a nonexistent file in a system directory. If the user can influence the contents of a file, the user could compromise the system by creating a new system configuration or spool file that would then be interpreted by the system. The test for a symbolic link which refers to a nonexistent file must be atomic with the creation of a new file.

In addition, the *open()* function refuses to open non-directories if the O_DIRECTORY flag is set. This avoids race conditions whereby a user might compromise the system by substituting a hard link to a sensitive file (e.g., a device or a FIFO) while a privileged application is running, where opening a file even for read access might have undesirable side-effects.

In addition, the *open()* function does not follow symbolic links if the O_NOFOLLOW flag is set. This avoids race conditions whereby a user might compromise the system by substituting a symbolic link to a sensitive file (e.g., a device) while a privileged application is running, where opening a file even for read access might have undesirable side-effects.

The POSIX.1-1990 standard required that the group ID of a newly created file be set to the group ID of its parent directory or to the effective group ID of the creating process. FIPS 151-2 required that implementations provide a way to have the group ID be set to the group ID of the containing directory, but did not prohibit implementations also supporting a way to set the group ID to the effective group ID of the creating process. Conforming applications should not assume which group ID will be used. If it matters, an application can use *chown()* to set the group ID after the file is created, or determine under what conditions the implementation will set the desired group ID.

The purpose of the *openat()* function is to enable opening files in directories other than the current working directory without exposure to race conditions. Any part of the path of a file could be changed in parallel to a call to *open()*, resulting in unspecified behavior. By opening a file descriptor for the target directory and using the *openat()* function it can be guaranteed that the opened file is located relative to the desired directory. Some implementations use the *openat()* function for other purposes as well. In some cases, if the *oflag* parameter has the O_XATTR bit set, the returned file descriptor provides access to extended attributes. This functionality is not standardized here.

FUTURE DIRECTIONS

None.

SEE ALSO

chmod(), *close()*, *creat()*, *dirfd()*, *dup()*, *exec*, *fcntl()*, *fdopendir()*, *link()*, *lseek()*, *mkdtemp()*, *mknod()*, *read()*, *symlink()*, *umask()*, *unlockpt()*, *write()*

XBD Chapter 11 (on page 199), [<fcntl.h>](#), [<sys/stat.h>](#), [<sys/types.h>](#)

CHANGE HISTORY

First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 5

The DESCRIPTION is updated for alignment with the POSIX Realtime Extension and the POSIX Threads Extension.

Large File Summit extensions are added.

Issue 6

In the SYNOPSIS, the optional include of the `<sys/types.h>` header is removed.

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The requirement to include `<sys/types.h>` has been removed. Although `<sys/types.h>` was required for conforming implementations of previous POSIX specifications, it was not required for UNIX applications.
- In the DESCRIPTION, `O_CREAT` is amended to state that the group ID of the file is set to the group ID of the file's parent directory or to the effective group ID of the process. This is a FIPS requirement.
- In the DESCRIPTION, text is added to indicate setting of the offset maximum in the open file description. This change is to support large files.
- In the ERRORS section, the `[EOVERFLOW]` condition is added. This change is to support large files.
- The `[ENXIO]` mandatory error condition is added.
- The `[EINVAL]`, `[ENAMETOOLONG]`, and `[ETXTBSY]` optional error conditions are added.

The DESCRIPTION and ERRORS sections are updated so that items related to the optional XSI STREAMS Option Group are marked.

The following changes were made to align with the IEEE P1003.1a draft standard:

- An explanation is added of the effect of the `O_CREAT` and `O_EXCL` flags when the path refers to a symbolic link.
- The `[ELOOP]` optional error condition is added.

The normative text is updated to avoid use of the term “must” for application requirements.

The DESCRIPTION of `O_EXCL` is updated in response to IEEE PASC Interpretation 1003.1c #48.

Issue 7

Austin Group Interpretations 1003.1-2001 #113 and #143 are applied.

Austin Group Interpretation 1003.1-2001 #144 is applied, adding the `O_TTY_INIT` flag.

Austin Group Interpretation 1003.1-2001 #171 is applied, adding support to set the `FD_CLOEXEC` flag atomically at `open()`, and adding the `F_DUPFD_CLOEXEC` flag.

SD5-XBD-ERN-4 is applied, changing the definition of the `[EMFILE]` error.

This page is revised and the `openat()` function is added from The Open Group Technical Standard, 2006, Extended API Set Part 2.

Functionality relating to the XSI STREAMS option is marked obsolescent.

Changes are made related to support for finegrained timestamps.

46329 Changes are made to allow a directory to be opened for searching.

46330 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0418 [292], XSH/TC1-2008/0419 [141], +

46331 XSH/TC1-2008/0420 [461], XSH/TC1-2008/0421 [390], XSH/TC1-2008/0422 [146], +

46332 XSH/TC1-2008/0423 [324], XSH/TC1-2008/0424 [292], XSH/TC1-2008/0425 [278], +

46333 XSH/TC1-2008/0426 [278], XSH/TC1-2008/0427 [291], and XSH/TC1-2008/0428 [307] are +

46334 applied.

NAME

open_memstream, open_wmemstream — open a dynamic memory buffer stream

SYNOPSIS

```
CX      #include <stdio.h>
FILE *open_memstream(char **bufp, size_t *sizep);
#include <wchar.h>
FILE *open_wmemstream(wchar_t **bufp, size_t *sizep);
```

DESCRIPTION

The *open_memstream()* and *open_wmemstream()* functions shall create an I/O stream associated with a dynamically allocated memory buffer. The stream shall be opened for writing and shall be seekable.

The stream associated with a call to *open_memstream()* shall be byte-oriented.

The stream associated with a call to *open_wmemstream()* shall be wide-oriented.

The stream shall maintain a current position in the allocated buffer and a current buffer length. The position shall be initially set to zero (the start of the buffer). Each write to the stream shall start at the current position and move this position by the number of successfully written bytes for *open_memstream()* or the number of successfully written wide characters for *open_wmemstream()*. The length shall be initially set to zero. If a write moves the position to a value larger than the current length, the current length shall be set to this position. In this case a null character for *open_memstream()* or a null wide character for *open_wmemstream()* shall be appended to the current buffer. For both functions the terminating null is not included in the calculation of the buffer length.

After a successful *fflush()* or *fclose()*, the pointer referenced by *bufp* shall contain the address of the buffer, and the variable pointed to by *sizep* shall contain the smaller of the current buffer length and the number of bytes for *open_memstream()*, or the number of wide characters for *open_wmemstream()*, between the beginning of the buffer and the current file position indicator.

After a successful *fflush()* the pointer referenced by *bufp* and the variable referenced by *sizep* remain valid only until the next write operation on the stream or a call to *fclose()*.

RETURN VALUE

Upon successful completion, these functions shall return a pointer to the object controlling the stream. Otherwise, a null pointer shall be returned, and *errno* shall be set to indicate the error.

ERRORS

These functions may fail if:

[EINVAL]	<i>bufp</i> or <i>sizep</i> are NULL.
[EMFILE]	{FOPEN_MAX} streams are currently open in the calling process.
[ENOMEM]	Memory for the stream or the buffer could not be allocated.

EXAMPLES

```

#include <stdio.h>
#include <stdlib.h>

int
main (void)
{
    FILE *stream;
    char *buf;
    size_t len;
    off_t eob;

    stream = open_memstream (&buf, &len);
    if (stream == NULL)
        /* handle error */ ;
    fprintf (stream, "hello my world");
    fflush (stream);
    printf ("buf=%s, len=%zu\n", buf, len);
    eob = ftello(stream);
    fseeko (stream, 0, SEEK_SET);
    fprintf (stream, "good-bye");
    fseeko (stream, eob, SEEK_SET);
    fclose (stream);
    printf ("buf=%s, len=%zu\n", buf, len);
    free (buf);
    return 0;
}

```

This program produces the following output:

```

buf=hello my world, len=14
buf=good-bye world, len=14

```

APPLICATION USAGE

The buffer created by these functions should be freed by the application after closing the stream, by means of a call to *free()*.

RATIONALE

These functions are similar to *fmemopen()* except that the memory is always allocated dynamically by the function, and the stream is opened only for output.

FUTURE DIRECTIONS

None.

SEE ALSO

fclose(), *fdopen()*, *fflush()*, *fmemopen()*, *fopen()*, *free()*, *freopen()*

XBD [*<stdio.h>*](#), [*<wchar.h>*](#)

CHANGE HISTORY

First released in Issue 7.

46412 **NAME**

46413 openat — open file relative to directory file descriptor

46414 **SYNOPSIS**

46415 #include <fcntl.h>

46416 int openat(int *fd*, const char **path*, int *oflag*, ...);46417 **DESCRIPTION**46418 Refer to *open()*.

NAME

opendir — open directory associated with file descriptor

SYNOPSIS

```
#include <dirent.h>
```

```
DIR *opendir(const char *dirname);
```

DESCRIPTION

Refer to *fdopendir()*.

46426 **NAME**

46427 openlog — open a connection to the logging facility

46428 **SYNOPSIS**

46429 XSI #include <syslog.h>

46430 void openlog(const char **ident*, int *logopt*, int *facility*);46431 **DESCRIPTION**46432 Refer to *closelog()*.

46433 **NAME**

46434 optarg, opterr, optind, optopt — options parsing variables

46435 **SYNOPSIS**

46436 #include <unistd.h>

46437 extern char *optarg;

46438 extern int opterr, optind, optopt;

46439 **DESCRIPTION**46440 Refer to *getopt()*.

46441 **NAME**

46442 pathconf — get configurable pathname variables

46443 **SYNOPSIS**

46444 #include <unistd.h>

46445 long pathconf(const char **path*, int *name*);46446 **DESCRIPTION**46447 Refer to *fpathconf()*.

46448 **NAME**

46449 pause — suspend the thread until a signal is received

46450 **SYNOPSIS**

46451 #include <unistd.h>

46452 int pause(void);

46453 **DESCRIPTION**

46454 The *pause()* function shall suspend the calling thread until delivery of a signal whose action is
46455 either to execute a signal-catching function or to terminate the process.

46456 If the action is to terminate the process, *pause()* shall not return.

46457 If the action is to execute a signal-catching function, *pause()* shall return after the signal-catching
46458 function returns.

46459 **RETURN VALUE**

46460 Since *pause()* suspends thread execution indefinitely unless interrupted by a signal, there is no
46461 successful completion return value. A value of -1 shall be returned and *errno* set to indicate the
46462 error.

46463 **ERRORS**

46464 The *pause()* function shall fail if:

46465 [EINTR] A signal is caught by the calling process and control is returned from the
46466 signal-catching function.

46467 **EXAMPLES**

46468 None.

46469 **APPLICATION USAGE**

46470 Many common uses of *pause()* have timing windows. The scenario involves checking a
46471 condition related to a signal and, if the signal has not occurred, calling *pause()*. When the signal
46472 occurs between the check and the call to *pause()*, the process often blocks indefinitely. The
46473 *sigprocmask()* and *sigsuspend()* functions can be used to avoid this type of problem.

46474 **RATIONALE**

46475 None.

46476 **FUTURE DIRECTIONS**

46477 None.

46478 **SEE ALSO**

46479 [sigsuspend\(\)](#)

46480 XBD [<unistd.h>](#)

46481 **CHANGE HISTORY**

46482 First released in Issue 1. Derived from Issue 1 of the SVID.

46483 **Issue 5**

46484 The DESCRIPTION is updated for alignment with the POSIX Threads Extension.

46485 **Issue 6**

46486 The APPLICATION USAGE section is added.

NAME

pclose — close a pipe stream to or from a process

SYNOPSIS

```
CX      #include <stdio.h>
int pclose(FILE *stream);
```

DESCRIPTION

The *pclose()* function shall close a stream that was opened by *popen()*, wait for the command to terminate, and return the termination status of the process that was running the command language interpreter. However, if a call caused the termination status to be unavailable to *pclose()*, then *pclose()* shall return -1 with *errno* set to [ECHILD] to report this situation. This can happen if the application calls one of the following functions:

- *wait()*
- *waitpid()* with a *pid* argument less than or equal to 0 or equal to the process ID of the command line interpreter
- Any other function not defined in this volume of POSIX.1-2008 that could do one of the above

In any case, *pclose()* shall not return before the child process created by *popen()* has terminated.

If the command language interpreter cannot be executed, the child termination status returned by *pclose()* shall be as if the command language interpreter terminated using *exit*(127) or *_exit*(127).

The *pclose()* function shall not affect the termination status of any child of the calling process other than the one created by *popen()* for the associated stream.

If the argument *stream* to *pclose()* is not a pointer to a stream created by *popen()*, the result of *pclose()* is undefined.

RETURN VALUE

Upon successful return, *pclose()* shall return the termination status of the command language interpreter. Otherwise, *pclose()* shall return -1 and set *errno* to indicate the error.

ERRORS

The *pclose()* function shall fail if:

[ECHILD] The status of the child process could not be obtained, as described above.

EXAMPLES

None.

APPLICATION USAGE

None.

RATIONALE

There is a requirement that *pclose()* not return before the child process terminates. This is intended to disallow implementations that return [EINTR] if a signal is received while waiting. If *pclose()* returned before the child terminated, there would be no way for the application to discover which child used to be associated with the stream, and it could not do the cleanup itself.

If the stream pointed to by *stream* was not created by *popen()*, historical implementations of *pclose()* return -1 without setting *errno*. To avoid requiring *pclose()* to set *errno* in this case, POSIX.1-2008 makes the behavior unspecified. An application should not use *pclose()* to close

any stream that was not created by *popen()*.

Some historical implementations of *pclose()* either block or ignore the signals SIGINT, SIGQUIT, and SIGHUP while waiting for the child process to terminate. Since this behavior is not described for the *pclose()* function in POSIX.1-2008, such implementations are not conforming. Also, some historical implementations return [EINTR] if a signal is received, even though the child process has not terminated. Such implementations are also considered non-conforming.

Consider, for example, an application that uses:

```
popen("command", "r")
```

to start *command*, which is part of the same application. The parent writes a prompt to its standard output (presumably the terminal) and then reads from the *popen()*ed stream. The child reads the response from the user, does some transformation on the response (pathname expansion, perhaps) and writes the result to its standard output. The parent process reads the result from the pipe, does something with it, and prints another prompt. The cycle repeats. Assuming that both processes do appropriate buffer flushing, this would be expected to work.

To conform to POSIX.1-2008, *pclose()* must use *waitpid()*, or some similar function, instead of *wait()*.

The code sample below illustrates how the *pclose()* function might be implemented on a system conforming to POSIX.1-2008.

```
int pclose(FILE *stream)
{
    int stat;
    pid_t pid;

    pid = <pid for process created for stream by popen()>
    (void) fclose(stream);
    while (waitpid(pid, &stat, 0) == -1) {
        if (errno != EINTR) {
            stat = -1;
            break;
        }
    }
    return(stat);
}
```

FUTURE DIRECTIONS

None.

SEE ALSO

fork(), *popen()*, *wait()*

XBD <stdio.h>

CHANGE HISTORY

First released in Issue 1. Derived from Issue 1 of the SVID.

46569 **NAME**

46570 perror — write error messages to standard error

46571 **SYNOPSIS**

46572 #include <stdio.h>

46573 void perror(const char *s);

46574 **DESCRIPTION**

46575 CX The functionality described on this reference page is aligned with the ISO C standard. Any
 46576 conflict between the requirements described here and the ISO C standard is unintentional. This
 46577 volume of POSIX.1-2008 defers to the ISO C standard.

46578 The *perror()* function shall map the error number accessed through the symbol *errno* to a
 46579 language-dependent error message, which shall be written to the standard error stream as
 46580 follows:

- 46581 • First (if *s* is not a null pointer and the character pointed to by *s* is not the null byte), the
 46582 string pointed to by *s* followed by a <colon> and a <space>.
- 46583 • Then an error message string followed by a <newline>.

46584 The contents of the error message strings shall be the same as those returned by *strerror()* with
 46585 argument *errno*.

46586 CX The *perror()* function shall mark for update the last data modification and last file status change
 46587 timestamps of the file associated with the standard error stream at some time between its
 46588 successful completion and *exit()*, *abort()*, or the completion of *fflush()* or *fclose()* on *stderr*.

46589 The *perror()* function shall not change the orientation of the standard error stream.

46590 On error, *perror()* shall set the error indicator for the stream to which *stderr* points, and shall set
 46591 *errno* to indicate the error.

46592 Since no value is returned, an application wishing to check for error situations should call
 46593 *clearerr(stderr)* before calling *perror()*, then if *ferror(stderr)* returns non-zero, the value of *errno*
 46594 indicates which error occurred.

46595 **RETURN VALUE**46596 The *perror()* function shall not return a value.46597 **ERRORS**46598 CX Refer to *fputc()*.46599 **EXAMPLES**46600 **Printing an Error Message for a Function**

46601 The following example replaces *bufptr* with a buffer that is the necessary size. If an error occurs,
 46602 the *perror()* function prints a message and the program exits.

```

46603 #include <stdio.h>
46604 #include <stdlib.h>
46605 ...
46606 char *bufptr;
46607 size_t szbuf;
46608 ...
46609 if ((bufptr = malloc(szbuf)) == NULL) {
46610     perror("malloc"); exit(2);
46611 }
```

46612 . . .

46613 **APPLICATION USAGE**

46614 Application writers may prefer to use alternative interfaces instead of *perror()*, such as |
46615 *strerror_r()* in combination with *fprintf()*.

46616 **RATIONALE**

46617 None.

46618 **FUTURE DIRECTIONS**

46619 None.

46620 **SEE ALSO**

46621 *fprintf()*, *fputc()*, *psiginfo()*, *strerror()*

46622 XBD <stdio.h>

46623 **CHANGE HISTORY**

46624 First released in Issue 1. Derived from Issue 1 of the SVID.

46625 **Issue 5**

46626 A paragraph is added to the DESCRIPTION indicating that *perror()* does not change the
46627 orientation of the standard error stream.

46628 **Issue 6**

46629 Extensions beyond the ISO C standard are marked.

46630 **Issue 7**

46631 SD5-XSH-ERN-95 is applied.

46632 Changes are made related to support for finegrained timestamps.

46633 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0429 [389,401], XSH/TC1-2008/0430 +
46634 [389], and XSH/TC1-2008/0431 [389,401] are applied.

46635 **NAME**

46636 pipe — create an interprocess channel

46637 **SYNOPSIS**

46638 #include <unistd.h>

46639 int pipe(int *fildes*[2]);46640 **DESCRIPTION**

46641 The *pipe()* function shall create a pipe and place two file descriptors, one each into the
 46642 arguments *fildes*[0] and *fildes*[1], that refer to the open file descriptions for the read and write
 46643 ends of the pipe. Their integer values shall be the two lowest available at the time of the *pipe()*
 46644 call. The O_NONBLOCK and FD_CLOEXEC flags shall be clear on both file descriptors. (The
 46645 *fcntl()* function can be used to set both these flags.)

46646 Data can be written to the file descriptor *fildes*[1] and read from the file descriptor *fildes*[0]. A
 46647 read on the file descriptor *fildes*[0] shall access data written to the file descriptor *fildes*[1] on a
 46648 first-in-first-out basis. It is unspecified whether *fildes*[0] is also open for writing and whether
 46649 *fildes*[1] is also open for reading.

46650 A process has the pipe open for reading (correspondingly writing) if it has a file descriptor open
 46651 that refers to the read end, *fildes*[0] (write end, *fildes*[1]).

46652 The pipe's user ID shall be set to the effective user ID of the calling process.

46653 The pipe's group ID shall be set to the effective group ID of the calling process.

46654 Upon successful completion, *pipe()* shall mark for update the last data access, last data
 46655 modification, and last file status change timestamps of the pipe.

46656 **RETURN VALUE**

46657 Upon successful completion, 0 shall be returned; otherwise, -1 shall be returned and *errno* set to
 46658 indicate the error.

46659 **ERRORS**

46660 The *pipe()* function shall fail if:

46661 [EMFILE] All, or all but one, of the file descriptors available to the process are currently
 46662 open.

46663 [ENFILE] The number of simultaneously open files in the system would exceed a
 46664 system-imposed limit.

46665 **EXAMPLES**46666 **Using a Pipe to Pass Data Between a Parent Process and a Child Process**

46667 The following example demonstrates the use of a pipe to transfer data between a parent process
 46668 and a child process. Error handling is excluded, but otherwise this code demonstrates good
 46669 practice when using pipes: after the *fork()* the two processes close the unused ends of the pipe
 46670 before they commence transferring data.

46671 #include <stdlib.h>

46672 #include <unistd.h>

46673 ...

46674 int fildes[2];

46675 const int BSIZE = 100;

46676 char buf[BSIZE];

46677 ssize_t nbytes;

```

46678     int status;
46679     status = pipe(fildes);
46680     if (status == -1 ) {
46681         /* an error occurred */
46682         ...
46683     }
46684     switch (fork()) {
46685     case -1: /* Handle error */
46686         break;
46687     case 0: /* Child - reads from pipe */
46688         close(fildes[1]); /* Write end is unused */
46689         nbytes = read(fildes[0], buf, BSIZE); /* Get data from pipe */
46690         /* At this point, a further read would see end of file ... */
46691         close(fildes[0]); /* Finished with pipe */
46692         exit(EXIT_SUCCESS);
46693     default: /* Parent - writes to pipe */
46694         close(fildes[0]); /* Read end is unused */
46695         write(fildes[1], "Hello world\n", 12); /* Write data on pipe */
46696         close(fildes[1]); /* Child will see EOF */
46697         exit(EXIT_SUCCESS);
46698     }

```

APPLICATION USAGE

None.

RATIONALE

The wording carefully avoids using the verb “to open” in order to avoid any implication of use of *open()*; see also *write()*.

FUTURE DIRECTIONS

None.

SEE ALSO

fcntl(), *read()*, *write()*

XBD [<fcntl.h>](#), [<unistd.h>](#)

CHANGE HISTORY

First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The DESCRIPTION is updated to indicate that certain dispositions of *fildes[0]* and *fildes[1]* are unspecified.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/65 is applied, adding the example to the EXAMPLES section.

Issue 7

SD5-XSH-ERN-156 is applied, updating the DESCRIPTION to state the setting of the pipe's user ID and group ID.

Changes are made related to support for finegrained timestamps.

46722 NAME

46723 poll — input/output multiplexing

46724 SYNOPSIS

46725 #include <poll.h>

46726 int poll(struct pollfd *fds*[], nfds_t *nfds*, int *timeout*);

46727 DESCRIPTION

46728 The *poll()* function provides applications with a mechanism for multiplexing input/output over
 46729 a set of file descriptors. For each member of the array pointed to by *fds*, *poll()* shall examine the
 46730 given file descriptor for the event(s) specified in *events*. The number of **pollfd** structures in the
 46731 *fds* array is specified by *nfds*. The *poll()* function shall identify those file descriptors on which an
 46732 application can read or write data, or on which certain events have occurred.

46733 The *fds* argument specifies the file descriptors to be examined and the events of interest for each
 46734 file descriptor. It is a pointer to an array with one member for each open file descriptor of
 46735 interest. The array's members are **pollfd** structures within which *fd* specifies an open file
 46736 descriptor and *events* and *revents* are bitmasks constructed by OR'ing a combination of the
 46737 following event flags:

46738	POLLIN	Data other than high-priority data may be read without blocking.
46739	OB XSR	For STREAMS, this flag is set in <i>revents</i> even if the message is of zero length.
46740		This flag shall be equivalent to POLLRDNORM POLLRDBAND.
46741	POLLRDNORM	Normal data may be read without blocking.
46742	OB XSR	For STREAMS, data on priority band 0 may be read without blocking. This
46743		flag is set in <i>revents</i> even if the message is of zero length.
46744	POLLRDBAND	Priority data may be read without blocking.
46745	OB XSR	For STREAMS, data on priority bands greater than 0 may be read without
46746		blocking. This flag is set in <i>revents</i> even if the message is of zero length.
46747	POLLPRI	High-priority data may be read without blocking.
46748	OB XSR	For STREAMS, this flag is set in <i>revents</i> even if the message is of zero length.
46749	POLLOUT	Normal data may be written without blocking.
46750	OB XSR	For STREAMS, data on priority band 0 may be written without blocking.
46751	POLLWRNORM	Equivalent to POLLOUT.
46752	POLLWRBAND	Priority data may be written.
46753	OB XSR	For STREAMS, data on priority bands greater than 0 may be written without
46754		blocking. If any priority band has been written to on this STREAM, this event
46755		only examines bands that have been written to at least once.
46756	POLLERR	An error has occurred on the device or stream. This flag is only valid in the
46757		<i>revents</i> bitmask; it shall be ignored in the <i>events</i> member.
46758	POLLHUP	A device has been disconnected, or a pipe or FIFO has been closed by the last
46759		process that had it open for writing. Once set, the hangup state of a FIFO shall
46760		persist until some process opens the FIFO for writing or until all read-only file
46761		descriptors for the FIFO are closed. This event and POLLOUT are mutually-
46762		exclusive; a stream can never be writable if a hangup has occurred. However,
46763		this event and POLLIN, POLLRDNORM, POLLRDBAND, or POLLPRI are
46764		not mutually-exclusive. This flag is only valid in the <i>revents</i> bitmask; it shall be

46765		ignored in the <i>events</i> member.
46766	POLLNVAL	The specified <i>fd</i> value is invalid. This flag is only valid in the <i>revents</i> member;
46767		it shall ignored in the <i>events</i> member.
46768		The significance and semantics of normal, priority, and high-priority data are file and device-
46769		specific.
46770		If the value of <i>fd</i> is less than 0, <i>events</i> shall be ignored, and <i>revents</i> shall be set to 0 in that entry on
46771		return from <i>poll()</i> .
46772		In each pollfd structure, <i>poll()</i> shall clear the <i>revents</i> member, except that where the application
46773		requested a report on a condition by setting one of the bits of <i>events</i> listed above, <i>poll()</i> shall set
46774		the corresponding bit in <i>revents</i> if the requested condition is true. In addition, <i>poll()</i> shall set the
46775		POLLHUP, POLLERR, and POLLNVAL flag in <i>revents</i> if the condition is true, even if the
46776		application did not set the corresponding bit in <i>events</i> .
46777		If none of the defined events have occurred on any selected file descriptor, <i>poll()</i> shall wait at
46778		least <i>timeout</i> milliseconds for an event to occur on any of the selected file descriptors. If the value
46779		of <i>timeout</i> is 0, <i>poll()</i> shall return immediately. If the value of <i>timeout</i> is -1, <i>poll()</i> shall block until
46780		a requested event occurs or until the call is interrupted.
46781		Implementations may place limitations on the granularity of timeout intervals. If the requested
46782		timeout interval requires a finer granularity than the implementation supports, the actual
46783		timeout interval shall be rounded up to the next supported value.
46784		The <i>poll()</i> function shall not be affected by the O_NONBLOCK flag.
46785		The <i>poll()</i> function shall support regular files, terminal and pseudo-terminal devices, FIFOs,
46786	OB XSR	pipes, sockets and STREAMS-based files . The behavior of <i>poll()</i> on elements of <i>fds</i> that refer to
46787		other types of file is unspecified.
46788		Regular files shall always poll TRUE for reading and writing.
46789		A file descriptor for a socket that is listening for connections shall indicate that it is ready for
46790		reading, once connections are available. A file descriptor for a socket that is connecting
46791		asynchronously shall indicate that it is ready for writing, once a connection has been established.
46792	RETURN VALUE	
46793		Upon successful completion, <i>poll()</i> shall return a non-negative value. A positive value indicates
46794		the total number of file descriptors that have been selected (that is, file descriptors for which the
46795		<i>revents</i> member is non-zero). A value of 0 indicates that the call timed out and no file descriptors
46796		have been selected. Upon failure, <i>poll()</i> shall return -1 and set <i>errno</i> to indicate the error.
46797	ERRORS	
46798		The <i>poll()</i> function shall fail if:
46799	[EAGAIN]	The allocation of internal data structures failed but a subsequent request may
46800		succeed.
46801	[EINTR]	A signal was caught during <i>poll()</i> .
46802	OB XSR [EINVAL]	The <i>nfds</i> argument is greater than {OPEN_MAX}, or one of the <i>fd</i> members
46803		refers to a STREAM or multiplexer that is linked (directly or indirectly)
46804		downstream from a multiplexer.

46805 **EXAMPLES**46806 **Checking for Events on a Stream**

46807 The following example opens a pair of STREAMS devices and then waits for either one to
 46808 become writable. This example proceeds as follows:

- 46809 1. Sets the *timeout* parameter to 500 milliseconds.
 - 46810 2. Opens the STREAMS devices */dev/dev0* and */dev/dev1*, and then polls them, specifying
 46811 POLLOUT and POLLWRBAND as the events of interest.
- 46812 The STREAMS device names */dev/dev0* and */dev/dev1* are only examples of how
 46813 STREAMS devices can be named; STREAMS naming conventions may vary among
 46814 systems conforming to the POSIX.1-2008.
- 46815 3. Uses the *ret* variable to determine whether an event has occurred on either of the two
 46816 STREAMS. The *poll()* function is given 500 milliseconds to wait for an event to occur (if it
 46817 has not occurred prior to the *poll()* call).
 - 46818 4. Checks the returned value of *ret*. If a positive value is returned, one of the following can
 46819 be done:
 - 46820 a. Priority data can be written to the open STREAM on priority bands greater than 0,
 46821 because the POLLWRBAND event occurred on the open STREAM (*fds[0]* or *fds[1]*).
 - 46822 b. Data can be written to the open STREAM on priority-band 0, because the
 46823 POLLOUT event occurred on the open STREAM (*fds[0]* or *fds[1]*).
 - 46824 5. If the returned value is not a positive value, permission to write data to the open
 46825 STREAM (on any priority band) is denied.
 - 46826 6. If the POLLHUP event occurs on the open STREAM (*fds[0]* or *fds[1]*), the device on the
 46827 open STREAM has disconnected.

```

46828 #include <stropts.h>
46829 #include <poll.h>
46830 ...
46831 struct pollfd fds[2];
46832 int timeout_msecs = 500;
46833 int ret;
46834     int i;

46835 /* Open STREAMS device. */
46836 fds[0].fd = open("/dev/dev0", ...);
46837 fds[1].fd = open("/dev/dev1", ...);
46838 fds[0].events = POLLOUT | POLLWRBAND;
46839 fds[1].events = POLLOUT | POLLWRBAND;

46840 ret = poll(fds, 2, timeout_msecs);

46841 if (ret > 0) {
46842     /* An event on one of the fds has occurred. */
46843     for (i=0; i<2; i++) {
46844         if (fds[i].revents & POLLWRBAND) {
46845             /* Priority data may be written on device number i. */
46846             ...
46847         }
46848         if (fds[i].revents & POLLOUT) {

```



```

46849             /* Data may be written on device number i. */
46850         ...
46851     }
46852     if (fds[i].revents & POLLHUP) {
46853         /* A hangup has occurred on device number i. */
46854         ...
46855     }
46856 }
46857 }

```

APPLICATION USAGE

None.

RATIONALE

The POLLHUP event does not occur for FIFOs just because the FIFO is not open for writing. It only occurs when the FIFO is closed by the last writer and persists until some process opens the FIFO for writing or until all read-only file descriptors for the FIFO are closed.

FUTURE DIRECTIONS

None.

SEE ALSO

Section 2.6 (on page 499), *getmsg()*, *pselect()*, *putmsg()*, *read()*, *write()*

XBD <poll.h>, <stropts.h>

CHANGE HISTORY

First released in Issue 4, Version 2.

Issue 5

Moved from X/OPEN UNIX extension to BASE.

The description of POLLWRBAND is updated.

Issue 6

Text referring to sockets is added to the DESCRIPTION.

Functionality relating to the XSI STREAMS Option Group is marked.

The Open Group Corrigendum U055/3 is applied, updating the DESCRIPTION of POLLWRBAND.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/66 is applied, correcting the spacing in the EXAMPLES section.

Issue 7

Austin Group Interpretation 1003.1-2001 #209 is applied, clarifying the POLLHUP event.

The *poll()* function is moved from the XSI option to the Base.

Functionality relating to the XSI STREAMS option is marked obsolescent.

NAME

popen — initiate pipe streams to or from a process

SYNOPSIS

```
CX      #include <stdio.h>
FILE *popen(const char *command, const char *mode);
```

DESCRIPTION

The *popen()* function shall execute the command specified by the string *command*. It shall create a pipe between the calling program and the executed command, and shall return a pointer to a stream that can be used to either read from or write to the pipe.

The environment of the executed command shall be as if a child process were created within the *popen()* call using the *fork()* function, and the child invoked the *sh* utility using the call:

```
execl(shell path, "sh", "-c", command, (char *)0);
```

where *shell path* is an unspecified pathname for the *sh* utility.

The *popen()* function shall ensure that any streams from previous *popen()* calls that remain open in the parent process are closed in the new child process.

The *mode* argument to *popen()* is a string that specifies I/O mode:

1. If *mode* is *r*, when the child process is started, its file descriptor `STDOUT_FILENO` shall be the writable end of the pipe, and the file descriptor *fileno(stream)* in the calling process, where *stream* is the stream pointer returned by *popen()*, shall be the readable end of the pipe.
2. If *mode* is *w*, when the child process is started its file descriptor `STDIN_FILENO` shall be the readable end of the pipe, and the file descriptor *fileno(stream)* in the calling process, where *stream* is the stream pointer returned by *popen()*, shall be the writable end of the pipe.
3. If *mode* is any other value, the result is unspecified.

After *popen()*, both the parent and the child process shall be capable of executing independently before either terminates.

Pipe streams are byte-oriented.

RETURN VALUE

Upon successful completion, *popen()* shall return a pointer to an open stream that can be used to read or write to the pipe. Otherwise, it shall return a null pointer and may set *errno* to indicate the error.

ERRORS

The *popen()* function shall fail if:

[EMFILE] {STREAM_MAX} streams are currently open in the calling process.

The *popen()* function may fail if:

[EMFILE] {FOPEN_MAX} streams are currently open in the calling process.

[EINVAL] The *mode* argument is invalid.

The *popen()* function may also set *errno* values as described by *fork()* or *pipe()*.

EXAMPLES

Using popen() to Obtain a List of Files from the ls Utility

The following example demonstrates the use of *popen()* and *pclose()* to execute the command *ls* in order to obtain a list of files in the current directory:

```
#include <stdio.h>
...

FILE *fp;
int status;
char path[PATH_MAX];

fp = popen("ls *", "r");
if (fp == NULL)
    /* Handle error */;

while (fgets(path, PATH_MAX, fp) != NULL)
    printf("%s", path);

status = pclose(fp);
if (status == -1) {
    /* Error reported by pclose() */
    ...
} else {
    /* Use macros described under wait() to inspect 'status' in order
       to determine success/failure of command executed by popen() */
    ...
}
```

APPLICATION USAGE

Since open files are shared, a mode *r* command can be used as an input filter and a mode *w* command as an output filter.

Buffered reading before opening an input filter may leave the standard input of that filter mispositioned. Similar problems with an output filter may be prevented by careful buffer flushing; for example, with *fflush()*.

A stream opened by *popen()* should be closed by *pclose()*.

The behavior of *popen()* is specified for values of *mode* of *r* and *w*. Other modes such as *rb* and *wb* might be supported by specific implementations, but these would not be portable features. Note that historical implementations of *popen()* only check to see if the first character of *mode* is *r*. Thus, a *mode* of *robert the robot* would be treated as *mode r*, and a *mode* of *anything else* would be treated as *mode w*.

If the application calls *waitpid()* or *waitid()* with a *pid* argument greater than 0, and it still has a stream that was called with *popen()* open, it must ensure that *pid* does not refer to the process started by *popen()*.

To determine whether or not the environment specified in the Shell and Utilities volume of POSIX.1-2008 is present, use the function call:

```
sysconf(_SC_2_VERSION)
```

(See *sysconf()*).

RATIONALE

The *popen()* function should not be used by programs that have set user (or group) ID privileges. The *fork()* and *exec* family of functions (except *execlp()* and *execvp()*), should be used instead. This prevents any unforeseen manipulation of the environment of the user that could cause execution of commands not anticipated by the calling program.

If the original and *popen()*ed processes both intend to read or write or read and write a common file, and either will be using FILE-type C functions (*fread()*, *fwrite()*, and so on), the rules for sharing file handles must be observed (see [Section 2.5.1](#), on page 496).

FUTURE DIRECTIONS

None.

SEE ALSO

[Section 2.5](#) (on page 495), *fork()*, *pclose()*, *pipe()*, *sysconf()*, *system()*, *wait()*, *waitid()*

XBD [<stdio.h>](#)

XCU *sh*

CHANGE HISTORY

First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 5

A statement is added to the DESCRIPTION indicating that pipe streams are byte-oriented.

Issue 6

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The optional [EMFILE] error condition is added.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/67 is applied, adding the example to the EXAMPLES section.

Issue 7

Austin Group Interpretation 1003.1-2001 #029 is applied, clarifying the values for *mode* in the DESCRIPTION.

SD5-XSH-ERN-149 is applied, changing the {STREAM_MAX} [EMFILE] error condition from a “may fail” to a “shall fail”.

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0432 [14] is applied.

+

NAME

posix_fadvise — file advisory information (**ADVANCED REALTIME**)

SYNOPSIS

```
#include <fcntl.h>

int posix_fadvise(int fd, off_t offset, off_t len, int advice);
```

DESCRIPTION

The *posix_fadvise()* function shall advise the implementation on the expected behavior of the application with respect to the data in the file associated with the open file descriptor, *fd*, starting at *offset* and continuing for *len* bytes. The specified range need not currently exist in the file. If *len* is zero, all data following *offset* is specified. The implementation may use this information to optimize handling of the specified data. The *posix_fadvise()* function shall have no effect on the semantics of other operations on the specified data, although it may affect the performance of other operations.

The advice to be applied to the data is specified by the *advice* parameter and may be one of the following values:

POSIX_FADV_NORMAL

Specifies that the application has no advice to give on its behavior with respect to the specified data. It is the default characteristic if no advice is given for an open file.

POSIX_FADV_SEQUENTIAL

Specifies that the application expects to access the specified data sequentially from lower offsets to higher offsets.

POSIX_FADV_RANDOM

Specifies that the application expects to access the specified data in a random order.

POSIX_FADV_WILLNEED

Specifies that the application expects to access the specified data in the near future.

POSIX_FADV_DONTNEED

Specifies that the application expects that it will not access the specified data in the near future.

POSIX_FADV_NOREUSE

Specifies that the application expects to access the specified data once and then not reuse it thereafter.

These values are defined in **<fcntl.h>**.

RETURN VALUE

Upon successful completion, *posix_fadvise()* shall return zero; otherwise, an error number shall be returned to indicate the error.

ERRORS

The *posix_fadvise()* function shall fail if:

[EBADF]	The <i>fd</i> argument is not a valid file descriptor.
[EINVAL]	The value of <i>advice</i> is invalid, or the value of <i>len</i> is less than zero.
[ESPIPE]	The <i>fd</i> argument is associated with a pipe or FIFO.

47036 EXAMPLES

47037 None.

47038 APPLICATION USAGE

47039 The *posix_fadvise()* function is part of the Advisory Information option and need not be
47040 provided on all implementations.

47041 RATIONALE

47042 None.

47043 FUTURE DIRECTIONS

47044 None.

47045 SEE ALSO

47046 *posix_madvise()*

47047 XBD **<fcntl.h>**

47048 CHANGE HISTORY

47049 First released in Issue 6. Derived from IEEE Std 1003.1d-1999.

47050 In the SYNOPSIS, the inclusion of **<sys/types.h>** is no longer required.

47051 IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/68 is applied, changing the function
47052 prototype in the SYNOPSIS section. The previous prototype was not large file-aware, and the
47053 standard developers felt it acceptable to make this change before implementations of this
47054 function become widespread.

47055 Issue 7

47056 Austin Group Interpretation 1003.1-2001 #024 is applied, changing the definition of the
47057 [EINVAL] error.

47058 **NAME**47059 posix_fallocate — file space control (**ADVANCED REALTIME**)47060 **SYNOPSIS**

```
47061 ADV    #include <fcntl.h>
47062        int posix_fallocate(int fd, off_t offset, off_t len);
```

47063 **DESCRIPTION**

47064 The *posix_fallocate()* function shall ensure that any required storage for regular file data starting
 47065 at *offset* and continuing for *len* bytes is allocated on the file system storage media. If
 47066 *posix_fallocate()* returns successfully, subsequent writes to the specified file data shall not fail due
 47067 to the lack of free space on the file system storage media.

47068 If the *offset+len* is beyond the current file size, then *posix_fallocate()* shall adjust the file size to
 47069 *offset+len*. Otherwise, the file size shall not be changed.

47070 It is implementation-defined whether a previous *posix_fadvise()* call influences allocation
 47071 strategy.

47072 Space allocated via *posix_fallocate()* shall be freed by a successful call to *creat()* or *open()* that
 47073 truncates the size of the file. Space allocated via *posix_fallocate()* may be freed by a successful call
 47074 to *ftruncate()* that reduces the file size to a size smaller than *offset+len*.

47075 **RETURN VALUE**

47076 Upon successful completion, *posix_fallocate()* shall return zero; otherwise, an error number shall
 47077 be returned to indicate the error.

47078 **ERRORS**

47079 The *posix_fallocate()* function shall fail if:

47080	[EBADF]	The <i>fd</i> argument is not a valid file descriptor.
47081	[EBADF]	The <i>fd</i> argument references a file that was opened without write permission.
47082	[EFBIG]	The value of <i>offset+len</i> is greater than the maximum file size.
47083	[EINTR]	A signal was caught during execution.
47084	[EINVAL]	The <i>len</i> argument is less than zero, or the <i>offset</i> argument is less than zero, or the underlying file system does not support this operation.
47085		
47086	[EIO]	An I/O error occurred while reading from or writing to a file system.
47087	[ENODEV]	The <i>fd</i> argument does not refer to a regular file.
47088	[ENOSPC]	There is insufficient free space remaining on the file system storage media.
47089	[ESPIPE]	The <i>fd</i> argument is associated with a pipe or FIFO.
47090		The <i>posix_fallocate()</i> function may fail if:
47091	[EINVAL]	The <i>len</i> argument is zero.

47092 EXAMPLES

47093 None.

47094 APPLICATION USAGE

47095 The *posix_fallocate()* function is part of the Advisory Information option and need not be
47096 provided on all implementations.

47097 RATIONALE

47098 None.

47099 FUTURE DIRECTIONS

47100 None.

47101 SEE ALSO

47102 *creat()*, *ftruncate()*, *open()*, *unlink()*

47103 XBD <fcntl.h>

47104 CHANGE HISTORY

47105 First released in Issue 6. Derived from IEEE Std 1003.1d-1999.

47106 In the SYNOPSIS, the inclusion of <sys/types.h> is no longer required.

47107 IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/69 is applied, changing the function
47108 prototype in the SYNOPSIS section. The previous prototype was not large file-aware, and the
47109 standard developers felt it acceptable to make this change before implementations of this
47110 function become widespread.

47111 Issue 7

47112 Austin Group Interpretations 1003.1-2001 #022, #024, and #162 are applied, changing the
47113 definition of the [EINVAL] error.

NAME

posix_madvise — memory advisory information and alignment control (**ADVANCED REALTIME**)

SYNOPSIS

```
#include <sys/mman.h>

int posix_madvise(void *addr, size_t len, int advice);
```

DESCRIPTION

The *posix_madvise()* function shall advise the implementation on the expected behavior of the application with respect to the data in the memory starting at address *addr*, and continuing for *len* bytes. The implementation may use this information to optimize handling of the specified data. The *posix_madvise()* function shall have no effect on the semantics of access to memory in the specified range, although it may affect the performance of access.

The implementation may require that *addr* be a multiple of the page size, which is the value returned by *sysconf()* when the name value *_SC_PAGESIZE* is used.

The advice to be applied to the memory range is specified by the *advice* parameter and may be one of the following values:

POSIX_MADV_NORMAL

Specifies that the application has no advice to give on its behavior with respect to the specified range. It is the default characteristic if no advice is given for a range of memory.

POSIX_MADV_SEQUENTIAL

Specifies that the application expects to access the specified range sequentially from lower addresses to higher addresses.

POSIX_MADV_RANDOM

Specifies that the application expects to access the specified range in a random order.

POSIX_MADV_WILLNEED

Specifies that the application expects to access the specified range in the near future.

POSIX_MADV_DONTNEED

Specifies that the application expects that it will not access the specified range in the near future.

These values are defined in the **<sys/mman.h>** header.

RETURN VALUE

Upon successful completion, *posix_madvise()* shall return zero; otherwise, an error number shall be returned to indicate the error.

ERRORS

The *posix_madvise()* function shall fail if:

[EINVAL] The value of *advice* is invalid.

[ENOMEM] Addresses in the range starting at *addr* and continuing for *len* bytes are partly or completely outside the range allowed for the address space of the calling process.

47153 The *posix_madvise()* function may fail if:

47154 [EINVAL] The value of *addr* is not a multiple of the value returned by *sysconf()* when the
47155 name value *_SC_PAGESIZE* is used.

47156 [EINVAL] The value of *len* is zero.

47157 **EXAMPLES**

47158 None.

47159 **APPLICATION USAGE**

47160 The *posix_madvise()* function is part of the Advisory Information option and need not be
47161 provided on all implementations.

47162 **RATIONALE**

47163 None.

47164 **FUTURE DIRECTIONS**

47165 None.

47166 **SEE ALSO**

47167 *mmap()*, *posix_fadvise()*, *sysconf()*

47168 XBD <*sys/mman.h*>

47169 **CHANGE HISTORY**

47170 First released in Issue 6. Derived from IEEE Std 1003.1d-1999.

47171 IEEE PASC Interpretation 1003.1 #102 is applied.

NAME

posix_mem_offset — find offset and length of a mapped typed memory block (ADVANCED REALTIME)

SYNOPSIS

```
#include <sys/mman.h>

int posix_mem_offset(const void *restrict addr, size_t len,
    off_t *restrict off, size_t *restrict contig_len,
    int *restrict fildes);
```

DESCRIPTION

The *posix_mem_offset()* function shall return in the variable pointed to by *off* a value that identifies the offset (or location), within a memory object, of the memory block currently mapped at *addr*. The function shall return in the variable pointed to by *fildes*, the descriptor used (via *mmap()*) to establish the mapping which contains *addr*. If that descriptor was closed since the mapping was established, the returned value of *fildes* shall be -1. The *len* argument specifies the length of the block of the memory object the user wishes the offset for; upon return, the value pointed to by *contig_len* shall equal either *len*, or the length of the largest contiguous block of the memory object that is currently mapped to the calling process starting at *addr*, whichever is smaller.

If the memory object mapped at *addr* is a typed memory object, then if the *off* and *contig_len* values obtained by calling *posix_mem_offset()* are used in a call to *mmap()* with a file descriptor that refers to the same memory pool as *fildes* (either through the same port or through a different port), and that was opened with neither the POSIX_TYPED_MEM_ALLOCATE nor the POSIX_TYPED_MEM_ALLOCATE_CONTIG flag, the typed memory area that is mapped shall be exactly the same area that was mapped at *addr* in the address space of the process that called *posix_mem_offset()*.

If the memory object specified by *fildes* is not a typed memory object, then the behavior of this function is implementation-defined.

RETURN VALUE

Upon successful completion, the *posix_mem_offset()* function shall return zero; otherwise, the corresponding error status value shall be returned.

ERRORS

The *posix_mem_offset()* function shall fail if:

[EACCES] The process has not mapped a memory object supported by this function at the given address *addr*.

This function shall not return an error code of [EINTR].

EXAMPLES

None.

APPLICATION USAGE

None.

RATIONALE

None.

47213 **FUTURE DIRECTIONS**

47214 None.

47215 **SEE ALSO**47216 [*mmap\(\)*](#), [*posix_typed_mem_open\(\)*](#)47217 XBD [**<sys/mman.h>**](#)47218 **CHANGE HISTORY**

47219 First released in Issue 6. Derived from IEEE Std 1003.1j-2000.

47220 **NAME**

47221 posix_memalign — aligned memory allocation (ADVANCED REALTIME)

47222 **SYNOPSIS**

```
47223 ADV    #include <stdlib.h>
47224        int posix_memalign(void **memptr, size_t alignment, size_t size);
```

47225 **DESCRIPTION**

47226 The *posix_memalign()* function shall allocate *size* bytes aligned on a boundary specified by
 47227 *alignment*, and shall return a pointer to the allocated memory in *memptr*. The value of *alignment*
 47228 shall be a power of two multiple of *sizeof(void *)*.

47229 Upon successful completion, the value pointed to by *memptr* shall be a multiple of *alignment*.

47230 If the size of the space requested is 0, the behavior is implementation-defined; the value returned
 47231 in *memptr* shall be either a null pointer or a unique pointer.

47232 CX The *free()* function shall deallocate memory that has previously been allocated by
 47233 *posix_memalign()*.

47234 **RETURN VALUE**

47235 Upon successful completion, *posix_memalign()* shall return zero; otherwise, an error number
 47236 shall be returned to indicate the error.

47237 **ERRORS**

47238 The *posix_memalign()* function shall fail if:

47239 [EINVAL] The value of the alignment parameter is not a power of two multiple of
 47240 *sizeof(void *)*.

47241 [ENOMEM] There is insufficient memory available with the requested alignment.

47242 **EXAMPLES**

47243 None.

47244 **APPLICATION USAGE**

47245 The *posix_memalign()* function is part of the Advisory Information option and need not be
 47246 provided on all implementations.

47247 **RATIONALE**

47248 None.

47249 **FUTURE DIRECTIONS**

47250 None.

47251 **SEE ALSO**

47252 *free()*, *malloc()*

47253 XBD <stdlib.h>

47254 **CHANGE HISTORY**

47255 First released in Issue 6. Derived from IEEE Std 1003.1d-1999.

47256 In the SYNOPSIS, the inclusion of <sys/types.h> is no longer required.

47257 **Issue 7**

47258 Austin Group Interpretation 1003.1-2001 #058 is applied, clarifying the value of the *alignment*
 47259 argument in the DESCRIPTION.

47260
47261

Austin Group Interpretation 1003.1-2001 #152 is applied, clarifying the behavior when the size of the space requested is 0.

NAME

posix_openpt — open a pseudo-terminal device

SYNOPSIS

```
#include <stdlib.h>
#include <fcntl.h>

int posix_openpt(int oflag);
```

DESCRIPTION

The *posix_openpt()* function shall establish a connection between a master device for a pseudo-terminal and a file descriptor. The file descriptor is used by other I/O functions that refer to that pseudo-terminal.

The file status flags and file access modes of the open file description shall be set according to the value of *oflag*.

Values for *oflag* are constructed by a bitwise-inclusive OR of flags from the following list, defined in *<fcntl.h>*:

O_RDWR	Open for reading and writing.
O_NOCTTY	If set <i>posix_openpt()</i> shall not cause the terminal device to become the controlling terminal for the process.

The behavior of other values for the *oflag* argument is unspecified.

RETURN VALUE

Upon successful completion, the *posix_openpt()* function shall open a master pseudo-terminal device and return a non-negative integer representing the lowest numbered unused file descriptor. Otherwise, -1 shall be returned and *errno* set to indicate the error.

ERRORS

The *posix_openpt()* function shall fail if:

[EMFILE]	All file descriptors available to the process are currently open.
[ENFILE]	The maximum allowable number of files is currently open in the system.

The *posix_openpt()* function may fail if:

[EINVAL]	The value of <i>oflag</i> is not valid.
[EAGAIN]	Out of pseudo-terminal resources.

[ENOSR]	Out of STREAMS resources.

EXAMPLES

Opening a Pseudo-Terminal and Returning the Name of the Slave Device and a File Descriptor

```
#include <fcntl.h>
#include <stdio.h>

int masterfd, slavefd;
char *slavedevice;

masterfd = posix_openpt(O_RDWR|O_NOCTTY);

if (masterfd == -1
    || grantpt (masterfd) == -1
```

```

47302         || unlockpt (masterfd) == -1
47303         || (slavedevice = ptsname (masterfd)) == NULL)
47304         return -1;

47305     printf("slave device is: %s\n", slavedevice);

47306     slavefd = open(slavedevice, O_RDWR|O_NOCTTY);
47307     if (slavefd < 0)
47308         return -1;

```

APPLICATION USAGE

This function is a method for portably obtaining a file descriptor of a master terminal device for a pseudo-terminal. The *grantpt()* and *ptsname()* functions can be used to manipulate mode and ownership permissions, and to obtain the name of the slave device, respectively.

RATIONALE

The standard developers considered the matter of adding a special device for cloning master pseudo-terminals: the */dev/ptmx* device. However, consensus could not be reached, and it was felt that adding a new function would permit other implementations. The *posix_openpt()* function is designed to complement the *grantpt()*, *ptsname()*, and *unlockpt()* functions.

On implementations supporting the */dev/ptmx* clone device, opening the master device of a pseudo-terminal is simply:

```

47320     mfdp = open("/dev/ptmx", oflag );
47321     if (mfdp < 0)
47322         return -1;

```

FUTURE DIRECTIONS

None.

SEE ALSO

grantpt(), *open()*, *ptsname()*, *unlockpt()*

XBD [<fcntl.h>](#), [<stdlib.h>](#)

CHANGE HISTORY

First released in Issue 6.

Issue 7

SD5-XBD-ERN-4 is applied, changing the definition of the [EMFILE] error.

SD5-XSH-ERN-51 is applied, correcting an error in the EXAMPLES section.

NAME

posix_spawn, posix_spawnp — spawn a process (**ADVANCED REALTIME**)

SYNOPSIS

```
#include <spawn.h>

int posix_spawn(pid_t *restrict pid, const char *restrict path,
               const posix_spawn_file_actions_t *file_actions,
               const posix_spawnattr_t *restrict attrp,
               char *const argv[restrict], char *const envp[restrict]);
int posix_spawnp(pid_t *restrict pid, const char *restrict file,
               const posix_spawn_file_actions_t *file_actions,
               const posix_spawnattr_t *restrict attrp,
               char *const argv[restrict], char *const envp[restrict]);
```

DESCRIPTION

The *posix_spawn()* and *posix_spawnp()* functions shall create a new process (child process) from the specified process image. The new process image shall be constructed from a regular executable file called the new process image file.

When a C program is executed as the result of this call, it shall be entered as a C-language function call as follows:

```
int main(int argc, char *argv[]);
```

where *argc* is the argument count and *argv* is an array of character pointers to the arguments themselves. In addition, the following variable:

```
extern char **environ;
```

shall be initialized as a pointer to an array of character pointers to the environment strings.

The argument *argv* is an array of character pointers to null-terminated strings. The last member of this array shall be a null pointer and is not counted in *argc*. These strings constitute the argument list available to the new process image. The value in *argv*[0] should point to a filename string that is associated with the process image being started by the *posix_spawn()* or *posix_spawnp()* function.

The argument *envp* is an array of character pointers to null-terminated strings. These strings constitute the environment for the new process image. The environment array is terminated by a null pointer.

The number of bytes available for the combined argument and environment lists of the child process is {ARG_MAX}. The implementation shall specify in the system documentation (see XBD Chapter 2, on page 15) whether any list overhead, such as length words, null terminators, pointers, or alignment bytes, is included in this total.

The *path* argument to *posix_spawn()* is a pathname that identifies the new process image file to execute.

The *file* parameter to *posix_spawnp()* shall be used to construct a pathname that identifies the new process image file. If the *file* parameter contains a <slash> character, the *file* parameter shall be used as the pathname for the new process image file. Otherwise, the path prefix for this file shall be obtained by a search of the directories passed as the environment variable *PATH* (see XBD Chapter 8, on page 173). If this environment variable is not defined, the results of the search are implementation-defined.

If *file_actions* is a null pointer, then file descriptors open in the calling process shall remain open

in the child process, except for those whose close-on-exec flag `FD_CLOEXEC` is set (see `fcntl()`). For those file descriptors that remain open, all attributes of the corresponding open file descriptions, including file locks (see `fcntl()`), shall remain unchanged.

If `file_actions` is not NULL, then the file descriptors open in the child process shall be those open in the calling process as modified by the spawn file actions object pointed to by `file_actions` and the `FD_CLOEXEC` flag of each remaining open file descriptor after the spawn file actions have been processed. The effective order of processing the spawn file actions shall be:

1. The set of open file descriptors for the child process shall initially be the same set as is open for the calling process. All attributes of the corresponding open file descriptions, including file locks (see `fcntl()`), shall remain unchanged.
2. The signal mask, signal default actions, and the effective user and group IDs for the child process shall be changed as specified in the attributes object referenced by `attrp`.
3. The file actions specified by the spawn file actions object shall be performed in the order in which they were added to the spawn file actions object.
4. Any file descriptor that has its `FD_CLOEXEC` flag set (see `fcntl()`) shall be closed.

If file descriptor 0, 1, or 2 would otherwise be closed in the new process image created by `posix_spawn()` or `posix_spawnnp()`, implementations may open an unspecified file for the file descriptor in the new process image. If a standard utility or a conforming application is executed with file descriptor 0 not open for reading or with file descriptor 1 or 2 not open for writing, the environment in which the utility or application is executed shall be deemed non-conforming, and consequently the utility or application might not behave as described in this standard.

The `posix_spawnattr_t` spawn attributes object type is defined in `<spawn.h>`. It shall contain at least the attributes defined below.

If the `POSIX_SPAWN_SETPGROUP` flag is set in the `spawn-flgs` attribute of the object referenced by `attrp`, and the `spawn-pgroup` attribute of the same object is non-zero, then the child's process group shall be as specified in the `spawn-pgroup` attribute of the object referenced by `attrp`.

As a special case, if the `POSIX_SPAWN_SETPGROUP` flag is set in the `spawn-flgs` attribute of the object referenced by `attrp`, and the `spawn-pgroup` attribute of the same object is set to zero, then the child shall be in a new process group with a process group ID equal to its process ID.

If the `POSIX_SPAWN_SETPGROUP` flag is not set in the `spawn-flgs` attribute of the object referenced by `attrp`, the new child process shall inherit the parent's process group.

PS If the `POSIX_SPAWN_SETSCHEDPARAM` flag is set in the `spawn-flgs` attribute of the object referenced by `attrp`, but `POSIX_SPAWN_SETSCHEDULER` is not set, the new process image shall initially have the scheduling policy of the calling process with the scheduling parameters specified in the `spawn-schedparam` attribute of the object referenced by `attrp`.

If the `POSIX_SPAWN_SETSCHEDULER` flag is set in the `spawn-flgs` attribute of the object referenced by `attrp` (regardless of the setting of the `POSIX_SPAWN_SETSCHEDPARAM` flag), the new process image shall initially have the scheduling policy specified in the `spawn-schedpolicy` attribute of the object referenced by `attrp` and the scheduling parameters specified in the `spawn-schedparam` attribute of the same object.

The `POSIX_SPAWN_RESETPID` flag in the `spawn-flgs` attribute of the object referenced by `attrp` governs the effective user ID of the child process. If this flag is not set, the child process shall inherit the effective user ID of the parent process. If this flag is set, the effective user ID of the child process shall be reset to the parent's real user ID. In either case, if the set-user-ID mode bit of the new process image file is set, the effective user ID of the child process shall become that

file's owner ID before the new process image begins execution.

The POSIX_SPAWN_RESETPID flag in the *spawn_flags* attribute of the object referenced by *attrp* also governs the effective group ID of the child process. If this flag is not set, the child process shall inherit the effective group ID of the parent process. If this flag is set, the effective group ID of the child process shall be reset to the parent's real group ID. In either case, if the set-group-ID mode bit of the new process image file is set, the effective group ID of the child process shall become that file's group ID before the new process image begins execution.

If the POSIX_SPAWN_SETSIGMASK flag is set in the *spawn_flags* attribute of the object referenced by *attrp*, the child process shall initially have the signal mask specified in the *spawn-sigmask* attribute of the object referenced by *attrp*.

If the POSIX_SPAWN_SETSIGDEF flag is set in the *spawn_flags* attribute of the object referenced by *attrp*, the signals specified in the *spawn-sigdefault* attribute of the same object shall be set to their default actions in the child process. Signals set to the default action in the parent process shall be set to the default action in the child process.

Signals set to be caught by the calling process shall be set to the default action in the child process.

Except for SIGCHLD, signals set to be ignored by the calling process image shall be set to be ignored by the child process, unless otherwise specified by the POSIX_SPAWN_SETSIGDEF flag being set in the *spawn_flags* attribute of the object referenced by *attrp* and the signals being indicated in the *spawn-sigdefault* attribute of the object referenced by *attrp*.

If the SIGCHLD signal is set to be ignored by the calling process, it is unspecified whether the SIGCHLD signal is set to be ignored or to the default action in the child process, unless otherwise specified by the POSIX_SPAWN_SETSIGDEF flag being set in the *spawn_flags* attribute of the object referenced by *attrp* and the SIGCHLD signal being indicated in the *spawn-sigdefault* attribute of the object referenced by *attrp*.

If the value of the *attrp* pointer is NULL, then the default values are used.

All process attributes, other than those influenced by the attributes set in the object referenced by *attrp* as specified above or by the file descriptor manipulations specified in *file_actions*, shall appear in the new process image as though *fork()* had been called to create a child process and then a member of the *exec* family of functions had been called by the child process to execute the new process image.

It is implementation-defined whether the fork handlers are run when *posix_spawn()* or *posix_spawnnp()* is called.

RETURN VALUE

Upon successful completion, *posix_spawn()* and *posix_spawnnp()* shall return the process ID of the child process to the parent process, in the variable pointed to by a non-NULL *pid* argument, and shall return zero as the function return value. Otherwise, no child process shall be created, the value stored into the variable pointed to by a non-NULL *pid* is unspecified, and an error number shall be returned as the function return value to indicate the error. If the *pid* argument is a null pointer, the process ID of the child is not returned to the caller.

ERRORS

These functions may fail if:

[EINVAL] The value specified by *file_actions* or *attrp* is invalid.

If this error occurs after the calling process successfully returns from the *posix_spawn()* or *posix_spawnnp()* function, the child process may exit with exit status 127.

If *posix_spawn()* or *posix_spawnnp()* fail for any of the reasons that would cause *fork()* or one of the *exec* family of functions to fail, an error value shall be returned as described by *fork()* and *exec*, respectively (or, if the error occurs after the calling process successfully returns, the child process shall exit with exit status 127).

If POSIX_SPAWN_SETPGROUP is set in the *spawn_flags* attribute of the object referenced by *attrp*, and *posix_spawn()* or *posix_spawnnp()* fails while changing the child's process group, an error value shall be returned as described by *setpgid()* (or, if the error occurs after the calling process successfully returns, the child process shall exit with exit status 127).

PS If POSIX_SPAWN_SETSCHEDPARAM is set and POSIX_SPAWN_SETSCHEDULER is not set in the *spawn_flags* attribute of the object referenced by *attrp*, then if *posix_spawn()* or *posix_spawnnp()* fails for any of the reasons that would cause *sched_setparam()* to fail, an error value shall be returned as described by *sched_setparam()* (or, if the error occurs after the calling process successfully returns, the child process shall exit with exit status 127).

If POSIX_SPAWN_SETSCHEDULER is set in the *spawn_flags* attribute of the object referenced by *attrp*, and if *posix_spawn()* or *posix_spawnnp()* fails for any of the reasons that would cause *sched_setscheduler()* to fail, an error value shall be returned as described by *sched_setscheduler()* (or, if the error occurs after the calling process successfully returns, the child process shall exit with exit status 127).

If the *file_actions* argument is not NULL, and specifies any *close*, *dup2*, or *open* actions to be performed, and if *posix_spawn()* or *posix_spawnnp()* fails for any of the reasons that would cause *close()*, *dup2()*, or *open()* to fail, an error value shall be returned as described by *close()*, *dup2()*, and *open()*, respectively (or, if the error occurs after the calling process successfully returns, the child process shall exit with exit status 127). An open file action may, by itself, result in any of the errors described by *close()* or *dup2()*, in addition to those described by *open()*.

EXAMPLES

None.

APPLICATION USAGE

These functions are part of the Spawn option and need not be provided on all implementations.

See also the APPLICATION USAGE section for *exec*.

+

RATIONALE

The *posix_spawn()* function and its close relation *posix_spawnnp()* have been introduced to overcome the following perceived difficulties with *fork()*: the *fork()* function is difficult or impossible to implement without swapping or dynamic address translation.

- Swapping is generally too slow for a realtime environment.
- Dynamic address translation is not available everywhere that POSIX might be useful.
- Processes are too useful to simply option out of POSIX whenever it must run without address translation or other MMU services.

Thus, POSIX needs process creation and file execution primitives that can be efficiently implemented without address translation or other MMU services.

The *posix_spawn()* function is implementable as a library routine, but both *posix_spawn()* and *posix_spawnnp()* are designed as kernel operations. Also, although they may be an efficient replacement for many *fork()/exec* pairs, their goal is to provide useful process creation primitives for systems that have difficulty with *fork()*, not to provide drop-in replacements for *fork()/exec*.

This view of the role of *posix_spawn()* and *posix_spawnnp()* influenced the design of their API. It

does not attempt to provide the full functionality of *fork()/exec* in which arbitrary user-specified operations of any sort are permitted between the creation of the child process and the execution of the new process image; any attempt to reach that level would need to provide a programming language as parameters. Instead, *posix_spawn()* and *posix_spawnnp()* are process creation primitives like the *Start_Process* and *Start_Process_Search* Ada language bindings package *POSIX_Process_Primitives* and also like those in many operating systems that are not UNIX systems, but with some POSIX-specific additions.

To achieve its coverage goals, *posix_spawn()* and *posix_spawnnp()* have control of six types of inheritance: file descriptors, process group ID, user and group ID, signal mask, scheduling, and whether each signal ignored in the parent will remain ignored in the child, or be reset to its default action in the child.

Control of file descriptors is required to allow an independently written child process image to access data streams opened by and even generated or read by the parent process without being specifically coded to know which parent files and file descriptors are to be used. Control of the process group ID is required to control how the job control of the child process relates to that of the parent.

Control of the signal mask and signal defaulting is sufficient to support the implementation of *system()*. Although support for *system()* is not explicitly one of the goals for *posix_spawn()* and *posix_spawnnp()*, it is covered under the “at least 50%” coverage goal.

The intention is that the normal file descriptor inheritance across *fork()*, the subsequent effect of the specified spawn file actions, and the normal file descriptor inheritance across one of the *exec* family of functions should fully specify open file inheritance. The implementation need make no decisions regarding the set of open file descriptors when the child process image begins execution, those decisions having already been made by the caller and expressed as the set of open file descriptors and their *FD_CLOEXEC* flags at the time of the call and the spawn file actions object specified in the call. We have been assured that in cases where the POSIX *Start_Process* Ada primitives have been implemented in a library, this method of controlling file descriptor inheritance may be implemented very easily.

We can identify several problems with *posix_spawn()* and *posix_spawnnp()*, but there does not appear to be a solution that introduces fewer problems. Environment modification for child process attributes not specifiable via the *attrp* or *file_actions* arguments must be done in the parent process, and since the parent generally wants to save its context, it is more costly than similar functionality with *fork()/exec*. It is also complicated to modify the environment of a multi-threaded process temporarily, since all threads must agree when it is safe for the environment to be changed. However, this cost is only borne by those invocations of *posix_spawn()* and *posix_spawnnp()* that use the additional functionality. Since extensive modifications are not the usual case, and are particularly unlikely in time-critical code, keeping much of the environment control out of *posix_spawn()* and *posix_spawnnp()* is appropriate design.

The *posix_spawn()* and *posix_spawnnp()* functions do not have all the power of *fork()/exec*. This is to be expected. The *fork()* function is a wonderfully powerful operation. We do not expect to duplicate its functionality in a simple, fast function with no special hardware requirements. It is worth noting that *posix_spawn()* and *posix_spawnnp()* are very similar to the process creation operations on many operating systems that are not UNIX systems.

Requirements

The requirements for *posix_spawn()* and *posix_spawnnp()* are:

- They must be implementable without an MMU or unusual hardware.
- They must be compatible with existing POSIX standards.

Additional goals are:

- They should be efficiently implementable.
- They should be able to replace at least 50% of typical executions of *fork()*.
- A system with *posix_spawn()* and *posix_spawnnp()* and without *fork()* should be useful, at least for realtime applications.
- A system with *fork()* and the *exec* family should be able to implement *posix_spawn()* and *posix_spawnnp()* as library routines.

Two-Syntax

POSIX *exec* has several calling sequences with approximately the same functionality. These appear to be required for compatibility with existing practice. Since the existing practice for the *posix_spawn*()* functions is otherwise substantially unlike POSIX, we feel that simplicity outweighs compatibility. There are, therefore, only two names for the *posix_spawn*()* functions.

The parameter list does not differ between *posix_spawn()* and *posix_spawnnp()*; *posix_spawnnp()* interprets the second parameter more elaborately than *posix_spawn()*.

Compatibility with POSIX.5 (Ada)

The *Start_Process* and *Start_Process_Search* procedures from the *POSIX_Process_Primitives* package from the Ada language binding to POSIX.1 encapsulate *fork()* and *exec* functionality in a manner similar to that of *posix_spawn()* and *posix_spawnnp()*. Originally, in keeping with our simplicity goal, the standard developers had limited the capabilities of *posix_spawn()* and *posix_spawnnp()* to a subset of the capabilities of *Start_Process* and *Start_Process_Search*; certain non-default capabilities were not supported. However, based on suggestions by the ballot group to improve file descriptor mapping or drop it, and on the advice of an Ada Language Bindings working group member, the standard developers decided that *posix_spawn()* and *posix_spawnnp()* should be sufficiently powerful to implement *Start_Process* and *Start_Process_Search*. The rationale is that if the Ada language binding to such a primitive had already been approved as an IEEE standard, there can be little justification for not approving the functionally-equivalent parts of a C binding. The only three capabilities provided by *posix_spawn()* and *posix_spawnnp()* that are not provided by *Start_Process* and *Start_Process_Search* are optionally specifying the child's process group ID, the set of signals to be reset to default signal handling in the child process, and the child's scheduling policy and parameters.

For the Ada language binding for *Start_Process* to be implemented with *posix_spawn()*, that binding would need to explicitly pass an empty signal mask and the parent's environment to *posix_spawn()* whenever the caller of *Start_Process* allowed these arguments to default, since *posix_spawn()* does not provide such defaults. The ability of *Start_Process* to mask user-specified signals during its execution is functionally unique to the Ada language binding and must be dealt with in the binding separately from the call to *posix_spawn()*.

Process Group

The process group inheritance field can be used to join the child process with an existing process group. By assigning a value of zero to the *spawn-pgroup* attribute of the object referenced by *attrp*, the *setpgid()* mechanism will place the child process in a new process group.

Threads

Without the *posix_spawn()* and *posix_spawnnp()* functions, systems without address translation can still use threads to give an abstraction of concurrency. In many cases, thread creation suffices, but it is not always a good substitute. The *posix_spawn()* and *posix_spawnnp()* functions are considerably “heavier” than thread creation. Processes have several important attributes that threads do not. Even without address translation, a process may have base-and-bound memory protection. Each process has a process environment including security attributes and file capabilities, and powerful scheduling attributes. Processes abstract the behavior of non-uniform-memory-architecture multi-processors better than threads, and they are more convenient to use for activities that are not closely linked.

The *posix_spawn()* and *posix_spawnnp()* functions may not bring support for multiple processes to every configuration. Process creation is not the only piece of operating system support required to support multiple processes. The total cost of support for multiple processes may be quite high in some circumstances. Existing practice shows that support for multiple processes is uncommon and threads are common among “tiny kernels”. There should, therefore, probably continue to be AEPs for operating systems with only one process.

Asynchronous Error Notification

A library implementation of *posix_spawn()* or *posix_spawnnp()* may not be able to detect all possible errors before it forks the child process. POSIX.1-2008 provides for an error indication returned from a child process which could not successfully complete the spawn operation via a special exit status which may be detected using the status value returned by *wait()*, *waitid()*, and *waitpid()*.

The *stat_val* interface and the macros used to interpret it are not well suited to the purpose of returning API errors, but they are the only path available to a library implementation. Thus, an implementation may cause the child process to exit with exit status 127 for any error detected during the spawn process after the *posix_spawn()* or *posix_spawnnp()* function has successfully returned.

The standard developers had proposed using two additional macros to interpret *stat_val*. The first, *WIFSPAWNFAIL*, would have detected a status that indicated that the child exited because of an error detected during the *posix_spawn()* or *posix_spawnnp()* operations rather than during actual execution of the child process image; the second, *WSPAWNERRNO*, would have extracted the error value if *WIFSPAWNFAIL* indicated a failure. Unfortunately, the ballot group strongly opposed this because it would make a library implementation of *posix_spawn()* or *posix_spawnnp()* dependent on kernel modifications to *waitpid()* to be able to embed special information in *stat_val* to indicate a spawn failure.

The 8 bits of child process exit status that are guaranteed by POSIX.1-2008 to be accessible to the waiting parent process are insufficient to disambiguate a spawn error from any other kind of error that may be returned by an arbitrary process image. No other bits of the exit status are required to be visible in *stat_val*, so these macros could not be strictly implemented at the library level. Reserving an exit status of 127 for such spawn errors is consistent with the use of this value by *system()* and *popen()* to signal failures in these operations that occur after the function has returned but before a shell is able to execute. The exit status of 127 does not uniquely

identify this class of error, nor does it provide any detailed information on the nature of the failure. Note that a kernel implementation of *posix_spawn()* or *posix_spawnnp()* is permitted (and encouraged) to return any possible error as the function value, thus providing more detailed failure information to the parent process.

Thus, no special macros are available to isolate asynchronous *posix_spawn()* or *posix_spawnnp()* errors. Instead, errors detected by the *posix_spawn()* or *posix_spawnnp()* operations in the context of the child process before the new process image executes are reported by setting the child's exit status to 127. The calling process may use the WIFEXITED and WEXITSTATUS macros on the *stat_val* stored by the *wait()* or *waitpid()* functions to detect spawn failures to the extent that other status values with which the child process image may exit (before the parent can conclusively determine that the child process image has begun execution) are distinct from exit status 127.

FUTURE DIRECTIONS

None.

SEE ALSO

alarm(), *chmod()*, *close()*, *dup()*, *exec*, *exit()*, *fcntl()*, *fork()*, *fstatat()*, *kill()*, *open()*, *posix_spawn_file_actions_addclose()*, *posix_spawn_file_actions_adddup2()*, *posix_spawn_file_actions_destroy()*, *posix_spawnattr_destroy()*, *posix_spawnattr_getsigdefault()*, *posix_spawnattr_getflags()*, *posix_spawnattr_getpgroup()*, *posix_spawnattr_getschedparam()*, *posix_spawnattr_getschedpolicy()*, *posix_spawnattr_getsigmask()*, *sched_setparam()*, *sched_setscheduler()*, *setpgid()*, *setuid()*, *times()*, *wait()*, *waitid()*

XBD Chapter 8 (on page 173), [<spawn.h>](#)

CHANGE HISTORY

First released in Issue 6. Derived from IEEE Std 1003.1d-1999.

IEEE PASC Interpretation 1003.1 #103 is applied, noting that the signal default actions are changed as well as the signal mask in step 2.

IEEE PASC Interpretation 1003.1 #132 is applied.

Issue 7

Functionality relating to the Threads option is moved to the Base.

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0433 [291], XSH/TC1-2008/0434 [173], + and XSH/TC1-2008/0435 [173] are applied.

47672 **NAME**

47673 `posix_spawn_file_actions_addclose`, `posix_spawn_file_actions_addopen` — add close or open
 47674 action to spawn file actions object (**ADVANCED REALTIME**)

47675 **SYNOPSIS**

```
47676 SPN      #include <spawn.h>
47677
47677      int posix_spawn_file_actions_addclose(posix_spawn_file_actions_t
47678          *file_actions, int fildes);
47679      int posix_spawn_file_actions_addopen(posix_spawn_file_actions_t
47680          *restrict file_actions, int fildes,
47681          const char *restrict path, int oflag, mode_t mode);
```

47682 **DESCRIPTION**

47683 These functions shall add or delete a close or open action to a spawn file actions object.

47684 A spawn file actions object is of type **posix_spawn_file_actions_t** (defined in **<spawn.h>**) and is
 47685 used to specify a series of actions to be performed by a *posix_spawn()* or *posix_spawnnp()*
 47686 operation in order to arrive at the set of open file descriptors for the child process given the set
 47687 of open file descriptors of the parent. POSIX.1-2008 does not define comparison or assignment
 47688 operators for the type **posix_spawn_file_actions_t**.

47689 A spawn file actions object, when passed to *posix_spawn()* or *posix_spawnnp()*, shall specify how
 47690 the set of open file descriptors in the calling process is transformed into a set of potentially open
 47691 file descriptors for the spawned process. This transformation shall be as if the specified sequence
 47692 of actions was performed exactly once, in the context of the spawned process (prior to execution
 47693 of the new process image), in the order in which the actions were added to the object;
 47694 additionally, when the new process image is executed, any file descriptor (from this new set)
 47695 which has its FD_CLOEXEC flag set shall be closed (see *posix_spawn()*).

47696 The *posix_spawn_file_actions_addclose()* function shall add a *close* action to the object referenced
 47697 by *file_actions* that shall cause the file descriptor *fildes* to be closed (as if *close(fildes)* had been
 47698 called) when a new process is spawned using this file actions object.

47699 The *posix_spawn_file_actions_addopen()* function shall add an *open* action to the object referenced
 47700 by *file_actions* that shall cause the file named by *path* to be opened (as if *open(path, oflag, mode)*
 47701 had been called, and the returned file descriptor, if not *fildes*, had been changed to *fildes*) when a
 47702 new process is spawned using this file actions object. If *fildes* was already an open file descriptor,
 47703 it shall be closed before the new file is opened.

47704 The string described by *path* shall be copied by the *posix_spawn_file_actions_addopen()* function.

47705 **RETURN VALUE**

47706 Upon successful completion, these functions shall return zero; otherwise, an error number shall
 47707 be returned to indicate the error.

47708 **ERRORS**

47709 The *posix_spawn_file_actions_addopen()* function shall fail if:

47710 [EBADF] The value specified by *fildes* is negative or greater than or equal to
 47711 {OPEN_MAX}.

47712 The *posix_spawn_file_actions_addclose()* function shall fail if:

47713 [EBADF] The value specified by *fildes* is negative.

47714 These functions may fail if:

47715 [EINVAL] The value specified by *file_actions* is invalid.

47716 [ENOMEM] Insufficient memory exists to add to the spawn file actions object.

47717 It shall not be considered an error for the *fildev* argument passed to these functions to specify a
47718 file descriptor for which the specified operation could not be performed at the time of the call.
47719 Any such error will be detected when the associated file actions object is later used during a
47720 *posix_spawn()* or *posix_spawnnp()* operation.

47721 EXAMPLES

47722 None.

47723 APPLICATION USAGE

47724 These functions are part of the Spawn option and need not be provided on all implementations.

47725 Implementations may use file descriptors that must be inherited into child processes for the
47726 child process to remain conforming, such as for message catalog or tracing purposes. Therefore,
47727 an application that calls *posix_spawn_file_actions_addclose()* with an arbitrary integer risks non-
47728 conforming behavior, and this function can only portably be used to close file descriptor values
47729 that the application has obtained through explicit actions, or for the three file descriptors
47730 corresponding to the standard file streams. In order to avoid a race condition of leaking an
47731 unintended file descriptor into a child process, an application should consider opening all file
47732 descriptors with the FD_CLOEXEC bit set unless the file descriptor is intended to be inherited
47733 across *exec*.

47734 RATIONALE

47735 A spawn file actions object may be initialized to contain an ordered sequence of *close()*, *dup2()*,
47736 and *open()* operations to be used by *posix_spawn()* or *posix_spawnnp()* to arrive at the set of open
47737 file descriptors inherited by the spawned process from the set of open file descriptors in the
47738 parent at the time of the *posix_spawn()* or *posix_spawnnp()* call. It had been suggested that the
47739 *close()* and *dup2()* operations alone are sufficient to rearrange file descriptors, and that files
47740 which need to be opened for use by the spawned process can be handled either by having the
47741 calling process open them before the *posix_spawn()* or *posix_spawnnp()* call (and close them after),
47742 or by passing pathnames to the spawned process (in *argv*) so that it may open them itself. The
47743 standard developers recommend that applications use one of these two methods when practical,
47744 since detailed error status on a failed open operation is always available to the application this
47745 way. However, the standard developers feel that allowing a spawn file actions object to specify
47746 open operations is still appropriate because:

- 47747 1. It is consistent with equivalent POSIX.5 (Ada) functionality.
- 47748 2. It supports the I/O redirection paradigm commonly employed by POSIX programs
47749 designed to be invoked from a shell. When such a program is the child process, it may not
47750 be designed to open files on its own.
- 47751 3. It allows file opens that might otherwise fail or violate file ownership/access rights if
47752 executed by the parent process.

47753 Regarding 2. above, note that the spawn open file action provides to *posix_spawn()* and
47754 *posix_spawnnp()* the same capability that the shell redirection operators provide to *system()*, only
47755 without the intervening execution of a shell; for example:

47756 `system ("myprog <file1 3<file2");`

47757 Regarding 3. above, note that if the calling process needs to open one or more files for access by
47758 the spawned process, but has insufficient spare file descriptors, then the open action is necessary

to allow the *open()* to occur in the context of the child process after other file descriptors have been closed (that must remain open in the parent).

Additionally, if a parent is executed from a file having a “set-user-id” mode bit set and the POSIX_SPAWN_RESETPIDS flag is set in the spawn attributes, a file created within the parent process will (possibly incorrectly) have the parent’s effective user ID as its owner, whereas a file created via an *open()* action during *posix_spawn()* or *posix_spawnnp()* will have the parent’s real ID as its owner; and an open by the parent process may successfully open a file to which the real user should not have access or fail to open a file to which the real user should have access.

File Descriptor Mapping

The standard developers had originally proposed using an array which specified the mapping of child file descriptors back to those of the parent. It was pointed out by the ballot group that it is not possible to reshuffle file descriptors arbitrarily in a library implementation of *posix_spawn()* or *posix_spawnnp()* without provision for one or more spare file descriptor entries (which simply may not be available). Such an array requires that an implementation develop a complex strategy to achieve the desired mapping without inadvertently closing the wrong file descriptor at the wrong time.

It was noted by a member of the Ada Language Bindings working group that the approved Ada Language *Start_Process* family of POSIX process primitives use a caller-specified set of file actions to alter the normal *fork()/exec* semantics for inheritance of file descriptors in a very flexible way, yet no such problems exist because the burden of determining how to achieve the final file descriptor mapping is completely on the application. Furthermore, although the file actions interface appears frightening at first glance, it is actually quite simple to implement in either a library or the kernel.

The *posix_spawn_file_actions_addclose()* function is not required to check whether the file descriptor is less than {OPEN_MAX} because on some implementations {OPEN_MAX} reflects the RLIMIT_NOFILE soft limit and therefore calling *setrlimit()* to reduce this limit can result in an {OPEN_MAX} value less than or equal to an already open file descriptor. Applications need to be able to close such file descriptors on spawn. On implementations where {OPEN_MAX} does not change, it is recommended that *posix_spawn_file_actions_addclose()* should return [EBADF] if *filides* is greater than or equal to {OPEN_MAX}.

FUTURE DIRECTIONS

None.

SEE ALSO

close(), *dup()*, *open()*, *posix_spawn()*, *posix_spawn_file_actions_adddup2()*, *posix_spawn_file_actions_destroy()*

XBD <spawn.h>

CHANGE HISTORY

First released in Issue 6. Derived from IEEE Std 1003.1d-1999.

IEEE PASC Interpretation 1003.1 #105 is applied, adding a note to the DESCRIPTION that the string pointed to by *path* is copied by the *posix_spawn_file_actions_addopen()* function.

Issue 7

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0436 [418], XSH/TC1-2008/0437 [149], XSH/TC1-2008/0438 [291], and XSH/TC1-2008/0439 [418] are applied.

NAME

posix_spawn_file_actions_adddup2 — add dup2 action to spawn file actions object (ADVANCED REALTIME)

SYNOPSIS

```
#include <spawn.h>

int posix_spawn_file_actions_adddup2(posix_spawn_file_actions_t
    *file_actions, int fildes, int newfildes);
```

DESCRIPTION

The *posix_spawn_file_actions_adddup2()* function shall add a *dup2()* action to the object referenced by *file_actions* that shall cause the file descriptor *fildes* to be duplicated as *newfildes* (as if *dup2(fildes, newfildes)* had been called) when a new process is spawned using this file actions object.

A spawn file actions object is as defined in *posix_spawn_file_actions_addclose()*.

RETURN VALUE

Upon successful completion, the *posix_spawn_file_actions_adddup2()* function shall return zero; otherwise, an error number shall be returned to indicate the error.

ERRORS

The *posix_spawn_file_actions_adddup2()* function shall fail if:

[EBADF] The value specified by *fildes* or *newfildes* is negative or greater than or equal to {OPEN_MAX}.

[ENOMEM] Insufficient memory exists to add to the spawn file actions object.

The *posix_spawn_file_actions_adddup2()* function may fail if:

[EINVAL] The value specified by *file_actions* is invalid.

It shall not be considered an error for the *fildes* argument passed to the *posix_spawn_file_actions_adddup2()* function to specify a file descriptor for which the specified operation could not be performed at the time of the call. Any such error will be detected when the associated file actions object is later used during a *posix_spawn()* or *posix_spawnnp()* operation.

EXAMPLES

None.

APPLICATION USAGE

The *posix_spawn_file_actions_adddup2()* function is part of the Spawn option and need not be provided on all implementations.

Implementations may use file descriptors that must be inherited into child processes for the child process to remain conforming, such as for message catalog or tracing purposes. Therefore, an application that calls *posix_spawn_file_actions_adddup2()* with an arbitrary integer for *newfildes* risks non-conforming behavior, and this function can only portably be used to overwrite file descriptor values that the application has obtained through explicit actions, or for the three file descriptors corresponding to the standard file streams. In order to avoid a race condition of leaking an unintended file descriptor into a child process, an application should consider opening all file descriptors with the FD_CLOEXEC bit set unless the file descriptor is intended to be inherited across *exec*.

47844 **RATIONALE**47845 Refer to the RATIONALE section in *posix_spawn_file_actions_addclose()*. |47846 **FUTURE DIRECTIONS**

47847 None.

47848 **SEE ALSO**47849 *dup()*, *posix_spawn()*, *posix_spawn_file_actions_addclose()*, *posix_spawn_file_actions_destroy()*

47850 XBD <spawn.h>

47851 **CHANGE HISTORY**

47852 First released in Issue 6. Derived from IEEE Std 1003.1d-1999.

47853 IEEE PASC Interpretation 1003.1 #104 is applied, noting that the [EBADF] error can apply to the

47854 *newfildes* argument in addition to *fildes*.47855 **Issue 7**

47856 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0440 [149] is applied. +

47857 NAME

47858 posix_spawn_file_actions_addopen — add open action to spawn file actions object
47859 (ADVANCED REALTIME)

47860 SYNOPSIS

```
47861 SPN    #include <spawn.h>
47862
47862    int posix_spawn_file_actions_addopen(posix_spawn_file_actions_t
47863        *restrict file_actions, int fildes,
47864        const char *restrict path, int oflag, mode_t mode);
```

47865 DESCRIPTION

47866 Refer to [posix_spawn_file_actions_addclose\(\)](#).

47867 **NAME**

47868 posix_spawn_file_actions_destroy, posix_spawn_file_actions_init — destroy and initialize
 47869 spawn file actions object (**ADVANCED REALTIME**)

47870 **SYNOPSIS**

```
47871 SPN      #include <spawn.h>
47872
47873      int posix_spawn_file_actions_destroy(posix_spawn_file_actions_t
47874      *file_actions);
47875      int posix_spawn_file_actions_init(posix_spawn_file_actions_t
47876      *file_actions);
```

47876 **DESCRIPTION**

47877 The *posix_spawn_file_actions_destroy()* function shall destroy the object referenced by *file_actions*;
 47878 the object becomes, in effect, uninitialized. An implementation may cause
 47879 *posix_spawn_file_actions_destroy()* to set the object referenced by *file_actions* to an invalid value. A
 47880 destroyed spawn file actions object can be reinitialized using *posix_spawn_file_actions_init()*; the
 47881 results of otherwise referencing the object after it has been destroyed are undefined.

47882 The *posix_spawn_file_actions_init()* function shall initialize the object referenced by *file_actions* to
 47883 contain no file actions for *posix_spawn()* or *posix_spawnnp()* to perform.

47884 A spawn file actions object is as defined in *posix_spawn_file_actions_addclose()*.

47885 The effect of initializing an already initialized spawn file actions object is undefined.

47886 **RETURN VALUE**

47887 Upon successful completion, these functions shall return zero; otherwise, an error number shall
 47888 be returned to indicate the error.

47889 **ERRORS**

47890 The *posix_spawn_file_actions_init()* function shall fail if:

47891 [ENOMEM] Insufficient memory exists to initialize the spawn file actions object.

47892 The *posix_spawn_file_actions_destroy()* function may fail if:

47893 [EINVAL] The value specified by *file_actions* is invalid.

47894 **EXAMPLES**

47895 None.

47896 **APPLICATION USAGE**

47897 These functions are part of the Spawn option and need not be provided on all implementations.

47898 **RATIONALE**

47899 Refer to the RATIONALE section in *posix_spawn_file_actions_addclose()*.

47900 **FUTURE DIRECTIONS**

47901 None.

47902 **SEE ALSO**

47903 *posix_spawn()*, *posix_spawn_file_actions_addclose()*

47904 XBD *<spawn.h>*

CHANGE HISTORY

- 47905
- 47906 First released in Issue 6. Derived from IEEE Std 1003.1d-1999.
- 47907 In the SYNOPSIS, the inclusion of **<sys/types.h>** is no longer required.

NAME

posix_spawnattr_destroy, posix_spawnattr_init — destroy and initialize spawn attributes object
(**ADVANCED REALTIME**)

SYNOPSIS

```
SPN      #include <spawn.h>

int posix_spawnattr_destroy(posix_spawnattr_t *attr);
int posix_spawnattr_init(posix_spawnattr_t *attr);
```

DESCRIPTION

The *posix_spawnattr_destroy()* function shall destroy a spawn attributes object. A destroyed *attr* attributes object can be reinitialized using *posix_spawnattr_init()*; the results of otherwise referencing the object after it has been destroyed are undefined. An implementation may cause *posix_spawnattr_destroy()* to set the object referenced by *attr* to an invalid value.

The *posix_spawnattr_init()* function shall initialize a spawn attributes object *attr* with the default value for all of the individual attributes used by the implementation. Results are undefined if *posix_spawnattr_init()* is called specifying an already initialized *attr* attributes object.

A spawn attributes object is of type **posix_spawnattr_t** (defined in **<spawn.h>**) and is used to specify the inheritance of process attributes across a spawn operation. POSIX.1-2008 does not define comparison or assignment operators for the type **posix_spawnattr_t**.

Each implementation shall document the individual attributes it uses and their default values unless these values are defined by POSIX.1-2008. Attributes not defined by POSIX.1-2008, their default values, and the names of the associated functions to get and set those attribute values are implementation-defined.

The resulting spawn attributes object (possibly modified by setting individual attribute values), is used to modify the behavior of *posix_spawn()* or *posix_spawnnp()*. After a spawn attributes object has been used to spawn a process by a call to a *posix_spawn()* or *posix_spawnnp()*, any function affecting the attributes object (including destruction) shall not affect any process that has been spawned in this way.

RETURN VALUE

Upon successful completion, *posix_spawnattr_destroy()* and *posix_spawnattr_init()* shall return zero; otherwise, an error number shall be returned to indicate the error.

ERRORS

The *posix_spawnattr_init()* function shall fail if:

[ENOMEM] Insufficient memory exists to initialize the spawn attributes object.

The *posix_spawnattr_destroy()* function may fail if:

[EINVAL] The value specified by *attr* is invalid.

EXAMPLES

None.

APPLICATION USAGE

These functions are part of the Spawn option and need not be provided on all implementations.

RATIONALE

The original spawn interface proposed in POSIX.1-2008 defined the attributes that specify the inheritance of process attributes across a spawn operation as a structure. In order to be able to separate optional individual attributes under their appropriate options (that is, the *spawn-schedparam* and *spawn-schedpolicy* attributes depending upon the Process Scheduling option), and

also for extensibility and consistency with the newer POSIX interfaces, the attributes interface has been changed to an opaque data type. This interface now consists of the type **posix_spawnattr_t**, representing a spawn attributes object, together with associated functions to initialize or destroy the attributes object, and to set or get each individual attribute. Although the new object-oriented interface is more verbose than the original structure, it is simple to use, more extensible, and easy to implement.

FUTURE DIRECTIONS

None.

SEE ALSO

posix_spawn(), *posix_spawnattr_getsigdefault()*, *posix_spawnattr_getflags()*,
posix_spawnattr_getpgroup(), *posix_spawnattr_getschedparam()*, *posix_spawnattr_getschedpolicy()*,
posix_spawnattr_getsigmask()

XBD <spawn.h>

CHANGE HISTORY

First released in Issue 6. Derived from IEEE Std 1003.1d-1999.

IEEE PASC Interpretation 1003.1 #106 is applied, noting that the effect of initializing an already initialized spawn attributes option is undefined.

47969 **NAME**

47970 `posix_spawnattr_getflags`, `posix_spawnattr_setflags` — get and set the spawn-flags attribute of a
 47971 spawn attributes object (**ADVANCED REALTIME**)

47972 **SYNOPSIS**

```
47973 SPN    #include <spawn.h>
47974
47975    int posix_spawnattr_getflags(const posix_spawnattr_t *restrict attr,
47976                                short *restrict flags);
47976    int posix_spawnattr_setflags(posix_spawnattr_t *attr, short flags);
```

47977 **DESCRIPTION**

47978 The `posix_spawnattr_getflags()` function shall obtain the value of the *spawn-flags* attribute from the
 47979 attributes object referenced by *attr*.

47980 The `posix_spawnattr_setflags()` function shall set the *spawn-flags* attribute in an initialized
 47981 attributes object referenced by *attr*.

47982 The *spawn-flags* attribute is used to indicate which process attributes are to be changed in the
 47983 new process image when invoking `posix_spawn()` or `posix_spawnnp()`. It is the bitwise-inclusive
 47984 OR of zero or more of the following flags:

```
47985 POSIX_SPAWN_RESETIDS
47986 POSIX_SPAWN_SETPGROUP
47987 POSIX_SPAWN_SETSIGDEF
47988 POSIX_SPAWN_SETSIGMASK
47989 PS  POSIX_SPAWN_SETSCHEDPARAM
47990    POSIX_SPAWN_SETSCHEDULER
```

47991 These flags are defined in **<spawn.h>**. The default value of this attribute shall be as if no flags
 47992 were set.

47993 **RETURN VALUE**

47994 Upon successful completion, `posix_spawnattr_getflags()` shall return zero and store the value of
 47995 the *spawn-flags* attribute of *attr* into the object referenced by the *flags* parameter; otherwise, an
 47996 error number shall be returned to indicate the error.

47997 Upon successful completion, `posix_spawnattr_setflags()` shall return zero; otherwise, an error
 47998 number shall be returned to indicate the error.

47999 **ERRORS**

48000 These functions may fail if:

48001 [EINVAL] The value specified by *attr* is invalid.

48002 The `posix_spawnattr_setflags()` function may fail if:

48003 [EINVAL] The value of the attribute being set is not valid.

48004 EXAMPLES

48005 None.

48006 APPLICATION USAGE

48007 These functions are part of the Spawn option and need not be provided on all implementations.

48008 RATIONALE

48009 None.

48010 FUTURE DIRECTIONS

48011 None.

48012 SEE ALSO

48013 *posix_spawn()*, *posix_spawnattr_destroy()*, *posix_spawnattr_getsigdefault()*,
48014 *posix_spawnattr_getpgroup()*, *posix_spawnattr_getschedparam()*, *posix_spawnattr_getschedpolicy()*,
48015 *posix_spawnattr_getsigmask()*

48016 XBD **<spawn.h>**

48017 CHANGE HISTORY

48018 First released in Issue 6. Derived from IEEE Std 1003.1d-1999.

NAME

posix_spawnattr_getpgroup, posix_spawnattr_setpgroup — get and set the spawn-pgroup attribute of a spawn attributes object (**ADVANCED REALTIME**)

SYNOPSIS

```
#include <spawn.h>

int posix_spawnattr_getpgroup(const posix_spawnattr_t *restrict attr,
    pid_t *restrict pgroup);
int posix_spawnattr_setpgroup(posix_spawnattr_t *attr, pid_t pgroup);
```

DESCRIPTION

The *posix_spawnattr_getpgroup()* function shall obtain the value of the *spawn-pgroup* attribute from the attributes object referenced by *attr*.

The *posix_spawnattr_setpgroup()* function shall set the *spawn-pgroup* attribute in an initialized attributes object referenced by *attr*.

The *spawn-pgroup* attribute represents the process group to be joined by the new process image in a spawn operation (if `POSIX_SPAWN_SETPGROUP` is set in the *spawn-flags* attribute). The default value of this attribute shall be zero.

RETURN VALUE

Upon successful completion, *posix_spawnattr_getpgroup()* shall return zero and store the value of the *spawn-pgroup* attribute of *attr* into the object referenced by the *pgroup* parameter; otherwise, an error number shall be returned to indicate the error.

Upon successful completion, *posix_spawnattr_setpgroup()* shall return zero; otherwise, an error number shall be returned to indicate the error.

ERRORS

These functions may fail if:

[EINVAL] The value specified by *attr* is invalid.

The *posix_spawnattr_setpgroup()* function may fail if:

[EINVAL] The value of the attribute being set is not valid.

EXAMPLES

None.

APPLICATION USAGE

These functions are part of the Spawn option and need not be provided on all implementations.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

posix_spawn(), *posix_spawnattr_destroy()*, *posix_spawnattr_getsigdefault()*,
posix_spawnattr_getflags(), *posix_spawnattr_getschedparam()*, *posix_spawnattr_getschedpolicy()*,
posix_spawnattr_getsigmask()

XBD `<spawn.h>`

48059	CHANGE HISTORY
48060	First released in Issue 6. Derived from IEEE Std 1003.1d-1999.

NAME

posix_spawnattr_getschedparam, posix_spawnattr_setschedparam — get and set the spawn-schedparam attribute of a spawn attributes object (**ADVANCED REALTIME**)

SYNOPSIS

```
SPN PS  #include <spawn.h>
         #include <sched.h>

int posix_spawnattr_getschedparam(const posix_spawnattr_t
    *restrict attr, struct sched_param *restrict schedparam);
int posix_spawnattr_setschedparam(posix_spawnattr_t *restrict attr,
    const struct sched_param *restrict schedparam);
```

DESCRIPTION

The *posix_spawnattr_getschedparam()* function shall obtain the value of the *spawn-schedparam* attribute from the attributes object referenced by *attr*.

The *posix_spawnattr_setschedparam()* function shall set the *spawn-schedparam* attribute in an initialized attributes object referenced by *attr*.

The *spawn-schedparam* attribute represents the scheduling parameters to be assigned to the new process image in a spawn operation (if `POSIX_SPAWN_SETSCHEDULER` or `POSIX_SPAWN_SETSCHEDPARAM` is set in the *spawn-flags* attribute). The default value of this attribute is unspecified.

RETURN VALUE

Upon successful completion, *posix_spawnattr_getschedparam()* shall return zero and store the value of the *spawn-schedparam* attribute of *attr* into the object referenced by the *schedparam* parameter; otherwise, an error number shall be returned to indicate the error.

Upon successful completion, *posix_spawnattr_setschedparam()* shall return zero; otherwise, an error number shall be returned to indicate the error.

ERRORS

These functions may fail if:

[EINVAL] The value specified by *attr* is invalid.

The *posix_spawnattr_setschedparam()* function may fail if:

[EINVAL] The value of the attribute being set is not valid.

EXAMPLES

None.

APPLICATION USAGE

These functions are part of the Spawn and Process Scheduling options and need not be provided on all implementations.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

posix_spawn(), *posix_spawnattr_destroy()*, *posix_spawnattr_getsigdefault()*,
posix_spawnattr_getflags(), *posix_spawnattr_getpgroup()*, *posix_spawnattr_getschedpolicy()*,
posix_spawnattr_getsigmask()

48104	XBD <sched.h>, <spawn.h>
48105	CHANGE HISTORY
48106	First released in Issue 6. Derived from IEEE Std 1003.1d-1999.

48107 **NAME**

48108 posix_spawnattr_getschedpolicy, posix_spawnattr_setschedpolicy — get and set the spawn-
 48109 schedpolicy attribute of a spawn attributes object (**ADVANCED REALTIME**)

48110 **SYNOPSIS**

```
48111 SPN PS #include <spawn.h>
48112         #include <sched.h>

48113         int posix_spawnattr_getschedpolicy(const posix_spawnattr_t
48114             *restrict attr, int *restrict schedpolicy);
48115         int posix_spawnattr_setschedpolicy(posix_spawnattr_t *attr,
48116             int schedpolicy);
```

48117 **DESCRIPTION**

48118 The *posix_spawnattr_getschedpolicy()* function shall obtain the value of the *spawn-schedpolicy*
 48119 attribute from the attributes object referenced by *attr*.

48120 The *posix_spawnattr_setschedpolicy()* function shall set the *spawn-schedpolicy* attribute in an
 48121 initialized attributes object referenced by *attr*.

48122 The *spawn-schedpolicy* attribute represents the scheduling policy to be assigned to the new
 48123 process image in a spawn operation (if POSIX_SPAWN_SETSCHEDULER is set in the *spawn-*
 48124 *flags* attribute). The default value of this attribute is unspecified.

48125 **RETURN VALUE**

48126 Upon successful completion, *posix_spawnattr_getschedpolicy()* shall return zero and store the
 48127 value of the *spawn-schedpolicy* attribute of *attr* into the object referenced by the *schedpolicy*
 48128 parameter; otherwise, an error number shall be returned to indicate the error.

48129 Upon successful completion, *posix_spawnattr_setschedpolicy()* shall return zero; otherwise, an
 48130 error number shall be returned to indicate the error.

48131 **ERRORS**

48132 These functions may fail if:

48133 [EINVAL] The value specified by *attr* is invalid.

48134 The *posix_spawnattr_setschedpolicy()* function may fail if:

48135 [EINVAL] The value of the attribute being set is not valid.

48136 **EXAMPLES**

48137 None.

48138 **APPLICATION USAGE**

48139 These functions are part of the Spawn and Process Scheduling options and need not be provided
 48140 on all implementations.

48141 **RATIONALE**

48142 None.

48143 **FUTURE DIRECTIONS**

48144 None.

48145 **SEE ALSO**

48146 *posix_spawn()*, *posix_spawnattr_destroy()*, *posix_spawnattr_getsigdefault()*,
 48147 *posix_spawnattr_getflags()*, *posix_spawnattr_getpgroup()*, *posix_spawnattr_getschedparam()*,
 48148 *posix_spawnattr_getsigmask()*

48149 XBD [<sched.h>](#), [<spawn.h>](#)

48150	CHANGE HISTORY
48151	First released in Issue 6. Derived from IEEE Std 1003.1d-1999.

NAME

posix_spawnattr_getsigdefault, posix_spawnattr_setsigdefault — get and set the spawn-sigdefault attribute of a spawn attributes object (**ADVANCED REALTIME**)

SYNOPSIS

```
SPN      #include <signal.h>
          #include <spawn.h>

          int posix_spawnattr_getsigdefault(const posix_spawnattr_t
          *restrict attr, sigset_t *restrict sigdefault);
          int posix_spawnattr_setsigdefault(posix_spawnattr_t *restrict attr,
          const sigset_t *restrict sigdefault);
```

DESCRIPTION

The *posix_spawnattr_getsigdefault()* function shall obtain the value of the *spawn-sigdefault* attribute from the attributes object referenced by *attr*.

The *posix_spawnattr_setsigdefault()* function shall set the *spawn-sigdefault* attribute in an initialized attributes object referenced by *attr*.

The *spawn-sigdefault* attribute represents the set of signals to be forced to default signal handling in the new process image (if *POSIX_SPAWN_SETSIGDEF* is set in the *spawn-flags* attribute) by a spawn operation. The default value of this attribute shall be an empty signal set.

RETURN VALUE

Upon successful completion, *posix_spawnattr_getsigdefault()* shall return zero and store the value of the *spawn-sigdefault* attribute of *attr* into the object referenced by the *sigdefault* parameter; otherwise, an error number shall be returned to indicate the error.

Upon successful completion, *posix_spawnattr_setsigdefault()* shall return zero; otherwise, an error number shall be returned to indicate the error.

ERRORS

These functions may fail if:

[EINVAL] The value specified by *attr* is invalid.

The *posix_spawnattr_setsigdefault()* function may fail if:

[EINVAL] The value of the attribute being set is not valid.

EXAMPLES

None.

APPLICATION USAGE

These functions are part of the Spawn option and need not be provided on all implementations.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

posix_spawn(), *posix_spawnattr_destroy()*, *posix_spawnattr_getflags()*, *posix_spawnattr_getpgroup()*, *posix_spawnattr_getschedparam()*, *posix_spawnattr_getschedpolicy()*, *posix_spawnattr_getsigmask()*

XBD *<signal.h>*, *<spawn.h>*

48193	CHANGE HISTORY First released in Issue 6. Derived from IEEE Std 1003.1d-1999.
48194	

48195 **NAME**

48196 `posix_spawnattr_getsigmask`, `posix_spawnattr_setsigmask` — get and set the spawn-sigmask
 48197 attribute of a spawn attributes object (**ADVANCED REALTIME**)

48198 **SYNOPSIS**

```
48199 SPN      #include <signal.h>
48200           #include <spawn.h>

48201           int posix_spawnattr_getsigmask(const posix_spawnattr_t *restrict attr,
48202           sigset_t *restrict sigmask);
48203           int posix_spawnattr_setsigmask(posix_spawnattr_t *restrict attr,
48204           const sigset_t *restrict sigmask);
```

48205 **DESCRIPTION**

48206 The `posix_spawnattr_getsigmask()` function shall obtain the value of the *spawn-sigmask* attribute
 48207 from the attributes object referenced by *attr*.

48208 The `posix_spawnattr_setsigmask()` function shall set the *spawn-sigmask* attribute in an initialized
 48209 attributes object referenced by *attr*.

48210 The *spawn-sigmask* attribute represents the signal mask in effect in the new process image of a
 48211 spawn operation (if `POSIX_SPAWN_SETSIGMASK` is set in the *spawn-flags* attribute). The
 48212 default value of this attribute is unspecified.

48213 **RETURN VALUE**

48214 Upon successful completion, `posix_spawnattr_getsigmask()` shall return zero and store the value
 48215 of the *spawn-sigmask* attribute of *attr* into the object referenced by the *sigmask* parameter;
 48216 otherwise, an error number shall be returned to indicate the error.

48217 Upon successful completion, `posix_spawnattr_setsigmask()` shall return zero; otherwise, an error
 48218 number shall be returned to indicate the error.

48219 **ERRORS**

48220 These functions may fail if:

48221 [EINVAL] The value specified by *attr* is invalid.

48222 The `posix_spawnattr_setsigmask()` function may fail if:

48223 [EINVAL] The value of the attribute being set is not valid.

48224 **EXAMPLES**

48225 None.

48226 **APPLICATION USAGE**

48227 These functions are part of the Spawn option and need not be provided on all implementations.

48228 **RATIONALE**

48229 None.

48230 **FUTURE DIRECTIONS**

48231 None.

48232 **SEE ALSO**

48233 [*posix_spawn\(\)*](#), [*posix_spawnattr_destroy\(\)*](#), [*posix_spawnattr_getsigdefault\(\)*](#),
 48234 [*posix_spawnattr_getflags\(\)*](#), [*posix_spawnattr_getpgroup\(\)*](#), [*posix_spawnattr_getschedparam\(\)*](#),
 48235 [*posix_spawnattr_getschedpolicy\(\)*](#)

48236 XBD [*<signal.h>*](#), [*<spawn.h>*](#)

48237	CHANGE HISTORY
48238	First released in Issue 6. Derived from IEEE Std 1003.1d-1999.

48239 **NAME**

48240 posix_spawnattr_init — initialize the spawn attributes object (ADVANCED REALTIME)

48241 **SYNOPSIS**48242 SPN `#include <spawn.h>`48243 `int posix_spawnattr_init(posix_spawnattr_t *attr);`48244 **DESCRIPTION**48245 Refer to *posix_spawnattr_destroy()*.

48246 NAME

48247 posix_spawnattr_setflags — set the spawn-flags attribute of a spawn attributes object
48248 (ADVANCED REALTIME)

48249 SYNOPSIS

```
48250 SPN    #include <spawn.h>  
48251        int posix_spawnattr_setflags(posix_spawnattr_t *attr, short flags);
```

48252 DESCRIPTION

48253 Refer to *posix_spawnattr_getflags()*.

48254 **NAME**

48255 **posix_spawnattr_setpgroup** — set the spawn-pgroup attribute of a spawn attributes object
48256 (**ADVANCED REALTIME**)

48257 **SYNOPSIS**

48258 SPN `#include <spawn.h>`
48259 `int posix_spawnattr_setpgroup(posix_spawnattr_t *attr, pid_t pgroup);`

48260 **DESCRIPTION**

48261 Refer to *posix_spawnattr_getpgroup()*.

48262 NAME

48263 posix_spawnattr_setschedparam — set the spawn-schedparam attribute of a spawn attributes
48264 object (**ADVANCED REALTIME**)

48265 SYNOPSIS

```
48266 SPN PS #include <sched.h>  
48267 #include <spawn.h>  
  
48268 int posix_spawnattr_setschedparam(posix_spawnattr_t *restrict attr,  
48269     const struct sched_param *restrict schedparam);
```

48270 DESCRIPTION

48271 Refer to [*posix_spawnattr_getschedparam\(\)*](#).

48272 NAME

48273 `posix_spawnattr_setschedpolicy` — set the spawn-schedpolicy attribute of a spawn attributes
48274 object (**ADVANCED REALTIME**)

48275 SYNOPSIS

```
48276 SPN PS  #include <sched.h>  
48277          #include <spawn.h>  
  
48278          int posix_spawnattr_setschedpolicy(posix_spawnattr_t *attr,  
48279          int schedpolicy);
```

48280 DESCRIPTION

48281 Refer to *[posix_spawnattr_getschedpolicy\(\)](#)*.

NAME

posix_spawnattr_setsigdefault — set the spawn-sigdefault attribute of a spawn attributes object
(ADVANCED REALTIME)

SYNOPSIS

```
SPN      #include <signal.h>
          #include <spawn.h>

          int posix_spawnattr_setsigdefault(posix_spawnattr_t *restrict attr,
          const sigset_t *restrict sigdefault);
```

DESCRIPTION

Refer to *posix_spawnattr_getsigdefault()*.

48292 **NAME**

48293 posix_spawnattr_setsigmask — set the spawn-sigmask attribute of a spawn attributes object
48294 (**ADVANCED REALTIME**)

48295 **SYNOPSIS**

```
48296 SPN      #include <signal.h>  
48297          #include <spawn.h>  
  
48298          int posix_spawnattr_setsigmask(posix_spawnattr_t *restrict attr,  
48299          const sigset_t *restrict sigmask);
```

48300 **DESCRIPTION**

48301 Refer to *posix_spawnattr_getsigmask()*.

48302 **NAME**48303 posix_spawn — spawn a process (**ADVANCED REALTIME**)48304 **SYNOPSIS**

```
48305 SPN      #include <spawn.h>
48306          int posix_spawn(pid_t *restrict pid, const char *restrict file,
48307                        const posix_spawn_file_actions_t *file_actions,
48308                        const posix_spawnattr_t *restrict attrp,
48309                        char *const argv[restrict], char *const envp[restrict]);
```

48310 **DESCRIPTION**48311 Refer to *posix_spawn()*.

48312 **NAME**

48313 `posix_trace_attr_destroy`, `posix_trace_attr_init` — destroy and initialize the trace stream
 48314 attributes object (**TRACING**)

48315 **SYNOPSIS**

```
48316 OB TRC  #include <trace.h>
48317          int posix_trace_attr_destroy(trace_attr_t *attr);
48318          int posix_trace_attr_init(trace_attr_t *attr);
```

48319 **DESCRIPTION**

48320 The `posix_trace_attr_destroy()` function shall destroy an initialized trace attributes object. A
 48321 destroyed *attr* attributes object can be reinitialized using `posix_trace_attr_init()`; the results of
 48322 otherwise referencing the object after it has been destroyed are undefined.

48323 The `posix_trace_attr_init()` function shall initialize a trace attributes object *attr* with the default
 48324 value for all of the individual attributes used by a given implementation. The read-only
 48325 *generation-version* and *clock-resolution* attributes of the newly initialized trace attributes object
 48326 shall be set to their appropriate values (see [Section 2.11.1.2](#), on page 540).

48327 Results are undefined if `posix_trace_attr_init()` is called specifying an already initialized *attr*
 48328 attributes object.

48329 Implementations may add extensions to the trace attributes object structure as permitted in XBD
 48330 [Chapter 2](#) (on page 15).

48331 The resulting attributes object (possibly modified by setting individual attributes values), when
 48332 used by `posix_trace_create()`, defines the attributes of the trace stream created. A single attributes
 48333 object can be used in multiple calls to `posix_trace_create()`. After one or more trace streams have
 48334 been created using an attributes object, any function affecting that attributes object, including
 48335 destruction, shall not affect any trace stream previously created. An initialized attributes object
 48336 also serves to receive the attributes of an existing trace stream or trace log when calling the
 48337 `posix_trace_get_attr()` function.

48338 **RETURN VALUE**

48339 Upon successful completion, these functions shall return a value of zero. Otherwise, they shall
 48340 return the corresponding error number.

48341 **ERRORS**

48342 The `posix_trace_attr_destroy()` function may fail if:

48343 [EINVAL] The value of *attr* is invalid.

48344 The `posix_trace_attr_init()` function shall fail if:

48345 [ENOMEM] Insufficient memory exists to initialize the trace attributes object.

48346 **EXAMPLES**

48347 None.

48348 **APPLICATION USAGE**

48349 None.

48350 **RATIONALE**

48351 None.

48352 FUTURE DIRECTIONS

48353 The *posix_trace_attr_destroy()* and *posix_trace_attr_init()* functions may be removed in a future
48354 version.

48355 SEE ALSO

48356 *posix_trace_create()*, *posix_trace_get_attr()*, *uname()*

48357 XBD <trace.h>

48358 CHANGE HISTORY

48359 First released in Issue 6. Derived from IEEE Std 1003.1q-2000.

48360 IEEE PASC Interpretation 1003.1 #123 is applied.

48361 Issue 7

48362 The *posix_trace_attr_destroy()* and *posix_trace_attr_init()* functions are marked obsolescent.

NAME

posix_trace_attr_getclockres, posix_trace_attr_getcreatetime, posix_trace_attr_getgenversion, posix_trace_attr_getname, posix_trace_attr_setname — retrieve and set information about a trace stream (**TRACING**)

SYNOPSIS

```
OB TRC #include <time.h>
#include <trace.h>

int posix_trace_attr_getclockres(const trace_attr_t *attr,
    struct timespec *resolution);
int posix_trace_attr_getcreatetime(const trace_attr_t *attr,
    struct timespec *createtime);

#include <trace.h>

int posix_trace_attr_getgenversion(const trace_attr_t *attr,
    char *genversion);
int posix_trace_attr_getname(const trace_attr_t *attr,
    char *tracename);
int posix_trace_attr_setname(trace_attr_t *attr,
    const char *tracename);
```

DESCRIPTION

The *posix_trace_attr_getclockres()* function shall copy the clock resolution of the clock used to generate timestamps from the *clock-resolution* attribute of the attributes object pointed to by the *attr* argument into the structure pointed to by the *resolution* argument.

The *posix_trace_attr_getcreatetime()* function shall copy the trace stream creation time from the *creation-time* attribute of the attributes object pointed to by the *attr* argument into the structure pointed to by the *createtime* argument. The *creation-time* attribute shall represent the time of creation of the trace stream.

The *posix_trace_attr_getgenversion()* function shall copy the string containing version information from the *generation-version* attribute of the attributes object pointed to by the *attr* argument into the string pointed to by the *genversion* argument. The *genversion* argument shall be the address of a character array which can store at least {TRACE_NAME_MAX} characters.

The *posix_trace_attr_getname()* function shall copy the string containing the trace name from the *trace-name* attribute of the attributes object pointed to by the *attr* argument into the string pointed to by the *tracename* argument. The *tracename* argument shall be the address of a character array which can store at least {TRACE_NAME_MAX} characters.

The *posix_trace_attr_setname()* function shall set the name in the *trace-name* attribute of the attributes object pointed to by the *attr* argument, using the trace name string supplied by the *tracename* argument. If the supplied string contains more than {TRACE_NAME_MAX} characters, the name copied into the *trace-name* attribute may be truncated to one less than the length of {TRACE_NAME_MAX} characters. The default value is a null string.

RETURN VALUE

Upon successful completion, these functions shall return a value of zero. Otherwise, they shall return the corresponding error number.

If successful, the *posix_trace_attr_getclockres()* function stores the *clock-resolution* attribute value in the object pointed to by *resolution*. Otherwise, the content of this object is unspecified.

If successful, the *posix_trace_attr_getcreatetime()* function stores the trace stream creation time in

48408 the object pointed to by *createtime*. Otherwise, the content of this object is unspecified.

48409 If successful, the *posix_trace_attr_getgenversion()* function stores the trace version information in

48410 the string pointed to by *genversion*. Otherwise, the content of this string is unspecified.

48411 If successful, the *posix_trace_attr_getname()* function stores the trace name in the string pointed

48412 to by *tracename*. Otherwise, the content of this string is unspecified.

48413 ERRORS

48414 The *posix_trace_attr_getclockres()*, *posix_trace_attr_getcreatetime()*, *posix_trace_attr_getgenversion()*,

48415 and *posix_trace_attr_getname()* functions may fail if:

48416 [EINVAL] The value specified by one of the arguments is invalid.

48417 EXAMPLES

48418 None.

48419 APPLICATION USAGE

48420 None.

48421 RATIONALE

48422 None.

48423 FUTURE DIRECTIONS

48424 The *posix_trace_attr_getclockres()*, *posix_trace_attr_getcreatetime()*, *posix_trace_attr_getgenversion()*,

48425 *posix_trace_attr_getname()*, and *posix_trace_attr_setname()* functions may be removed in a future

48426 version.

48427 SEE ALSO

48428 *posix_trace_attr_destroy()*, *posix_trace_create()*, *posix_trace_get_attr()*, *uname()*

48429 XBD [<time.h>](#), [<trace.h>](#)

48430 CHANGE HISTORY

48431 First released in Issue 6. Derived from IEEE Std 1003.1q-2000.

48432 Issue 7

48433 The *posix_trace_attr_getclockres()*, *posix_trace_attr_getcreatetime()*, *posix_trace_attr_getgenversion()*,

48434 *posix_trace_attr_getname()*, and *posix_trace_attr_setname()* functions are marked obsolescent.

48435 **NAME**

48436 posix_trace_attr_getinherited, posix_trace_attr_getlogfullpolicy,
 48437 posix_trace_attr_getstreamfullpolicy, posix_trace_attr_setinherited,
 48438 posix_trace_attr_setlogfullpolicy, posix_trace_attr_setstreamfullpolicy — retrieve and set the
 48439 behavior of a trace stream (**TRACING**)

48440 **SYNOPSIS**

```
48441 OB TRC  #include <trace.h>

48442 TRI     int posix_trace_attr_getinherited(const trace_attr_t *restrict attr,
48443                                     int *restrict inheritancepolicy);
48444 TRL     int posix_trace_attr_getlogfullpolicy(const trace_attr_t *restrict attr,
48445                                     int *restrict logpolicy);
48446         int posix_trace_attr_getstreamfullpolicy(const trace_attr_t *restrict
48447         attr, int *restrict streampolicy);
48448 TRI     int posix_trace_attr_setinherited(trace_attr_t *attr,
48449                                     int inheritancepolicy);
48450 TRL     int posix_trace_attr_setlogfullpolicy(trace_attr_t *attr,
48451                                     int logpolicy);
48452         int posix_trace_attr_setstreamfullpolicy(trace_attr_t *attr,
48453         int streampolicy);
```

48454 **DESCRIPTION**

48455 TRI The *posix_trace_attr_getinherited()* and *posix_trace_attr_setinherited()* functions, respectively, shall
 48456 get and set the inheritance policy stored in the *inheritance* attribute for traced processes across the
 48457 *fork()* and *spawn()* operations. The *inheritance* attribute of the attributes object pointed to by the
 48458 *attr* argument shall be set to one of the following values defined by manifest constants in the
 48459 **<trace.h>** header:

48460 **POSIX_TRACE_CLOSE_FOR_CHILD**

48461 After a *fork()* or *spawn()* operation, the child shall not be traced, and tracing of the parent
 48462 shall continue.

48463 **POSIX_TRACE_INHERITED**

48464 After a *fork()* or *spawn()* operation, if the parent is being traced, its child shall be
 48465 concurrently traced using the same trace stream.

48466 The default value for the *inheritance* attribute is **POSIX_TRACE_CLOSE_FOR_CHILD**.

48467 TRL The *posix_trace_attr_getlogfullpolicy()* and *posix_trace_attr_setlogfullpolicy()* functions,
 48468 respectively, shall get and set the trace log full policy stored in the *log-full-policy* attribute of the
 48469 attributes object pointed to by the *attr* argument.

48470 The *log-full-policy* attribute shall be set to one of the following values defined by manifest
 48471 constants in the **<trace.h>** header:

48472 **POSIX_TRACE_LOOP**

48473 The trace log shall loop until the associated trace stream is stopped. This policy means that
 48474 when the trace log gets full, the file system shall reuse the resources allocated to the oldest
 48475 trace events that were recorded. In this way, the trace log will always contain the most
 48476 recent trace events flushed.

48477 **POSIX_TRACE_UNTIL_FULL**

48478 The trace stream shall be flushed to the trace log until the trace log is full. This condition can
 48479 be deduced from the *posix_log_full_status* member status (see the **posix_trace_status_info**
 48480 structure defined in **<trace.h>**). The last recorded trace event shall be the
 48481 **POSIX_TRACE_STOP** trace event.

POSIX_TRACE_APPEND

The associated trace stream shall be flushed to the trace log without log size limitation. If the application specifies POSIX_TRACE_APPEND, the implementation shall ignore the *log-max-size* attribute.

The default value for the *log-full-policy* attribute is POSIX_TRACE_LOOP.

The *posix_trace_attr_getstreamfullpolicy()* and *posix_trace_attr_setstreamfullpolicy()* functions, respectively, shall get and set the trace stream full policy stored in the *stream-full-policy* attribute of the attributes object pointed to by the *attr* argument.

The *stream-full-policy* attribute shall be set to one of the following values defined by manifest constants in the **<trace.h>** header:

POSIX_TRACE_LOOP

The trace stream shall loop until explicitly stopped by the *posix_trace_stop()* function. This policy means that when the trace stream is full, the trace system shall reuse the resources allocated to the oldest trace events recorded. In this way, the trace stream will always contain the most recent trace events recorded.

POSIX_TRACE_UNTIL_FULL

The trace stream will run until the trace stream resources are exhausted. Then the trace stream will stop. This condition can be deduced from *posix_stream_status* and *posix_stream_full_status* (see the **posix_trace_status_info** structure defined in **<trace.h>**). When this trace stream is read, a POSIX_TRACE_STOP trace event shall be reported after reporting the last recorded trace event. The trace system shall reuse the resources allocated to any trace events already reported—see the *posix_trace_getnext_event()*, *posix_trace_trygetnext_event()*, and *posix_trace_timedgetnext_event()* functions—or already flushed for an active trace stream with log if the Trace Log option is supported; see the *posix_trace_flush()* function. The trace system shall restart the trace stream when it is empty and may restart it sooner. A POSIX_TRACE_START trace event shall be reported before reporting the next recorded trace event.

POSIX_TRACE_FLUSH

If the Trace Log option is supported, this policy is identical to the POSIX_TRACE_UNTIL_FULL trace stream full policy except that the trace stream shall be flushed regularly as if *posix_trace_flush()* had been explicitly called. Defining this policy for an active trace stream without log shall be invalid.

The default value for the *stream-full-policy* attribute shall be POSIX_TRACE_LOOP for an active trace stream without log.

If the Trace Log option is supported, the default value for the *stream-full-policy* attribute shall be POSIX_TRACE_FLUSH for an active trace stream with log.

RETURN VALUE

Upon successful completion, these functions shall return a value of zero. Otherwise, they shall return the corresponding error number.

If successful, the *posix_trace_attr_getinherited()* function shall store the *inheritance* attribute value in the object pointed to by *inheritancepolicy*. Otherwise, the content of this object is undefined.

If successful, the *posix_trace_attr_getlogfullpolicy()* function shall store the *log-full-policy* attribute value in the object pointed to by *logpolicy*. Otherwise, the content of this object is undefined.

If successful, the *posix_trace_attr_getstreamfullpolicy()* function shall store the *stream-full-policy* attribute value in the object pointed to by *streampolicy*. Otherwise, the content of this object is undefined.

ERRORS

These functions may fail if:

[EINVAL] The value specified by at least one of the arguments is invalid.

EXAMPLES

None.

APPLICATION USAGE

None.

RATIONALE

None.

FUTURE DIRECTIONS

The following functions:

posix_trace_attr_getinherited()
posix_trace_attr_getlogfullpolicy()
posix_trace_attr_getstreamfullpolicy()
posix_trace_attr_setinherited()
posix_trace_attr_setlogfullpolicy()
posix_trace_attr_setstreamfullpolicy()

may be removed in a future version.

SEE ALSO

fork(), *posix_trace_attr_destroy()*, *posix_trace_create()*, *posix_trace_get_attr()*,
posix_trace_getnext_event(), *posix_trace_start()*

XBD <trace.h>

CHANGE HISTORY

First released in Issue 6. Derived from IEEE Std 1003.1q-2000.

IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/39 is applied, adding the TRL and TRC margin codes to the *posix_trace_attr_setlogfullpolicy()* function.

Issue 7

SD5-XSH-ERN-116 is applied, adding the missing **restrict** keyword to the *posix_trace_attr_getstreamfullpolicy()* function declaration.

These functions are marked obsolescent.

NAME

posix_trace_attr_getlogsize, posix_trace_attr_getmaxdatasize, posix_trace_attr_getmaxsystemeventsize, posix_trace_attr_getmaxusereventsize, posix_trace_attr_getstreamsize, posix_trace_attr_setlogsize, posix_trace_attr_setmaxdatasize, posix_trace_attr_setstreamsize — retrieve and set trace stream size attributes (TRACING)

SYNOPSIS

```
#include <sys/types.h>
#include <trace.h>

int posix_trace_attr_getlogsize(const trace_attr_t *restrict attr,
    size_t *restrict logsize);

int posix_trace_attr_getmaxdatasize(const trace_attr_t *restrict attr,
    size_t *restrict maxdatasize);

int posix_trace_attr_getmaxsystemeventsize(
    const trace_attr_t *restrict attr,
    size_t *restrict eventsize);

int posix_trace_attr_getmaxusereventsize(
    const trace_attr_t *restrict attr,
    size_t data_len, size_t *restrict eventsize);

int posix_trace_attr_getstreamsize(const trace_attr_t *restrict attr,
    size_t *restrict streamsize);

int posix_trace_attr_setlogsize(trace_attr_t *attr,
    size_t logsize);

int posix_trace_attr_setmaxdatasize(trace_attr_t *attr,
    size_t maxdatasize);

int posix_trace_attr_setstreamsize(trace_attr_t *attr,
    size_t streamsize);
```

DESCRIPTION

The *posix_trace_attr_getlogsize()* function shall copy the log size, in bytes, from the *log-max-size* attribute of the attributes object pointed to by the *attr* argument into the variable pointed to by the *logsize* argument. This log size is the maximum total of bytes that shall be allocated for system and user trace events in the trace log. The default value for the *log-max-size* attribute is implementation-defined.

The *posix_trace_attr_setlogsize()* function shall set the maximum allowed size, in bytes, in the *log-max-size* attribute of the attributes object pointed to by the *attr* argument, using the size value supplied by the *logsize* argument.

The trace log size shall be used if the *log-full-policy* attribute is set to *POSIX_TRACE_LOOP* or *POSIX_TRACE_UNTIL_FULL*. If the *log-full-policy* attribute is set to *POSIX_TRACE_APPEND*, the implementation shall ignore the *log-max-size* attribute.

The *posix_trace_attr_getmaxdatasize()* function shall copy the maximum user trace event data size, in bytes, from the *max-data-size* attribute of the attributes object pointed to by the *attr* argument into the variable pointed to by the *maxdatasize* argument. The default value for the *max-data-size* attribute is implementation-defined.

The *posix_trace_attr_getmaxsystemeventsize()* function shall calculate the maximum memory size, in bytes, required to store a single system trace event. This value is calculated for the trace stream attributes object pointed to by the *attr* argument and is returned in the variable pointed to by the *eventsize* argument.

The values returned as the maximum memory sizes of the user and system trace events shall be such that if the sum of the maximum memory sizes of a set of the trace events that may be

recorded in a trace stream is less than or equal to the *stream-min-size* attribute of that trace stream, the system provides the necessary resources for recording all those trace events, without loss.

The *posix_trace_attr_getmaxusereventsize()* function shall calculate the maximum memory size, in bytes, required to store a single user trace event generated by a call to *posix_trace_event()* with a *data_len* parameter equal to the *data_len* value specified in this call. This value is calculated for the trace stream attributes object pointed to by the *attr* argument and is returned in the variable pointed to by the *eventsize* argument.

The *posix_trace_attr_getstreamsize()* function shall copy the stream size, in bytes, from the *stream-min-size* attribute of the attributes object pointed to by the *attr* argument into the variable pointed to by the *streamsize* argument.

This stream size is the current total memory size reserved for system and user trace events in the trace stream. The default value for the *stream-min-size* attribute is implementation-defined. The stream size refers to memory used to store trace event records. Other stream data (for example, trace attribute values) shall not be included in this size.

The *posix_trace_attr_setmaxdatasize()* function shall set the maximum allowed size, in bytes, in the *max-data-size* attribute of the attributes object pointed to by the *attr* argument, using the size value supplied by the *maxdatasize* argument. This maximum size is the maximum allowed size for the user data argument which may be passed to *posix_trace_event()*. The implementation shall be allowed to truncate data passed to *trace_user_event* which is longer than *maxdatasize*.

The *posix_trace_attr_setstreamsize()* function shall set the minimum allowed size, in bytes, in the *stream-min-size* attribute of the attributes object pointed to by the *attr* argument, using the size value supplied by the *streamsize* argument.

RETURN VALUE

Upon successful completion, these functions shall return a value of zero. Otherwise, they shall return the corresponding error number.

TRL The *posix_trace_attr_getlogsize()* function stores the maximum trace log allowed size in the object pointed to by *logsize*, if successful.

The *posix_trace_attr_getmaxdatasize()* function stores the maximum trace event record memory size in the object pointed to by *maxdatasize*, if successful.

The *posix_trace_attr_getmaxsystemeventsize()* function stores the maximum memory size to store a single system trace event in the object pointed to by *eventsize*, if successful.

The *posix_trace_attr_getmaxusereventsize()* function stores the maximum memory size to store a single user trace event in the object pointed to by *eventsize*, if successful.

The *posix_trace_attr_getstreamsize()* function stores the maximum trace stream allowed size in the object pointed to by *streamsize*, if successful.

ERRORS

These functions may fail if:

[EINVAL] The value specified by one of the arguments is invalid.

48645 EXAMPLES

48646 None.

48647 APPLICATION USAGE

48648 None.

48649 RATIONALE

48650 None.

48651 FUTURE DIRECTIONS

48652 The following functions:

48653 *posix_trace_attr_getlogsize()*
 48654 *posix_trace_attr_getmaxdatasize()*
 48655 *posix_trace_attr_getmaxsystemeventszsize()*
 48656 *posix_trace_attr_getmaxusereventszsize()*
 48657 *posix_trace_attr_getstreamsize()*
 48658 *posix_trace_attr_setlogsize()*
 48659 *posix_trace_attr_setmaxdatasize()*
 48660 *posix_trace_attr_setstreamsize()*

48661 may be removed in a future version.

48662 SEE ALSO

48663 *posix_trace_attr_destroy()*, *posix_trace_create()*, *posix_trace_event()*, *posix_trace_get_attr()*

48664 XBD <sys/types.h>, <trace.h>

48665 CHANGE HISTORY

48666 First released in Issue 6. Derived from IEEE Std 1003.1q-2000.

48667 Issue 7

48668 These functions are marked obsolescent.

48669 NAME

48670 posix_trace_attr_getname — retrieve and set information about a trace stream (TRACING)

48671 SYNOPSIS

48672 OB TRC #include <trace.h>

```
48673       int posix_trace_attr_getname(const trace_attr_t *attr,  
48674                           char *tracename);
```

48675 DESCRIPTION

48676 Refer to *posix_trace_attr_getclockres()*.

48677 NAME

48678 posix_trace_attr_getstreamfullpolicy — retrieve and set the behavior of a trace stream
48679 (TRACING)

48680 SYNOPSIS

```
48681 OB TRC #include <trace.h>  
48682        int posix_trace_attr_getstreamfullpolicy(const trace_attr_t *restrict  
48683        attr, int *restrict streampolicy);
```

48684 DESCRIPTION

48685 Refer to *posix_trace_attr_getinherited()*.

48686 NAME

48687 posix_trace_attr_getstreamsize — retrieve and set trace stream size attributes (TRACING)

48688 SYNOPSIS

```
48689 OB TRC #include <sys/types.h>
48690         #include <trace.h>
48691
48691         int posix_trace_attr_getstreamsize(const trace_attr_t *restrict attr,
48692         size_t *restrict streamsize);
```

48693 DESCRIPTION

48694 Refer to *posix_trace_attr_getlogsize()*.

48695 NAME

48696 posix_trace_attr_init — initialize the trace stream attributes object (TRACING)

48697 SYNOPSIS

48698 OB TRC `#include <trace.h>`

48699 `int posix_trace_attr_init(trace_attr_t *attr);`

48700 DESCRIPTION

48701 Refer to *posix_trace_attr_destroy()*.

48702 NAME

48703 posix_trace_attr_setinherited, posix_trace_attr_setlogfullpolicy — retrieve and set the behavior
48704 of a trace stream (**TRACING**)

48705 SYNOPSIS

```
48706 OB TRC  #include <trace.h>
48707 TRI      int posix_trace_attr_setinherited(trace_attr_t *attr,
48708      int inheritancepolicy);
48709 TRL      int posix_trace_attr_setlogfullpolicy(trace_attr_t *attr,
48710      int logpolicy);
```

48711 DESCRIPTION

48712 Refer to *posix_trace_attr_getinherited()*.

48713 NAME

48714 posix_trace_attr_setlogsize, posix_trace_attr_setmaxdatasize — retrieve and set trace stream size
48715 attributes (TRACING)

48716 SYNOPSIS

```
48717 OB TRC #include <sys/types.h>  
48718         #include <trace.h>  
  
48719 TRL     int posix_trace_attr_setlogsize(trace_attr_t *attr,  
48720         size_t logsize);  
48721 OB TRC  int posix_trace_attr_setmaxdatasize(trace_attr_t *attr,  
48722         size_t maxdatasize);
```

48723 DESCRIPTION

48724 Refer to [posix_trace_attr_getlogsize\(\)](#).

48725 **NAME**

48726 posix_trace_attr_setname — retrieve and set information about a trace stream (TRACING)

48727 **SYNOPSIS**

48728 OB TRC #include <trace.h>

48729 int posix_trace_attr_setname(trace_attr_t *attr,
48730 const char *tracename);48731 **DESCRIPTION**48732 Refer to *posix_trace_attr_getclockres()*.

48733 NAME

48734 posix_trace_attr_setstreamfullpolicy — retrieve and set the behavior of a trace stream
48735 (TRACING)

48736 SYNOPSIS

```
48737 OB TRC #include <trace.h>  
48738 int posix_trace_attr_setstreamfullpolicy(trace_attr_t *attr,  
48739 int streampolicy);
```

48740 DESCRIPTION

48741 Refer to *posix_trace_attr_getinherited()*.

48742 **NAME**

48743 posix_trace_attr_setstreamsize — retrieve and set trace stream size attributes (TRACING)

48744 **SYNOPSIS**

```
48745 OB TRC #include <sys/types.h>
48746         #include <trace.h>
48747
48747         int posix_trace_attr_setstreamsize(trace_attr_t *attr,
48748         size_t streamsize);
```

48749 **DESCRIPTION**48750 Refer to *posix_trace_attr_getlogsize()*.

NAME

posix_trace_clear — clear trace stream and trace log (TRACING)

SYNOPSIS

```
OB TRC  #include <sys/types.h>
         #include <trace.h>
         int posix_trace_clear(trace_id_t trid);
```

DESCRIPTION

The *posix_trace_clear()* function shall reinitialize the trace stream identified by the argument *trid* as if it were returning from the *posix_trace_create()* function, except that the same allocated resources shall be reused, the mapping of trace event type identifiers to trace event names shall be unchanged, and the trace stream status shall remain unchanged (that is, if it was running, it remains running and if it was suspended, it remains suspended).

All trace events in the trace stream recorded before the call to *posix_trace_clear()* shall be lost. The *posix_stream_full_status* status shall be set to `POSIX_TRACE_NOT_FULL`. There is no guarantee that all trace events that occurred during the *posix_trace_clear()* call are recorded; the behavior with respect to trace points that may occur during this call is unspecified.

OB TRL If the Trace Log option is supported and the trace stream has been created with a log, the *posix_trace_clear()* function shall reinitialize the trace stream with the same behavior as if the trace stream was created without the log, plus it shall reinitialize the trace log associated with the trace stream identified by the argument *trid* as if it were returning from the *posix_trace_create_withlog()* function, except that the same allocated resources, for the trace log, may be reused and the associated trace stream status remains unchanged. The first trace event recorded in the trace log after the call to *posix_trace_clear()* shall be the same as the first trace event recorded in the active trace stream after the call to *posix_trace_clear()*. The *posix_log_full_status* status shall be set to `POSIX_TRACE_NOT_FULL`. There is no guarantee that all trace events that occurred during the *posix_trace_clear()* call are recorded in the trace log; the behavior with respect to trace points that may occur during this call is unspecified. If the log full policy is `POSIX_TRACE_APPEND`, the effect of a call to this function is unspecified for the trace log associated with the trace stream identified by the *trid* argument.

RETURN VALUE

Upon successful completion, the *posix_trace_clear()* function shall return a value of zero. Otherwise, it shall return the corresponding error number.

ERRORS

The *posix_trace_clear()* function shall fail if:

[EINVAL] The value of the *trid* argument does not correspond to an active trace stream.

EXAMPLES

None.

APPLICATION USAGE

None.

RATIONALE

None.

FUTURE DIRECTIONS

The *posix_trace_clear()* function may be removed in a future version.

48794 **SEE ALSO**48795 *posix_trace_attr_destroy()*, *posix_trace_create()*, *posix_trace_get_attr()*

48796 XBD <sys/types.h>, <trace.h>

48797 **CHANGE HISTORY**

48798 First released in Issue 6. Derived from IEEE Std 1003.1q-2000.

48799 IEEE PASC Interpretation 1003.1 #123 is applied.

48800 **Issue 7**48801 The *posix_trace_clear()* function is marked obsolescent.

NAME

posix_trace_close, posix_trace_open, posix_trace_rewind — trace log management (**TRACING**)

SYNOPSIS

```
#include <trace.h>
```

```
int posix_trace_close(trace_id_t trid);
int posix_trace_open(int file_desc, trace_id_t *trid);
int posix_trace_rewind(trace_id_t trid);
```

DESCRIPTION

The *posix_trace_close()* function shall deallocate the trace log identifier indicated by *trid*, and all of its associated resources. If there is no valid trace log pointed to by the *trid*, this function shall fail.

The *posix_trace_open()* function shall allocate the necessary resources and establish the connection between a trace log identified by the *file_desc* argument and a trace stream identifier identified by the object pointed to by the *trid* argument. The *file_desc* argument should be a valid open file descriptor that corresponds to a trace log. The *file_desc* argument shall be open for reading. The current trace event timestamp, which specifies the timestamp of the trace event that will be read by the next call to *posix_trace_getnext_event()*, shall be set to the timestamp of the oldest trace event recorded in the trace log identified by *trid*.

The *posix_trace_open()* function shall return a trace stream identifier in the variable pointed to by the *trid* argument, that may only be used by the following functions:

<i>posix_trace_close()</i>	<i>posix_trace_get_attr()</i>
<i>posix_trace_eventid_equal()</i>	<i>posix_trace_get_status()</i>
<i>posix_trace_eventid_get_name()</i>	<i>posix_trace_getnext_event()</i>
<i>posix_trace_eventtypelist_getnext_id()</i>	<i>posix_trace_rewind()</i>
<i>posix_trace_eventtypelist_rewind()</i>	

In particular, notice that the operations normally used by a trace controller process, such as *posix_trace_start()*, *posix_trace_stop()*, or *posix_trace_shutdown()*, cannot be invoked using the trace stream identifier returned by the *posix_trace_open()* function.

The *posix_trace_rewind()* function shall reset the current trace event timestamp, which specifies the timestamp of the trace event that will be read by the next call to *posix_trace_getnext_event()*, to the timestamp of the oldest trace event recorded in the trace log identified by *trid*.

RETURN VALUE

Upon successful completion, these functions shall return a value of zero. Otherwise, they shall return the corresponding error number.

If successful, the *posix_trace_open()* function stores the trace stream identifier value in the object pointed to by *trid*.

ERRORS

The *posix_trace_open()* function shall fail if:

[EINTR]	The operation was interrupted by a signal and thus no trace log was opened.
[EINVAL]	The object pointed to by <i>file_desc</i> does not correspond to a valid trace log.

48842 The *posix_trace_close()* and *posix_trace_rewind()* functions may fail if:

48843 [EINVAL] The object pointed to by *trid* does not correspond to a valid trace log.

48844 **EXAMPLES**

48845 None.

48846 **APPLICATION USAGE**

48847 None.

48848 **RATIONALE**

48849 None.

48850 **FUTURE DIRECTIONS**

48851 The *posix_trace_close()*, *posix_trace_open()*, and *posix_trace_rewind()* functions may be removed in
48852 a future version.

48853 **SEE ALSO**

48854 *posix_trace_get_attr()*, *posix_trace_get_filter()*, *posix_trace_getnext_event()*

48855 XBD <trace.h>

48856 **CHANGE HISTORY**

48857 First released in Issue 6. Derived from IEEE Std 1003.1q-2000.

48858 IEEE PASC Interpretation 1003.1 #123 is applied.

48859 **Issue 7**

48860 The *posix_trace_close()*, *posix_trace_open()*, and *posix_trace_rewind()* functions are marked
48861 obsolescent.

NAME

posix_trace_create, posix_trace_create_withlog, posix_trace_flush, posix_trace_shutdown — trace stream initialization, flush, and shutdown from a process (TRACING)

SYNOPSIS

```
#include <sys/types.h>
#include <trace.h>

int posix_trace_create(pid_t pid,
    const trace_attr_t *restrict attr,
    trace_id_t *restrict trid);
int posix_trace_create_withlog(pid_t pid,
    const trace_attr_t *restrict attr, int file_desc,
    trace_id_t *restrict trid);
int posix_trace_flush(trace_id_t trid);
int posix_trace_shutdown(trace_id_t trid);
```

DESCRIPTION

The *posix_trace_create()* function shall create an active trace stream. It allocates all the resources needed by the trace stream being created for tracing the process specified by *pid* in accordance with the *attr* argument. The *attr* argument represents the initial attributes of the trace stream and shall have been initialized by the function *posix_trace_attr_init()* prior to the *posix_trace_create()* call. If the argument *attr* is NULL, the default attributes shall be used. The *attr* attributes object shall be manipulated through a set of functions described in the *posix_trace_attr* family of functions. If the attributes of the object pointed to by *attr* are modified later, the attributes of the trace stream shall not be affected. The *creation-time* attribute of the newly created trace stream shall be set to the value of the system clock, if the Timers option is not supported, or to the value of the CLOCK_REALTIME clock, if the Timers option is supported.

The *pid* argument represents the target process to be traced. If the process executing this function does not have appropriate privileges to trace the process identified by *pid*, an error shall be returned. If the *pid* argument is zero, the calling process shall be traced.

The *posix_trace_create()* function shall store the trace stream identifier of the new trace stream in the object pointed to by the *trid* argument. This trace stream identifier shall be used in subsequent calls to control tracing. The *trid* argument may only be used by the following functions:

<i>posix_trace_clear()</i>	<i>posix_trace_getnext_event()</i>
<i>posix_trace_eventid_equal()</i>	<i>posix_trace_shutdown()</i>
<i>posix_trace_eventid_get_name()</i>	<i>posix_trace_start()</i>
<i>posix_trace_eventtypelist_getnext_id()</i>	<i>posix_trace_stop()</i>
<i>posix_trace_eventtypelist_rewind()</i>	<i>posix_trace_timedgetnext_event()</i>
<i>posix_trace_get_attr()</i>	<i>posix_trace_trid_eventid_open()</i>
<i>posix_trace_get_status()</i>	<i>posix_trace_trygetnext_event()</i>

If the Trace Event Filter option is supported, the following additional functions may use the *trid* argument:

posix_trace_get_filter() *posix_trace_set_filter()*

In particular, notice that the operations normally used by a trace analyzer process, such as *posix_trace_rewind()* or *posix_trace_close()*, cannot be invoked using the trace stream identifier returned by the *posix_trace_create()* function.

48907	TEF	A trace stream shall be created in a suspended state.	If the Trace Event Filter option is
48908		supported, its trace event type filter shall be empty.	

The `posix_trace_create()` function may be called multiple times from the same or different processes, with the system-wide limit indicated by the runtime invariant value `{TRACE_SYS_MAX}`, which has the minimum value `{_POSIX_TRACE_SYS_MAX}`.

The trace stream identifier returned by the `posix_trace_create()` function in the argument pointed to by `trid` is valid only in the process that made the function call. If it is used from another process, that is a child process, in functions defined in POSIX.1-2008, these functions shall return with the error [EINVAL].

48916 TRL The *posix_trace_create_withlog()* function shall be equivalent to *posix_trace_create()*, except that it
48917 associates a trace log with this stream. The *file_desc* argument shall be the file descriptor
48918 designating the trace log destination. The function shall fail if this file descriptor refers to a file
48919 with a file type that is not compatible with the log policy associated with the trace log. The list of
48920 the appropriate file types that are compatible with each log policy is implementation-defined.

The `posix_trace_create_withlog()` function shall return in the parameter pointed to by `trid` the trace stream identifier, which uniquely identifies the newly created trace stream, and shall be used in subsequent calls to control tracing. The `trid` argument may only be used by the following functions:

48925	<i>posix_trace_clear()</i>	<i>posix_trace_get_status()</i>
48926	<i>posix_trace_eventid_equal()</i>	<i>posix_trace_getnext_event()</i>
48927	<i>posix_trace_eventid_get_name()</i>	<i>posix_trace_shutdown()</i>
48928	<i>posix_trace_eventtypelist_getnext_id()</i>	<i>posix_trace_start()</i>
48929	<i>posix_trace_eventtypelist_rewind()</i>	<i>posix_trace_stop()</i>
48930	<i>posix_trace_flush()</i>	<i>posix_trace_timedgetnext_event()</i>
48931	<i>posix_trace_get_attr()</i>	<i>posix_trace_trid_eventid_open()</i>

48932	TEF	TRL	If the Trace Event Filter option is supported, the following additional functions may use the <i>trid</i> argument:
48933			

48934 *posix_trace_get_filter()* *posix_trace_set_filter()*

In particular, notice that the operations normally used by a trace analyzer process, such as `posix_trace_rewind()` or `posix_trace_close()`, cannot be invoked using the trace stream identifier returned by the `posix_trace_create_withlog()` function.

The *posix_trace_flush()* function shall initiate a flush operation which copies the contents of the trace stream identified by the argument *trid* into the trace log associated with the trace stream at the creation time. If no trace log has been associated with the trace stream pointed to by *trid*, this function shall return an error. The termination of the flush operation can be polled by the *posix_trace_get_status()* function. During the flush operation, it shall be possible to trace new trace events up to the point when the trace stream becomes full. After flushing is completed, the space used by the flushed trace events shall be available for tracing new trace events.

48945 If flushing the trace stream causes the resulting trace log to become full, the trace log full policy
48946 shall be applied. If the trace *log-full-policy* attribute is set, the following occurs:

48947 POSIX TRACE UNTIL FULL

48948 The trace events that have not yet been flushed shall be discarded.

48949 POSIX_TRACE_LOOP
 48950 The trace events that have not yet been flushed shall be written to the beginning of the trace
 48951 log, overwriting previous trace events stored there.

48952 POSIX_TRACE_APPEND
 48953 The trace events that have not yet been flushed shall be appended to the trace log.

48954 The *posix_trace_shutdown()* function shall stop the tracing of trace events in the trace stream
 48955 identified by *trid*, as if *posix_trace_stop()* had been invoked. The *posix_trace_shutdown()* function
 48956 shall free all the resources associated with the trace stream.

48957 The *posix_trace_shutdown()* function shall not return until all the resources associated with the
 48958 trace stream have been freed. When the *posix_trace_shutdown()* function returns, the *trid*
 48959 argument becomes an invalid trace stream identifier. A call to this function shall unconditionally
 48960 deallocate the resources regardless of whether all trace events have been retrieved by the
 48961 analyzer process. Any thread blocked on one of the *trace_getnext_event()* functions (which
 48962 specified this *trid*) before this call is unblocked with the error [EINVAL].

48963 If the process exits, invokes a member of the *exec* family of functions, or is terminated, the trace
 48964 streams that the process had created and that have not yet been shut down, shall be
 48965 automatically shut down as if an explicit call were made to the *posix_trace_shutdown()* function.

48966 TRL For an active trace stream with log, when the *posix_trace_shutdown()* function is called, all trace
 48967 events that have not yet been flushed to the trace log shall be flushed, as in the
 48968 *posix_trace_flush()* function, and the trace log shall be closed.

48969 When a trace log is closed, all the information that may be retrieved later from the trace log
 48970 through the trace interface shall have been written to the trace log. This information includes the
 48971 trace attributes, the list of trace event types (with the mapping between trace event names and
 48972 trace event type identifiers), and the trace status.

48973 In addition, unspecified information shall be written to the trace log to allow detection of a valid
 48974 trace log during the *posix_trace_open()* operation.

48975 The *posix_trace_shutdown()* function shall not return until all trace events have been flushed.

48976 RETURN VALUE

48977 Upon successful completion, these functions shall return a value of zero. Otherwise, they shall
 48978 return the corresponding error number.

48979 TRL The *posix_trace_create()* and *posix_trace_create_withlog()* functions store the trace stream
 48980 identifier value in the object pointed to by *trid*, if successful.

48981 ERRORS

48982 TRL The *posix_trace_create()* and *posix_trace_create_withlog()* functions shall fail if:

48983 [EAGAIN] No more trace streams can be started now. {TRACE_SYS_MAX} has been
 48984 exceeded.

48985 [EINTR] The operation was interrupted by a signal. No trace stream was created.

48986 [EINVAL] One or more of the trace parameters specified by the *attr* parameter is invalid.

48987 [ENOMEM] The implementation does not currently have sufficient memory to create the
 48988 trace stream with the specified parameters.

48989 [EPERM] The caller does not have appropriate privileges to trace the process specified
 48990 by *pid*.

48991 [ESRCH] The *pid* argument does not refer to an existing process.

48992 TRL The *posix_trace_create_withlog()* function shall fail if:

48993 [EBADF] The *file_desc* argument is not a valid file descriptor open for writing.

48994 [EINVAL] The *file_desc* argument refers to a file with a file type that does not support the

48995 log policy associated with the trace log.

48996 [ENOSPC] No space left on device. The device corresponding to the argument *file_desc*

48997 does not contain the space required to create this trace log.

48998 TRL The *posix_trace_flush()* and *posix_trace_shutdown()* functions shall fail if:

48999 [EINVAL] The value of the *trid* argument does not correspond to an active trace stream

49000 with log.

49001 [EFBIG] The trace log file has attempted to exceed an implementation-defined

49002 maximum file size.

49003 [ENOSPC] No space left on device.

49004 EXAMPLES

49005 None.

49006 APPLICATION USAGE

49007 None.

49008 RATIONALE

49009 None.

49010 FUTURE DIRECTIONS

49011 The *posix_trace_create()*, *posix_trace_create_withlog()*, *posix_trace_flush()*, and

49012 *posix_trace_shutdown()* functions may be removed in a future version.

49013 SEE ALSO

49014 *clock_getres()*, *exec*, *posix_trace_attr_destroy()*, *posix_trace_clear()*, *posix_trace_close()*,

49015 *posix_trace_eventid_equal()*, *posix_trace_eventtypelist_getnext_id()*, *posix_trace_get_attr()*,

49016 *posix_trace_get_filter()*, *posix_trace_getnext_event()*, *posix_trace_start()*, *posix_trace_start()*, *time()*

49017 XBD <sys/types.h>, <trace.h>

49018 CHANGE HISTORY

49019 First released in Issue 6. Derived from IEEE Std 1003.1q-2000.

49020 Issue 7

49021 These functions are marked obsolescent.

49022 SD5-XSH-ERN-154 is applied, updating the DESCRIPTION to remove the

49023 *posix_trace_trygetnext_event()* function from the list of functions that use the *trid* argument.

49024 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0441 [358] is applied.

+

NAME

posix_trace_event, posix_trace_eventid_open — trace functions for instrumenting application code (TRACING)

SYNOPSIS

```
OB TRC #include <sys/types.h>
#include <trace.h>

void posix_trace_event(trace_event_id_t event_id,
    const void *restrict data_ptr, size_t data_len);
int posix_trace_eventid_open(const char *restrict event_name,
    trace_event_id_t *restrict event_id);
```

DESCRIPTION

The *posix_trace_event()* function shall record the *event_id* and the user data pointed to by *data_ptr* in the trace stream into which the calling process is being traced and in which *event_id* is not filtered out. If the total size of the user trace event data represented by *data_len* is not greater than the declared maximum size for user trace event data, then the *truncation-status* attribute of the trace event recorded is POSIX_TRACE_NOT_TRUNCATED. Otherwise, the user trace event data is truncated to this declared maximum size and the *truncation-status* attribute of the trace event recorded is POSIX_TRACE_TRUNCATED_RECORD.

If there is no trace stream created for the process or if the created trace stream is not running, or if the trace event specified by *event_id* is filtered out in the trace stream, the *posix_trace_event()* function shall have no effect.

The *posix_trace_eventid_open()* function shall associate a user trace event name with a trace event type identifier for the calling process. The trace event name is the string pointed to by the argument *event_name*. It shall have a maximum of {TRACE_EVENT_NAME_MAX} characters (which has the minimum value {POSIX_TRACE_EVENT_NAME_MAX}). The number of user trace event type identifiers that can be defined for any given process is limited by the maximum value {TRACE_USER_EVENT_MAX}, which has the minimum value {POSIX_TRACE_USER_EVENT_MAX}.

If the Trace Inherit option is not supported, the *posix_trace_eventid_open()* function shall associate the user trace event name pointed to by the *event_name* argument with a trace event type identifier that is unique for the traced process, and is returned in the variable pointed to by the *event_id* argument. If the user trace event name has already been mapped for the traced process, then the previously assigned trace event type identifier shall be returned. If the per-process user trace event name limit represented by {TRACE_USER_EVENT_MAX} has been reached, the pre-defined POSIX_TRACE_UNNAMED_USEREVENT (see Table 2-7, on page 544) user trace event shall be returned.

TRI

If the Trace Inherit option is supported, the *posix_trace_eventid_open()* function shall associate the user trace event name pointed to by the *event_name* argument with a trace event type identifier that is unique for all the processes being traced in this same trace stream, and is returned in the variable pointed to by the *event_id* argument. If the user trace event name has already been mapped for the traced processes, then the previously assigned trace event type identifier shall be returned. If the per-process user trace event name limit represented by {TRACE_USER_EVENT_MAX} has been reached, the pre-defined POSIX_TRACE_UNNAMED_USEREVENT (Table 2-7, on page 544) user trace event shall be returned.

Note: The above procedure, together with the fact that multiple processes can only be traced into the same trace stream by inheritance, ensure that all the processes that are traced into a trace stream have the same mapping of trace event names to trace event type identifiers.

49073 If there is no trace stream created, the *posix_trace_eventid_open()* function shall store this
49074 information for future trace streams created for this process.

49075 RETURN VALUE

49076 No return value is defined for the *posix_trace_event()* function.

49077 Upon successful completion, the *posix_trace_eventid_open()* function shall return a value of zero.
49078 Otherwise, it shall return the corresponding error number. The *posix_trace_eventid_open()*
49079 function stores the trace event type identifier value in the object pointed to by *event_id*, if
49080 successful.

49081 ERRORS

49082 The *posix_trace_eventid_open()* function shall fail if:

49083 [ENAMETOOLONG]

49084 The size of the name pointed to by the *event_name* argument was longer than
49085 the implementation-defined value {TRACE_EVENT_NAME_MAX}.

49086 EXAMPLES

49087 None.

49088 APPLICATION USAGE

49089 None.

49090 RATIONALE

49091 None.

49092 FUTURE DIRECTIONS

49093 The *posix_trace_event()* and *posix_trace_eventid_open()* functions may be removed in a future
49094 version.

49095 SEE ALSO

49096 [Table 2-7](#) (on page 544), *exec*, *posix_trace_eventid_equal()*, *posix_trace_start()*

49097 XBD [<sys/types.h>](#), [<trace.h>](#)

49098 CHANGE HISTORY

49099 First released in Issue 6. Derived from IEEE Std 1003.1q-2000.

49100 IEEE PASC Interpretation 1003.1 #123 is applied.

49101 IEEE PASC Interpretation 1003.1 #127 is applied, correcting some editorial errors in the names of
49102 the *posix_trace_eventid_open()* function and the *event_id* argument.

49103 Issue 7

49104 The *posix_trace_event()* and *posix_trace_eventid_open()* functions are marked obsolescent.

NAME

posix_trace_eventid_equal, posix_trace_eventid_get_name, posix_trace_trid_eventid_open — manipulate the trace event type identifier (TRACING)

SYNOPSIS

```
OB TRC  #include <trace.h>

int posix_trace_eventid_equal(trace_id_t trid, trace_event_id_t event1,
    trace_event_id_t event2);
int posix_trace_eventid_get_name(trace_id_t trid,
    trace_event_id_t event, char *event_name);
TEF int posix_trace_trid_eventid_open(trace_id_t trid,
    const char *restrict event_name,
    trace_event_id_t *restrict event);
```

DESCRIPTION

The *posix_trace_eventid_equal()* function shall compare the trace event type identifiers *event1* and *event2* from the same trace stream or the same trace log identified by the *trid* argument. If the trace event type identifiers *event1* and *event2* are from different trace streams, the return value shall be unspecified.

The *posix_trace_eventid_get_name()* function shall return, in the argument pointed to by *event_name*, the trace event name associated with the trace event type identifier identified by the argument *event*, for the trace stream or for the trace log identified by the *trid* argument. The name of the trace event shall have a maximum of {TRACE_EVENT_NAME_MAX} characters (which has the minimum value {_POSIX_TRACE_EVENT_NAME_MAX}). Successive calls to this function with the same trace event type identifier and the same trace stream identifier shall return the same event name.

TEF The *posix_trace_trid_eventid_open()* function shall associate a user trace event name with a trace event type identifier for a given trace stream. The trace stream is identified by the *trid* argument, and it shall be an active trace stream. The trace event name is the string pointed to by the argument *event_name*. It shall have a maximum of {TRACE_EVENT_NAME_MAX} characters (which has the minimum value {_POSIX_TRACE_EVENT_NAME_MAX}). The number of user trace event type identifiers that can be defined for any given process is limited by the maximum value {TRACE_USER_EVENT_MAX}, which has the minimum value {_POSIX_TRACE_USER_EVENT_MAX}.

If the Trace Inherit option is not supported, the *posix_trace_trid_eventid_open()* function shall associate the user trace event name pointed to by the *event_name* argument with a trace event type identifier that is unique for the process being traced in the trace stream identified by the *trid* argument, and is returned in the variable pointed to by the *event* argument. If the user trace event name has already been mapped for the traced process, then the previously assigned trace event type identifier shall be returned. If the per-process user trace event name limit represented by {TRACE_USER_EVENT_MAX} has been reached, the pre-defined POSIX_TRACE_UNNAMED_USEREVENT (see [Table 2-7](#), on page 544) user trace event shall be returned.

TEF TRI If the Trace Inherit option is supported, the *posix_trace_trid_eventid_open()* function shall associate the user trace event name pointed to by the *event_name* argument with a trace event type identifier that is unique for all the processes being traced in the trace stream identified by the *trid* argument, and is returned in the variable pointed to by the *event* argument. If the user trace event name has already been mapped for the traced processes, then the previously assigned trace event type identifier shall be returned. If the per-process user trace event name limit represented by {TRACE_USER_EVENT_MAX} has been reached, the pre-defined

49153 POSIX_TRACE_UNNAMED_USEREVENT (see Table 2-7, on page 544) user trace event shall be
 49154 returned.

49155 RETURN VALUE

49156 TEF Upon successful completion, the `posix_trace_eventid_get_name()` and
 49157 `posix_trace_trid_eventid_open()` functions shall return a value of zero. Otherwise, they shall return
 49158 the corresponding error number.

49159 The `posix_trace_eventid_equal()` function shall return a non-zero value if *event1* and *event2* are
 49160 equal; otherwise, a value of zero shall be returned. No errors are defined. If either *event1* or
 49161 *event2* are not valid trace event type identifiers for the trace stream specified by *trid* or if the *trid*
 49162 is invalid, the behavior shall be unspecified.

49163 The `posix_trace_eventid_get_name()` function stores the trace event name value in the object
 49164 pointed to by *event_name*, if successful.

49165 TEF The `posix_trace_trid_eventid_open()` function stores the trace event type identifier value in the
 49166 object pointed to by *event*, if successful.

49167 ERRORS

49168 TEF The `posix_trace_eventid_get_name()` and `posix_trace_trid_eventid_open()` functions shall fail if:

49169 [EINVAL] The *trid* argument was not a valid trace stream identifier.

49170 TEF The `posix_trace_trid_eventid_open()` function shall fail if:

49171 TEF [ENAMETOOLONG]

49172 The size of the name pointed to by the *event_name* argument was longer than
 49173 the implementation-defined value {TRACE_EVENT_NAME_MAX}.

49174 The `posix_trace_eventid_get_name()` function shall fail if:

49175 [EINVAL] The trace event type identifier *event* was not associated with any name.

49176 EXAMPLES

49177 None.

49178 APPLICATION USAGE

49179 None.

49180 RATIONALE

49181 None.

49182 FUTURE DIRECTIONS

49183 The `posix_trace_eventid_equal()`, `posix_trace_eventid_get_name()`, and
 49184 `posix_trace_trid_eventid_open()` functions may be removed in a future version.

49185 SEE ALSO

49186 Table 2-7 (on page 544), `exec`, `posix_trace_event()`, `posix_trace_getnext_event()`

49187 XBD <trace.h>

49188 CHANGE HISTORY

49189 First released in Issue 6. Derived from IEEE Std 1003.1q-2000.

49190 IEEE PASC Interpretations 1003.1 #123 and #129 are applied.

49191 Issue 7

49192 These functions are marked obsolescent.

NAME

posix_trace_eventid_open — trace functions for instrumenting application code (**TRACING**)

SYNOPSIS

```
OB TRC #include <sys/types.h>
#include <trace.h>

int posix_trace_eventid_open(const char *restrict event_name,
    trace_event_id_t *restrict event_id);
```

DESCRIPTION

Refer to *posix_trace_event()*.

NAME

posix_trace_eventset_add, posix_trace_eventset_del, posix_trace_eventset_empty,
 posix_trace_eventset_fill, posix_trace_eventset_ismember — manipulate trace event type sets
 (TRACING)

SYNOPSIS

```
#include <trace.h>

int posix_trace_eventset_add(trace_event_id_t event_id,
    trace_event_set_t *set);
int posix_trace_eventset_del(trace_event_id_t event_id,
    trace_event_set_t *set);
int posix_trace_eventset_empty(trace_event_set_t *set);
int posix_trace_eventset_fill(trace_event_set_t *set, int what);
int posix_trace_eventset_ismember(trace_event_id_t event_id,
    const trace_event_set_t *restrict set, int *restrict ismember);
```

DESCRIPTION

These primitives manipulate sets of trace event types. They operate on data objects addressable by the application, not on the current trace event filter of any trace stream.

The *posix_trace_eventset_add()* and *posix_trace_eventset_del()* functions, respectively, shall add or delete the individual trace event type specified by the value of the argument *event_id* to or from the trace event type set pointed to by the argument *set*. Adding a trace event type already in the set or deleting a trace event type not in the set shall not be considered an error.

The *posix_trace_eventset_empty()* function shall initialize the trace event type set pointed to by the *set* argument such that all trace event types defined, both system and user, shall be excluded from the set.

The *posix_trace_eventset_fill()* function shall initialize the trace event type set pointed to by the argument *set*, such that the set of trace event types defined by the argument *what* shall be included in the set. The value of the argument *what* shall consist of one of the following values, as defined in the **<trace.h>** header:

POSIX_TRACE_WOPID_EVENTS

All the process-independent implementation-defined system trace event types are included in the set.

POSIX_TRACE_SYSTEM_EVENTS

All the implementation-defined system trace event types are included in the set, as are those defined in POSIX.1-2008.

POSIX_TRACE_ALL_EVENTS

All trace event types defined, both system and user, are included in the set.

Applications shall call either *posix_trace_eventset_empty()* or *posix_trace_eventset_fill()* at least once for each object of type **trace_event_set_t** prior to any other use of that object. If such an object is not initialized in this way, but is nonetheless supplied as an argument to any of the *posix_trace_eventset_add()*, *posix_trace_eventset_del()*, or *posix_trace_eventset_ismember()* functions, the results are undefined.

The *posix_trace_eventset_ismember()* function shall test whether the trace event type specified by the value of the argument *event_id* is a member of the set pointed to by the argument *set*. The value returned in the object pointed to by *ismember* argument is zero if the trace event type identifier is not a member of the set and a value different from zero if it is a member of the set.

49247 RETURN VALUE

49248 Upon successful completion, these functions shall return a value of zero. Otherwise, they shall
49249 return the corresponding error number.

49250 ERRORS

49251 These functions may fail if:

49252 [EINVAL] The value of one of the arguments is invalid.

49253 EXAMPLES

49254 None.

49255 APPLICATION USAGE

49256 None.

49257 RATIONALE

49258 None.

49259 FUTURE DIRECTIONS

49260 The `posix_trace_eventset_add()`, `posix_trace_eventset_del()`, `posix_trace_eventset_empty()`,
49261 `posix_trace_eventset_fill()`, and `posix_trace_eventset_ismember()` functions may be removed in a
49262 future version.

49263 SEE ALSO

49264 [*posix_trace_eventid_equal\(\)*](#), [*posix_trace_get_filter\(\)*](#)

49265 XBD [**<trace.h>**](#)

49266 CHANGE HISTORY

49267 First released in Issue 6. Derived from IEEE Std 1003.1q-2000.

49268 Issue 7

49269 The `posix_trace_eventset_add()`, `posix_trace_eventset_del()`, `posix_trace_eventset_empty()`,
49270 `posix_trace_eventset_fill()`, and `posix_trace_eventset_ismember()` functions are marked obsolescent.

49271 **NAME**

49272 `posix_trace_eventtypelist_getnext_id`, `posix_trace_eventtypelist_rewind` — iterate over a
 49273 mapping of trace event types (**TRACING**)

49274 **SYNOPSIS**

```
49275 OB TRC #include <trace.h>
49276
49276     int posix_trace_eventtypelist_getnext_id(trace_id_t trid,
49277         trace_event_id_t *restrict event, int *restrict unavailable);
49278     int posix_trace_eventtypelist_rewind(trace_id_t trid);
```

49279 **DESCRIPTION**

49280 The first time `posix_trace_eventtypelist_getnext_id()` is called, the function shall return in the
 49281 variable pointed to by *event* the first trace event type identifier of the list of trace events of the
 49282 trace stream identified by the *trid* argument. Successive calls to
 49283 `posix_trace_eventtypelist_getnext_id()` return in the variable pointed to by *event* the next trace
 49284 event type identifier in that same list. Each time a trace event type identifier is successfully
 49285 written into the variable pointed to by the *event* argument, the variable pointed to by the
 49286 *unavailable* argument shall be set to zero. When no more trace event type identifiers are available,
 49287 and so none is returned, the variable pointed to by the *unavailable* argument shall be set to a
 49288 value different from zero.

49289 The `posix_trace_eventtypelist_rewind()` function shall reset the next trace event type identifier to
 49290 be read to the first trace event type identifier from the list of trace events used in the trace stream
 49291 identified by *trid*.

49292 **RETURN VALUE**

49293 Upon successful completion, these functions shall return a value of zero. Otherwise, they shall
 49294 return the corresponding error number.

49295 The `posix_trace_eventtypelist_getnext_id()` function stores the trace event type identifier value in
 49296 the object pointed to by *event*, if successful.

49297 **ERRORS**

49298 These functions shall fail if:

49299 [EINVAL] The *trid* argument was not a valid trace stream identifier.

49300 **EXAMPLES**

49301 None.

49302 **APPLICATION USAGE**

49303 None.

49304 **RATIONALE**

49305 None.

49306 **FUTURE DIRECTIONS**

49307 The `posix_trace_eventtypelist_getnext_id()` and `posix_trace_eventtypelist_rewind()` functions may be
 49308 removed in a future version.

49309 **SEE ALSO**

49310 [*posix_trace_event\(\)*](#), [*posix_trace_eventid_equal\(\)*](#), [*posix_trace_getnext_event\(\)*](#)

49311 XBD [*<trace.h>*](#)

CHANGE HISTORY

- 49312 First released in Issue 6. Derived from IEEE Std 1003.1q-2000.
49313
49314 IEEE PASC Interpretations 1003.1 #123 and #129 are applied.

Issue 7

- 49315 The *posix_trace_eventtypelist_getnext_id()* and *posix_trace_eventtypelist_rewind()* functions are
49316 marked obsolescent.
49317

49318 **NAME**49319 posix_trace_flush — trace stream flush from a process (**TRACING**)49320 **SYNOPSIS**

49321 OB TRC #include <sys/types.h>

49322 #include <trace.h>

49323 TRL int posix_trace_flush(trace_id_t *trid*);49324 **DESCRIPTION**49325 Refer to *posix_trace_create()*.

NAME

posix_trace_get_attr, posix_trace_get_status — retrieve the trace attributes or trace status (TRACING)

SYNOPSIS

```
#include <trace.h>

int posix_trace_get_attr(trace_id_t trid, trace_attr_t *attr);
int posix_trace_get_status(trace_id_t trid,
    struct posix_trace_status_info *statusinfo);
```

DESCRIPTION

The *posix_trace_get_attr()* function shall copy the attributes of the active trace stream identified by *trid* into the object pointed to by the *attr* argument. If the Trace Log option is supported, *trid* may represent a pre-recorded trace log.

The *posix_trace_get_status()* function shall return, in the structure pointed to by the *statusinfo* argument, the current trace status for the trace stream identified by the *trid* argument. These status values returned in the structure pointed to by *statusinfo* shall have been appropriately read to ensure that the returned values are consistent. If the Trace Log option is supported and the *trid* argument refers to a pre-recorded trace stream, the status shall be the status of the completed trace stream.

Each time the *posix_trace_get_status()* function is used, the overrun status of the trace stream shall be reset to POSIX_TRACE_NO_OVERRUN immediately after the call completes. If the Trace Log option is supported, the *posix_trace_get_status()* function shall behave the same as when the option is not supported except for the following differences:

- If the *trid* argument refers to a trace stream with log, each time the *posix_trace_get_status()* function is used, the log overrun status of the trace stream shall be reset to POSIX_TRACE_NO_OVERRUN and the *flush_error* status shall be reset to zero immediately after the call completes.
- If the *trid* argument refers to a pre-recorded trace stream, the status returned shall be the status of the completed trace stream and the status values of the trace stream shall not be reset.

RETURN VALUE

Upon successful completion, these functions shall return a value of zero. Otherwise, they shall return the corresponding error number.

The *posix_trace_get_attr()* function stores the trace attributes in the object pointed to by *attr*, if successful.

The *posix_trace_get_status()* function stores the trace status in the object pointed to by *statusinfo*, if successful.

ERRORS

These functions shall fail if:

[EINVAL]	The trace stream argument <i>trid</i> does not correspond to a valid active trace stream or a valid trace log.

49366 EXAMPLES

49367 None.

49368 APPLICATION USAGE

49369 None.

49370 RATIONALE

49371 None.

49372 FUTURE DIRECTIONS

49373 The *posix_trace_get_attr()* and *posix_trace_get_status()* functions may be removed in a future
49374 version.

49375 SEE ALSO

49376 *posix_trace_attr_destroy()*, *posix_trace_close()*, *posix_trace_create()*

49377 XBD <trace.h>

49378 CHANGE HISTORY

49379 First released in Issue 6. Derived from IEEE Std 1003.1q-2000.

49380 IEEE PASC Interpretation 1003.1 #123 is applied.

49381 Issue 7

49382 The *posix_trace_get_attr()* and *posix_trace_get_status()* functions are marked obsolescent.

NAME

posix_trace_get_filter, posix_trace_set_filter — retrieve and set the filter of an initialized trace stream (TRACING)

SYNOPSIS

```
OB TRC #include <trace.h>

TEF int posix_trace_get_filter(trace_id_t trid, trace_event_set_t *set);
int posix_trace_set_filter(trace_id_t trid,
    const trace_event_set_t *set, int how);
```

DESCRIPTION

The *posix_trace_get_filter()* function shall retrieve, into the argument pointed to by *set*, the actual trace event filter from the trace stream specified by *trid*.

The *posix_trace_set_filter()* function shall change the set of filtered trace event types after a trace stream identified by the *trid* argument is created. This function may be called prior to starting the trace stream, or while the trace stream is active. By default, if no call is made to *posix_trace_set_filter()*, all trace events shall be recorded (that is, none of the trace event types are filtered out).

If this function is called while the trace is in progress, a special system trace event, POSIX_TRACE_FILTER, shall be recorded in the trace indicating both the old and the new sets of filtered trace event types (see Table 2-4 (on page 542) and Table 2-6, on page 543).

If the *posix_trace_set_filter()* function is interrupted by a signal, an error shall be returned and the filter shall not be changed. In this case, the state of the trace stream shall not be changed.

The value of the argument *how* indicates the manner in which the set is to be changed and shall have one of the following values, as defined in the **<trace.h>** header:

POSIX_TRACE_SET_EVENTSET

The resulting set of trace event types to be filtered shall be the trace event type set pointed to by the argument *set*.

POSIX_TRACE_ADD_EVENTSET

The resulting set of trace event types to be filtered shall be the union of the current set and the trace event type set pointed to by the argument *set*.

POSIX_TRACE_SUB_EVENTSET

The resulting set of trace event types to be filtered shall be all trace event types in the current set that are not in the set pointed to by the argument *set*; that is, remove each element of the specified set from the current filter.

RETURN VALUE

Upon successful completion, these functions shall return a value of zero. Otherwise, they shall return the corresponding error number.

The *posix_trace_get_filter()* function stores the set of filtered trace event types in *set*, if successful.

ERRORS

These functions shall fail if:

- [EINVAL] The value of the *trid* argument does not correspond to an active trace stream or the value of the argument pointed to by *set* is invalid.
- [EINTR] The operation was interrupted by a signal.

49425 EXAMPLES

49426 None.

49427 APPLICATION USAGE

49428 None.

49429 RATIONALE

49430 None.

49431 FUTURE DIRECTIONS

49432 The *posix_trace_get_filter()* and *posix_trace_set_filter()* functions may be removed in a future
49433 version.

49434 SEE ALSO

49435 [Table 2-4](#) (on page 542), [Table 2-6](#) (on page 543), [posix_trace_eventset_add\(\)](#)

49436 XBD [<trace.h>](#)

49437 CHANGE HISTORY

49438 First released in Issue 6. Derived from IEEE Std 1003.1q-2000.

49439 IEEE PASC Interpretation 1003.1 #123 is applied.

49440 Issue 7

49441 The *posix_trace_get_filter()* and *posix_trace_set_filter()* functions are marked obsolescent.

49442 NAME

49443 posix_trace_get_status — retrieve the trace status (TRACING)

49444 SYNOPSIS

```
49445 OB TRC #include <trace.h>
49446 int posix_trace_get_status(trace_id_t trid,
49447 struct posix_trace_status_info *statusinfo);
```

49448 DESCRIPTION

49449 Refer to *posix_trace_get_attr()*.

NAME

posix_trace_getnext_event, posix_trace_timedgetnext_event, posix_trace_trygetnext_event — retrieve a trace event (**TRACING**)

SYNOPSIS

```
#include <sys/types.h>
#include <trace.h>

int posix_trace_getnext_event(trace_id_t trid,
    struct posix_trace_event_info *restrict event,
    void *restrict data, size_t num_bytes,
    size_t *restrict data_len, int *restrict unavailable);
int posix_trace_timedgetnext_event(trace_id_t trid,
    struct posix_trace_event_info *restrict event,
    void *restrict data, size_t num_bytes,
    size_t *restrict data_len, int *restrict unavailable,
    const struct timespec *restrict abstime);
int posix_trace_trygetnext_event(trace_id_t trid,
    struct posix_trace_event_info *restrict event,
    void *restrict data, size_t num_bytes,
    size_t *restrict data_len, int *restrict unavailable);
```

DESCRIPTION

The *posix_trace_getnext_event()* function shall report a recorded trace event either from an active trace stream without log or a pre-recorded trace stream identified by the *trid* argument. The *posix_trace_trygetnext_event()* function shall report a recorded trace event from an active trace stream without log identified by the *trid* argument.

The trace event information associated with the recorded trace event shall be copied by the function into the structure pointed to by the argument *event* and the data associated with the trace event shall be copied into the buffer pointed to by the *data* argument.

The *posix_trace_getnext_event()* function shall block if the *trid* argument identifies an active trace stream and there is currently no trace event ready to be retrieved. When returning, if a recorded trace event was reported, the variable pointed to by the *unavailable* argument shall be set to zero. Otherwise, the variable pointed to by the *unavailable* argument shall be set to a value different from zero.

The *posix_trace_timedgetnext_event()* function shall attempt to get another trace event from an active trace stream without log, as in the *posix_trace_getnext_event()* function. However, if no trace event is available from the trace stream, the implied wait shall be terminated when the timeout specified by the argument *abstime* expires, and the function shall return the error [ETIMEDOUT].

The timeout shall expire when the absolute time specified by *abstime* passes, as measured by the clock upon which timeouts are based (that is, when the value of that clock equals or exceeds *abstime*), or if the absolute time specified by *abstime* has already passed at the time of the call.

The timeout shall be based on the CLOCK_REALTIME clock. The resolution of the timeout shall be the resolution of the clock on which it is based. The **timespec** data type is defined in the **<time.h>** header.

Under no circumstance shall the function fail with a timeout if a trace event is immediately available from the trace stream. The validity of the *abstime* argument need not be checked if a trace event is immediately available from the trace stream.

49496 The behavior of this function for a pre-recorded trace stream is unspecified.

49497 TRL The *posix_trace_trygetnext_event()* function shall not block. This function shall return an error if
 49498 the *trid* argument identifies a pre-recorded trace stream. If a recorded trace event was reported,
 49499 the variable pointed to by the *unavailable* argument shall be set to zero. Otherwise, if no trace
 49500 event was reported, the variable pointed to by the *unavailable* argument shall be set to a value
 49501 different from zero.

49502 The argument *num_bytes* shall be the size of the buffer pointed to by the *data* argument. The
 49503 argument *data_len* reports to the application the length in bytes of the data record just
 49504 transferred. If *num_bytes* is greater than or equal to the size of the data associated with the trace
 49505 event pointed to by the *event* argument, all the recorded data shall be transferred. In this case,
 49506 the *truncation-status* member of the trace event structure shall be either
 49507 POSIX_TRACE_NOT_TRUNCATED, if the trace event data was recorded without truncation
 49508 while tracing, or POSIX_TRACE_TRUNCATED_RECORD, if the trace event data was truncated
 49509 when it was recorded. If the *num_bytes* argument is less than the length of recorded trace event
 49510 data, the data transferred shall be truncated to a length of *num_bytes*, the value stored in the
 49511 variable pointed to by *data_len* shall be equal to *num_bytes*, and the *truncation-status* member of
 49512 the *event* structure argument shall be set to POSIX_TRACE_TRUNCATED_READ (see the
 49513 **posix_trace_event_info** structure defined in <trace.h>).

49514 The report of a trace event shall be sequential starting from the oldest recorded trace event. Trace
 49515 events shall be reported in the order in which they were generated, up to an implementation-
 49516 defined time resolution that causes the ordering of trace events occurring very close to each
 49517 other to be unknown. Once reported, a trace event cannot be reported again from an active trace
 49518 stream. Once a trace event is reported from an active trace stream without log, the trace stream
 49519 shall make the resources associated with that trace event available to record future generated
 49520 trace events.

49521 **RETURN VALUE**

49522 Upon successful completion, these functions shall return a value of zero. Otherwise, they shall
 49523 return the corresponding error number.

49524 If successful, these functions store:

- 49525 • The recorded trace event in the object pointed to by *event*
- 49526 • The trace event information associated with the recorded trace event in the object pointed
 49527 to by *data*
- 49528 • The length of this trace event information in the object pointed to by *data_len*
- 49529 • The value of zero in the object pointed to by *unavailable*

49530 **ERRORS**

49531 These functions shall fail if:

49532 [EINVAL] The trace stream identifier argument *trid* is invalid.

49533 The *posix_trace_getnext_event()* and *posix_trace_timedgetnext_event()* functions shall fail if:

49534 [EINTR] The operation was interrupted by a signal, and so the call had no effect.

49535 The *posix_trace_trygetnext_event()* function shall fail if:

49536 [EINVAL] The trace stream identifier argument *trid* does not correspond to an active
 49537 trace stream.

49538 The *posix_trace_timedgetnext_event()* function shall fail if:

49539 [EINVAL] There is no trace event immediately available from the trace stream, and the

49540 *timeout* argument is invalid.

49541 [ETIMEDOUT] No trace event was available from the trace stream before the specified

49542 timeout *timeout* expired.

49543 EXAMPLES

49544 None.

49545 APPLICATION USAGE

49546 None.

49547 RATIONALE

49548 None.

49549 FUTURE DIRECTIONS

49550 These functions may be removed in a future version.

49551 SEE ALSO

49552 *posix_trace_close()*, *posix_trace_create()*

49553 XBD <sys/types.h>, <trace.h>

49554 CHANGE HISTORY

49555 First released in Issue 6. Derived from IEEE Std 1003.1q-2000.

49556 IEEE PASC Interpretation 1003.1 #123 is applied.

49557 Issue 7

49558 The *posix_trace_getnext_event()*, *posix_trace_timedgetnext_event()*, and

49559 *posix_trace_trygetnext_event()* functions are marked obsolescent.

49560 Functionality relating to the Timers option is moved to the Base.

49561 NAME

49562 posix_trace_open, posix_trace_rewind — trace log management (TRACING)

49563 SYNOPSIS

```
49564 OB TRC #include <trace.h>
49565 TRL     int posix_trace_open(int file_desc, trace_id_t *trid);
49566         int posix_trace_rewind(trace_id_t trid);
```

49567 DESCRIPTION

49568 Refer to *posix_trace_close()*.

49569 **NAME**49570 posix_trace_set_filter — set filter of an initialized trace stream (**TRACING**)49571 **SYNOPSIS**

```
49572 OB TRC  #include <trace.h>
49573 TEF      int posix_trace_set_filter(trace_id_t trid,
49574                                     const trace_event_set_t *set, int how);
```

49575 **DESCRIPTION**49576 Refer to *posix_trace_get_filter()*.

49577 NAME

49578 posix_trace_shutdown — trace stream shutdown from a process (**TRACING**)

49579 SYNOPSIS

```
49580 OB TRC #include <sys/types.h>  
49581 #include <trace.h>  
49582 int posix_trace_shutdown(trace_id_t trid);
```

49583 DESCRIPTION

49584 Refer to *posix_trace_create()*.

49585 **NAME**

49586 posix_trace_start, posix_trace_stop — trace start and stop (TRACING)

49587 **SYNOPSIS**

```
49588 OB TRC  #include <trace.h>
49589         int posix_trace_start(trace_id_t trid);
49590         int posix_trace_stop (trace_id_t trid);
```

49591 **DESCRIPTION**

49592 The *posix_trace_start()* and *posix_trace_stop()* functions, respectively, shall start and stop the trace
 49593 stream identified by the argument *trid*.

49594 The effect of calling the *posix_trace_start()* function shall be recorded in the trace stream as the
 49595 POSIX_TRACE_START system trace event and the status of the trace stream shall become
 49596 POSIX_TRACE_RUNNING. If the trace stream is in progress when this function is called, the
 49597 POSIX_TRACE_START system trace event shall not be recorded and the trace stream shall
 49598 continue to run. If the trace stream is full, the POSIX_TRACE_START system trace event shall
 49599 not be recorded and the status of the trace stream shall not be changed.

49600 The effect of calling the *posix_trace_stop()* function shall be recorded in the trace stream as the
 49601 POSIX_TRACE_STOP system trace event and the status of the trace stream shall become
 49602 POSIX_TRACE_SUSPENDED. If the trace stream is suspended when this function is called, the
 49603 POSIX_TRACE_STOP system trace event shall not be recorded and the trace stream shall remain
 49604 suspended. If the trace stream is full, the POSIX_TRACE_STOP system trace event shall not be
 49605 recorded and the status of the trace stream shall not be changed.

49606 **RETURN VALUE**

49607 Upon successful completion, these functions shall return a value of zero. Otherwise, they shall
 49608 return the corresponding error number.

49609 **ERRORS**

49610 These functions shall fail if:

49611	[EINVAL]	The value of the argument <i>trid</i> does not correspond to an active trace stream
49612		and thus no trace stream was started or stopped.
49613	[EINTR]	The operation was interrupted by a signal and thus the trace stream was not
49614		necessarily started or stopped.

49615 **EXAMPLES**

49616 None.

49617 **APPLICATION USAGE**

49618 None.

49619 **RATIONALE**

49620 None.

49621 **FUTURE DIRECTIONS**

49622 The *posix_trace_start()* and *posix_trace_stop()* functions may be removed in a future version.

49623 **SEE ALSO**

49624 *posix_trace_create()*

49625 XBD *<trace.h>*

CHANGE HISTORY

49626
49627 First released in Issue 6. Derived from IEEE Std 1003.1q-2000.

49628 IEEE PASC Interpretation 1003.1 #123 is applied.

Issue 7

49629
49630 The *posix_trace_start()* and *posix_trace_stop()* functions are marked obsolescent.

49631 **NAME**49632 posix_trace_timedgetnext_event — retrieve a trace event (**TRACING**)49633 **SYNOPSIS**

```
49634 OB TRC #include <sys/types.h>
49635         #include <trace.h>

49636         int posix_trace_timedgetnext_event(trace_id_t trid,
49637         struct posix_trace_event_info *restrict event,
49638         void *restrict data, size_t num_bytes,
49639         size_t *restrict data_len, int *restrict unavailable,
49640         const struct timespec *restrict abstime);
```

49641 **DESCRIPTION**49642 Refer to [posix_trace_getnext_event\(\)](#).

49643 NAME

49644 posix_trace_trid_eventid_open — open a trace event type identifier (**TRACING**)

49645 SYNOPSIS

```
49646 OB TRC #include <trace.h>
49647 TEF int posix_trace_trid_eventid_open(trace_id_t trid,
49648 const char *restrict event_name,
49649 trace_event_id_t *restrict event);
```

49650 DESCRIPTION

49651 Refer to *posix_trace_eventid_equal()*.

49652 **NAME**

49653 posix_trace_trygetnext_event — retrieve a trace event (TRACING)

49654 **SYNOPSIS**

```
49655 OB TRC #include <sys/types.h>
49656         #include <trace.h>
49657
49657         int posix_trace_trygetnext_event(trace_id_t trid,
49658         struct posix_trace_event_info *restrict event,
49659         void *restrict data, size_t num_bytes,
49660         size_t *restrict data_len, int *restrict unavailable);
```

49661 **DESCRIPTION**49662 Refer to [posix_trace_getnext_event\(\)](#).

NAME

posix_typed_mem_get_info — query typed memory information (ADVANCED REALTIME)

SYNOPSIS

```
#include <sys/mman.h>

int posix_typed_mem_get_info(int fildes,
    struct posix_typed_mem_info *info);
```

DESCRIPTION

The *posix_typed_mem_get_info()* function shall return, in the *posix_tmi_length* field of the **posix_typed_mem_info** structure pointed to by *info*, the maximum length which may be successfully allocated by the typed memory object designated by *fildes*. This maximum length shall take into account the flag `POSIX_TYPED_MEM_ALLOCATE` or `POSIX_TYPED_MEM_ALLOCATE_CONTIG` specified when the typed memory object represented by *fildes* was opened. The maximum length is dynamic; therefore, the value returned is valid only while the current mapping of the corresponding typed memory pool remains unchanged.

If *fildes* represents a typed memory object opened with neither the `POSIX_TYPED_MEM_ALLOCATE` flag nor the `POSIX_TYPED_MEM_ALLOCATE_CONTIG` flag specified, the returned value of *info*->*posix_tmi_length* is unspecified.

The *posix_typed_mem_get_info()* function may return additional implementation-defined information in other fields of the **posix_typed_mem_info** structure pointed to by *info*.

If the memory object specified by *fildes* is not a typed memory object, then the behavior of this function is undefined.

RETURN VALUE

Upon successful completion, the *posix_typed_mem_get_info()* function shall return zero; otherwise, the corresponding error status value shall be returned.

ERRORS

The *posix_typed_mem_get_info()* function shall fail if:

- [EBADF] The *fildes* argument is not a valid open file descriptor.
- [ENODEV] The *fildes* argument is not connected to a memory object supported by this function.

This function shall not return an error code of [EINTR].

EXAMPLES

None.

APPLICATION USAGE

None.

RATIONALE

An application that needs to allocate a block of typed memory with length dependent upon the amount of memory currently available must either query the typed memory object to obtain the amount available, or repeatedly invoke *mmap()* attempting to guess an appropriate length. While the latter method is existing practice with *malloc()*, it is awkward and imprecise. The *posix_typed_mem_get_info()* function allows an application to immediately determine available memory. This is particularly important for typed memory objects that may in some cases be scarce resources. Note that when a typed memory pool is a shared resource, some form of mutual-exclusion or synchronization may be required while typed memory is being queried and

49707 allocated to prevent race conditions.

49708 The existing *fstat()* function is not suitable for this purpose. We realize that implementations
49709 may wish to provide other attributes of typed memory objects (for example, alignment
49710 requirements, page size, and so on). The *fstat()* function returns a structure which is not
49711 extensible and, furthermore, contains substantial information that is inappropriate for typed
49712 memory objects.

49713 **FUTURE DIRECTIONS**

49714 None.

49715 **SEE ALSO**

49716 *fstat()*, *mmap()*, *posix_typed_mem_open()*

49717 XBD <sys/mman.h>

49718 **CHANGE HISTORY**

49719 First released in Issue 6. Derived from IEEE Std 1003.1j-2000.

NAME

posix_typed_mem_open — open a typed memory object (**ADVANCED REALTIME**)

SYNOPSIS

```
#include <sys/mman.h>

int posix_typed_mem_open(const char *name, int oflag, int tflag);
```

DESCRIPTION

The *posix_typed_mem_open()* function shall establish a connection between the typed memory object specified by the string pointed to by *name* and a file descriptor. It shall create an open file description that refers to the typed memory object and a file descriptor that refers to that open file description. The file descriptor is used by other functions to refer to that typed memory object. It is unspecified whether the name appears in the file system and is visible to other functions that take pathnames as arguments. The *name* argument conforms to the construction rules for a pathname, except that the interpretation of <slash> characters other than the leading <slash> character in *name* is implementation-defined, and that the length limits for the *name* argument are implementation-defined and need not be the same as the pathname limits {PATH_MAX} and {NAME_MAX}. If *name* begins with the <slash> character, then processes calling *posix_typed_mem_open()* with the same value of *name* shall refer to the same typed memory object. If *name* does not begin with the <slash> character, the effect is implementation-defined.

Each typed memory object supported in a system shall be identified by a name which specifies not only its associated typed memory pool, but also the path or port by which it is accessed. That is, the same typed memory pool accessed via several different ports shall have several different corresponding names. The binding between names and typed memory objects is established in an implementation-defined manner. Unlike shared memory objects, there is no way within POSIX.1-2008 for a program to create a typed memory object.

The value of *tflag* shall determine how the typed memory object behaves when subsequently mapped by calls to *mmap()*. At most, one of the following flags defined in **<sys/mman.h>** may be specified:

POSIX_TYPED_MEM_ALLOCATE

Allocate on *mmap()*.

POSIX_TYPED_MEM_ALLOCATE_CONTIG

Allocate contiguously on *mmap()*.

POSIX_TYPED_MEM_MAP_ALLOCATABLE

Map on *mmap()*, without affecting allocatability.

If *tflag* has the flag POSIX_TYPED_MEM_ALLOCATE specified, any subsequent call to *mmap()* using the returned file descriptor shall result in allocation and mapping of typed memory from the specified typed memory pool. The allocated memory may be a contiguous previously unallocated area of the typed memory pool or several non-contiguous previously unallocated areas (mapped to a contiguous portion of the process address space). If *tflag* has the flag POSIX_TYPED_MEM_ALLOCATE_CONTIG specified, any subsequent call to *mmap()* using the returned file descriptor shall result in allocation and mapping of a single contiguous previously unallocated area of the typed memory pool (also mapped to a contiguous portion of the process address space). If *tflag* has none of the flags POSIX_TYPED_MEM_ALLOCATE or POSIX_TYPED_MEM_ALLOCATE_CONTIG specified, any subsequent call to *mmap()* using the returned file descriptor shall map an application-chosen area from the specified typed memory pool such that this mapped area becomes unavailable for allocation until unmapped by all processes. If *tflag* has the flag POSIX_TYPED_MEM_MAP_ALLOCATABLE specified, any

subsequent call to *mmap()* using the returned file descriptor shall map an application-chosen area from the specified typed memory pool without an effect on the availability of that area for allocation; that is, mapping such an object leaves each byte of the mapped area unallocated if it was unallocated prior to the mapping or allocated if it was allocated prior to the mapping. Appropriate privileges to specify the `POSIX_TYPED_MEM_MAP_ALLOCATABLE` flag are implementation-defined.

If successful, *posix_typed_mem_open()* shall return a file descriptor for the typed memory object that is the lowest numbered file descriptor not currently open for that process. The open file description is new, and therefore the file descriptor shall not share it with any other processes. It is unspecified whether the file offset is set. The `FD_CLOEXEC` file descriptor flag associated with the new file descriptor shall be cleared.

The behavior of *msync()*, *ftruncate()*, and all file operations other than *mmap()*, *posix_mem_offset()*, *posix_typed_mem_get_info()*, *fstat()*, *dup()*, *dup2()*, and *close()*, is unspecified when passed a file descriptor connected to a typed memory object by this function.

The file status flags of the open file description shall be set according to the value of *oflag*. Applications shall specify exactly one of the three access mode values described below and defined in the `<fcntl.h>` header, as the value of *oflag*.

<code>O_RDONLY</code>	Open for read access only.
<code>O_WRONLY</code>	Open for write access only.
<code>O_RDWR</code>	Open for read or write access.

RETURN VALUE

Upon successful completion, the *posix_typed_mem_open()* function shall return a non-negative integer representing the lowest numbered unused file descriptor. Otherwise, it shall return `-1` and set *errno* to indicate the error.

ERRORS

The *posix_typed_mem_open()* function shall fail if:

<code>[EACCES]</code>	The typed memory object exists and the permissions specified by <i>oflag</i> are denied.	

<code>[EINTR]</code>	The <i>posix_typed_mem_open()</i> operation was interrupted by a signal.	

<code>[EINVAL]</code>	The flags specified in <i>tflag</i> are invalid (more than one of <code>POSIX_TYPED_MEM_ALLOCATE</code> , <code>POSIX_TYPED_MEM_ALLOCATE_CONTIG</code> , or <code>POSIX_TYPED_MEM_MAP_ALLOCATABLE</code> is specified).	

<code>[EMFILE]</code>	All file descriptors available to the process are currently open.	

<code>[ENFILE]</code>	Too many file descriptors are currently open in the system.	+

<code>[ENOENT]</code>	The named typed memory object does not exist.	+

<code>[EPERM]</code>	The caller lacks appropriate privileges to specify the <code>POSIX_TYPED_MEM_MAP_ALLOCATABLE</code> flag in the <i>tflag</i> argument.	+

	The <i>posix_typed_mem_open()</i> function may fail if:	+

<code>[ENAMETOOLONG]</code>		

	The length of the <i>name</i> argument exceeds <code>{_POSIX_PATH_MAX}</code> on systems that do not support the XSI option <code>or exceeds <code>{_XOPEN_PATH_MAX}</code> on XSI systems,</code> or has a pathname component that is longer than	

49810 XSI { _POSIX_NAME_MAX } on systems that do not support the XSI option or
 49811 longer than { _XOPEN_NAME_MAX } on XSI systems. -

49812 EXAMPLES

49813 None.

49814 APPLICATION USAGE

49815 None.

49816 RATIONALE

49817 None.

49818 FUTURE DIRECTIONS

49819 None.

49820 SEE ALSO

49821 *close(), dup(), exec, fcntl(), fstat(), ftruncate(), mmap(), msync(), posix_mem_offset(),*
 49822 *posix_typed_mem_get_info(), umask()*

49823 XBD <fcntl.h>, <sys/mman.h>

49824 CHANGE HISTORY

49825 First released in Issue 6. Derived from IEEE Std 1003.1j-2000.

49826 Issue 7

49827 Austin Group Interpretation 1003.1-2001 #143 is applied.

49828 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0442 [119,428] is applied. +

49829 **NAME**

49830 pow, powf, powl — power function

49831 **SYNOPSIS**

```
49832 #include <math.h>
49833 double pow(double x, double y);
49834 float powf(float x, float y);
49835 long double powl(long double x, long double y);
```

49836 **DESCRIPTION**

49837 CX The functionality described on this reference page is aligned with the ISO C standard. Any
 49838 conflict between the requirements described here and the ISO C standard is unintentional. This
 49839 volume of POSIX.1-2008 defers to the ISO C standard.

49840 These functions shall compute the value of x raised to the power y , x^y . If x is negative, the
 49841 application shall ensure that y is an integer value.

49842 An application wishing to check for error situations should set *errno* to zero and call
 49843 *feclearexcept*(FE_ALL_EXCEPT) before calling these functions. On return, if *errno* is non-zero or
 49844 *fetestexcept*(FE_INVALID | FE_DIVBYZERO | FE_OVERFLOW | FE_UNDERFLOW) is non-
 49845 zero, an error has occurred.

49846 **RETURN VALUE**

49847 Upon successful completion, these functions shall return the value of x raised to the power y .

49848 MX For finite values of $x < 0$, and finite non-integer values of y , a domain error shall occur and
 49849 either a NaN (if representable), or an implementation-defined value shall be returned.

49850 If the correct value would cause overflow, a range error shall occur and *pow()*, *powf()*, and
 49851 *powl()* shall return $\pm\text{HUGE\_VAL}$, $\pm\text{HUGE\_VALF}$, and $\pm\text{HUGE\_VALL}$, respectively, with the
 49852 same sign as the correct value of the function.

49853 MXX If the correct value would cause underflow, and is not representable, a range error may occur, |
 49854 MXX and *pow()*, *powf()*, and *powl()* shall return 0.0, or (if IEC 60559 Floating-Point is not supported) |
 49855 an implementation-defined value no greater in magnitude than DBL_MIN, FLT_MIN, and |
 49856 LDBL_MIN, respectively.

49857 CX For $y < 0$, if x is zero, a pole error may occur and *pow()*, *powf()*, and *powl()* shall return |
 49858 MX $\pm\text{HUGE\_VAL}$, $\pm\text{HUGE\_VALF}$, and $\pm\text{HUGE\_VALL}$, respectively. On systems that support the |
 49859 IEC 60559 Floating-Point option, if x is ± 0 , a pole error shall occur and *pow()*, *powf()*, and *powl()* |
 49860 shall return $\pm\text{HUGE\_VAL}$, $\pm\text{HUGE\_VALF}$, and $\pm\text{HUGE\_VALL}$, respectively if y is an odd integer, +
 49861 or HUGE_VAL , HUGE_VALF , and HUGE_VALL , respectively if y is not an odd integer.

49862 MX If x or y is a NaN, a NaN shall be returned (unless specified elsewhere in this description).

49863 For any value of y (including NaN), if x is +1, 1.0 shall be returned.

49864 For any value of x (including NaN), if y is ± 0 , 1.0 shall be returned.

49865 For any odd integer value of $y > 0$, if x is ± 0 , ± 0 shall be returned.

49866 For $y > 0$ and not an odd integer, if x is ± 0 , +0 shall be returned.

49867 If x is -1, and y is $\pm\text{Inf}$, 1.0 shall be returned.

49868 For $|x| < 1$, if y is $-\text{Inf}$, $+\text{Inf}$ shall be returned.

49869 For $|x| > 1$, if y is $-\text{Inf}$, +0 shall be returned.

49870 For $|x| < 1$, if y is $+\text{Inf}$, +0 shall be returned.

49871 For $|x| > 1$, if y is $+\text{Inf}$, $+\text{Inf}$ shall be returned.

49872 For y an odd integer < 0 , if x is $-\text{Inf}$, -0 shall be returned.

49873 For $y < 0$ and not an odd integer, if x is $-\text{Inf}$, $+0$ shall be returned.

49874 For y an odd integer > 0 , if x is $-\text{Inf}$, $-\text{Inf}$ shall be returned.

49875 For $y > 0$ and not an odd integer, if x is $-\text{Inf}$, $+\text{Inf}$ shall be returned.

49876 For $y < 0$, if x is $+\text{Inf}$, $+0$ shall be returned.

49877 For $y > 0$, if x is $+\text{Inf}$, $+\text{Inf}$ shall be returned.

49878 MXX If the correct value would cause underflow, and is representable, a range error may occur and
49879 the correct value shall be returned.

ERRORS

49880 These functions shall fail if:

49881

49882 Domain Error The value of x is negative and y is a finite non-integer.

49883 If the integer expression $(\text{math\_errhandling} \ \& \ \text{MATH\_ERRNO})$ is non-zero,
49884 then *errno* shall be set to [EDOM]. If the integer expression $(\text{math\_errhandling}$
49885 $\ \& \ \text{MATH\_ERREXCEPT})$ is non-zero, then the invalid floating-point exception
49886 shall be raised.

49887 MX Pole Error The value of x is zero and y is negative.

49888 If the integer expression $(\text{math\_errhandling} \ \& \ \text{MATH\_ERRNO})$ is non-zero,
49889 then *errno* shall be set to [ERANGE]. If the integer expression
49890 $(\text{math\_errhandling} \ \& \ \text{MATH\_ERREXCEPT})$ is non-zero, then the divide-by-zero
49891 floating-point exception shall be raised.

49892 Range Error The result overflows.

49893 If the integer expression $(\text{math\_errhandling} \ \& \ \text{MATH\_ERRNO})$ is non-zero,
49894 then *errno* shall be set to [ERANGE]. If the integer expression
49895 $(\text{math\_errhandling} \ \& \ \text{MATH\_ERREXCEPT})$ is non-zero, then the overflow
49896 floating-point exception shall be raised.

49897 These functions may fail if:

49898 Pole Error The value of x is zero and y is negative.

49899 If the integer expression $(\text{math\_errhandling} \ \& \ \text{MATH\_ERRNO})$ is non-zero,
49900 then *errno* shall be set to [ERANGE]. If the integer expression
49901 $(\text{math\_errhandling} \ \& \ \text{MATH\_ERREXCEPT})$ is non-zero, then the divide-by-zero
49902 floating-point exception shall be raised.

49903 Range Error The result underflows.

49904 If the integer expression $(\text{math\_errhandling} \ \& \ \text{MATH\_ERRNO})$ is non-zero,
49905 then *errno* shall be set to [ERANGE]. If the integer expression
49906 $(\text{math\_errhandling} \ \& \ \text{MATH\_ERREXCEPT})$ is non-zero, then the underflow
49907 floating-point exception shall be raised.

EXAMPLES

None.

APPLICATION USAGE

On error, the expressions `(math_errhandling & MATH_ERRNO)` and `(math_errhandling & MATH_ERREXCEPT)` are independent of each other, but at least one of them must be non-zero.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

`exp()`, `feclearexcept()`, `fetestexcept()`, `isnan()`

XBD Section 4.19 (on page 116), `<math.h>`

CHANGE HISTORY

First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 5

The DESCRIPTION is updated to indicate how an application should check for an error. This text was previously published in the APPLICATION USAGE section.

Issue 6

The normative text is updated to avoid use of the term “must” for application requirements.

The `powf()` and `powl()` functions are added for alignment with the ISO/IEC 9899:1999 standard.

The DESCRIPTION, RETURN VALUE, ERRORS, and APPLICATION USAGE sections are revised to align with the ISO/IEC 9899:1999 standard.

IEC 60559:1989 standard floating-point extensions over the ISO/IEC 9899:1999 standard are marked.

IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/42 is applied, correcting the third paragraph in the RETURN VALUE section.

Issue 7

ISO/IEC 9899:1999 standard, Technical Corrigendum 2 #51 (SD5-XSH-ERN-81) is applied.

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0443 [68], XSH/TC1-2008/0444 [148], + and XSH/TC1-2008/0445 [68] are applied.

49938 **NAME**

49939

```
pread
```

 — read from a file

49940 **SYNOPSIS**

49941

```
#include <unistd.h>
```

49942

```
ssize_t pread(int fildes, void *buf, size_t nbyte, off_t offset);
```

49943 **DESCRIPTION**

49944 Refer to *read()*.

49945 **NAME**

49946 printf — print formatted output

49947 **SYNOPSIS**

49948 #include <stdio.h>

49949 int printf(const char *restrict *format*, ...);49950 **DESCRIPTION**49951 Refer to *fprintf()*.

NAME

pselect, select — synchronous I/O multiplexing

SYNOPSIS

```
#include <sys/select.h>

int pselect(int nfds, fd_set *restrict readfds,
            fd_set *restrict writefds, fd_set *restrict errorfds,
            const struct timespec *restrict timeout,
            const sigset_t *restrict sigmask);
int select(int nfds, fd_set *restrict readfds,
            fd_set *restrict writefds, fd_set *restrict errorfds,
            struct timeval *restrict timeout);
void FD_CLR(int fd, fd_set *fdset);
int FD_ISSET(int fd, fd_set *fdset);
void FD_SET(int fd, fd_set *fdset);
void FD_ZERO(fd_set *fdset);
```

DESCRIPTION

The *pselect()* function shall examine the file descriptor sets whose addresses are passed in the *readfds*, *writefds*, and *errorfds* parameters to see whether some of their descriptors are ready for reading, are ready for writing, or have an exceptional condition pending, respectively.

The *select()* function shall be equivalent to the *pselect()* function, except as follows:

- For the *select()* function, the timeout period is given in seconds and microseconds in an argument of type **struct timeval**, whereas for the *pselect()* function the timeout period is given in seconds and nanoseconds in an argument of type **struct timespec**.
- The *select()* function has no *sigmask* argument; it shall behave as *pselect()* does when *sigmask* is a null pointer.
- Upon successful completion, the *select()* function may modify the object pointed to by the *timeout* argument.

The *pselect()* and *select()* functions shall support regular files, terminal and pseudo-terminal devices, **STREAMS-based files**, FIFOs, pipes, and sockets. The behavior of *pselect()* and *select()* on file descriptors that refer to other types of file is unspecified.

The *nfds* argument specifies the range of descriptors to be tested. The first *nfds* descriptors shall be checked in each set; that is, the descriptors from zero through *nfds*–1 in the descriptor sets shall be examined.

If the *readfds* argument is not a null pointer, it points to an object of type **fd_set** that on input specifies the file descriptors to be checked for being ready to read, and on output indicates which file descriptors are ready to read.

If the *writefds* argument is not a null pointer, it points to an object of type **fd_set** that on input specifies the file descriptors to be checked for being ready to write, and on output indicates which file descriptors are ready to write.

If the *errorfds* argument is not a null pointer, it points to an object of type **fd_set** that on input specifies the file descriptors to be checked for error conditions pending, and on output indicates which file descriptors have error conditions pending.

Upon successful completion, the *pselect()* or *select()* function shall modify the objects pointed to by the *readfds*, *writefds*, and *errorfds* arguments to indicate which file descriptors are ready for reading, ready for writing, or have an error condition pending, respectively, and shall return the total number of ready descriptors in all the output sets. For each file descriptor less than *nfds*, the

49998 corresponding bit shall be set upon successful completion if it was set on input and the
 49999 associated condition is true for that file descriptor.

50000 If none of the selected descriptors are ready for the requested operation, the *pselect()* or *select()*
 50001 function shall block until at least one of the requested operations becomes ready, until the
 50002 *timeout* occurs, or until interrupted by a signal. The *timeout* parameter controls how long the
 50003 *pselect()* or *select()* function shall take before timing out. If the *timeout* parameter is not a null
 50004 pointer, it specifies a maximum interval to wait for the selection to complete. If the specified
 50005 time interval expires without any requested operation becoming ready, the function shall return.
 50006 If the *timeout* parameter is a null pointer, then the call to *pselect()* or *select()* shall block
 50007 indefinitely until at least one descriptor meets the specified criteria. To effect a poll, the *timeout*
 50008 parameter should not be a null pointer, and should point to a zero-valued **timespec** structure.

50009 The use of a timeout does not affect any pending timers set up by *alarm()* or *setitimer()*.

50010 Implementations may place limitations on the maximum timeout interval supported. All
 50011 implementations shall support a maximum timeout interval of at least 31 days. If the *timeout*
 50012 argument specifies a timeout interval greater than the implementation-defined maximum value,
 50013 the maximum value shall be used as the actual timeout value. Implementations may also place
 50014 limitations on the granularity of timeout intervals. If the requested timeout interval requires a
 50015 finer granularity than the implementation supports, the actual timeout interval shall be rounded
 50016 up to the next supported value.

50017 If *sigmask* is not a null pointer, then the *pselect()* function shall replace the signal mask of the
 50018 caller by the set of signals pointed to by *sigmask* before examining the descriptors, and shall
 50019 restore the signal mask of the calling thread before returning.

50020 A descriptor shall be considered ready for reading when a call to an input function with
 50021 O_NONBLOCK clear would not block, whether or not the function would transfer data
 50022 successfully. (The function might return data, an end-of-file indication, or an error other than
 50023 one indicating that it is blocked, and in each of these cases the descriptor shall be considered
 50024 ready for reading.)

50025 A descriptor shall be considered ready for writing when a call to an output function with
 50026 O_NONBLOCK clear would not block, whether or not the function would transfer data
 50027 successfully.

50028 If a socket has a pending error, it shall be considered to have an exceptional condition pending.
 50029 Otherwise, what constitutes an exceptional condition is file type-specific. For a file descriptor for
 50030 use with a socket, it is protocol-specific except as noted below. For other file types it is
 50031 implementation-defined. If the operation is meaningless for a particular file type, *pselect()* or
 50032 *select()* shall indicate that the descriptor is ready for read or write operations, and shall indicate
 50033 that the descriptor has no exceptional condition pending.

50034 If a descriptor refers to a socket, the implied input function is the *recvmsg()* function with
 50035 parameters requesting normal and ancillary data, such that the presence of either type shall
 50036 cause the socket to be marked as readable. The presence of out-of-band data shall be checked if
 50037 the socket option SO_OOBINLINE has been enabled, as out-of-band data is enqueued with
 50038 normal data. If the socket is currently listening, then it shall be marked as readable if an
 50039 incoming connection request has been received, and a call to the *accept()* function shall complete
 50040 without blocking.

50041 If a descriptor refers to a socket, the implied output function is the *sendmsg()* function supplying
 50042 an amount of normal data equal to the current value of the SO_SNDLOWAT option for the
 50043 socket. If a non-blocking call to the *connect()* function has been made for a socket, and the
 50044 connection attempt has either succeeded or failed leaving a pending error, the socket shall be

50045 marked as writable.

50046 A socket shall be considered to have an exceptional condition pending if a receive operation
 50047 with `O_NONBLOCK` clear for the open file description and with the `MSG_OOB` flag set would
 50048 return out-of-band data without blocking. (It is protocol-specific whether the `MSG_OOB` flag
 50049 would be used to read out-of-band data.) A socket shall also be considered to have an
 50050 exceptional condition pending if an out-of-band data mark is present in the receive queue. Other
 50051 circumstances under which a socket may be considered to have an exceptional condition
 50052 pending are protocol-specific and implementation-defined.

50053 If the *readfds*, *writfds*, and *errorfds* arguments are all null pointers and the *timeout* argument is
 50054 not a null pointer, the *pselect()* or *select()* function shall block for the time specified, or until
 50055 interrupted by a signal. If the *readfds*, *writfds*, and *errorfds* arguments are all null pointers and
 50056 the *timeout* argument is a null pointer, the *pselect()* or *select()* function shall block until
 50057 interrupted by a signal.

50058 File descriptors associated with regular files shall always select true for ready to read, ready to
 50059 write, and error conditions.

50060 On failure, the objects pointed to by the *readfds*, *writfds*, and *errorfds* arguments shall not be
 50061 modified. If the timeout interval expires without the specified condition being true for any of the
 50062 specified file descriptors, the objects pointed to by the *readfds*, *writfds*, and *errorfds* arguments
 50063 shall have all bits set to 0.

50064 File descriptor masks of type **fd_set** can be initialized and tested with *FD_CLR()*, *FD_ISSET()*,
 50065 *FD_SET()*, and *FD_ZERO()*. It is unspecified whether each of these is a macro or a function. If a
 50066 macro definition is suppressed in order to access an actual function, or a program defines an
 50067 external identifier with any of these names, the behavior is undefined.

50068 *FD_CLR(fd, fdsetp)* shall remove the file descriptor *fd* from the set pointed to by *fdsetp*. If *fd* is not
 50069 a member of this set, there shall be no effect on the set, nor will an error be returned.

50070 *FD_ISSET(fd, fdsetp)* shall evaluate to non-zero if the file descriptor *fd* is a member of the set
 50071 pointed to by *fdsetp*, and shall evaluate to zero otherwise.

50072 *FD_SET(fd, fdsetp)* shall add the file descriptor *fd* to the set pointed to by *fdsetp*. If the file
 50073 descriptor *fd* is already in this set, there shall be no effect on the set, nor will an error be
 50074 returned.

50075 *FD_ZERO(fdsetp)* shall initialize the descriptor set pointed to by *fdsetp* to the null set. No error is
 50076 returned if the set is not empty at the time *FD_ZERO()* is invoked.

50077 The behavior of these macros is undefined if the *fd* argument is less than 0 or greater than or
 50078 equal to `FD_SETSIZE`, or if *fd* is not a valid file descriptor, or if any of the arguments are
 50079 expressions with side-effects.

50080 If a thread gets canceled during a *pselect()* call, the signal mask in effect when executing the
 50081 registered cleanup functions is either the original signal mask or the signal mask installed as part
 50082 of the *pselect()* call.

50083 **RETURN VALUE**

50084 Upon successful completion, the *pselect()* and *select()* functions shall return the total number of
 50085 bits set in the bit masks. Otherwise, `-1` shall be returned, and *errno* shall be set to indicate the
 50086 error.

50087 *FD_CLR()*, *FD_SET()*, and *FD_ZERO()* do not return a value. *FD_ISSET()* shall return a non-
 50088 zero value if the bit for the file descriptor *fd* is set in the file descriptor set pointed to by *fdset*, and
 50089 0 otherwise.

ERRORS

Under the following conditions, *pselect()* and *select()* shall fail and set *errno* to:

[EBADF] One or more of the file descriptor sets specified a file descriptor that is not a valid open file descriptor.

[EINTR] The function was interrupted before any of the selected events occurred and before the timeout interval expired.

If SA_RESTART has been set for the interrupting signal, it is implementation-defined whether the function restarts or returns with [EINTR].

[EINVAL] An invalid timeout interval was specified.

[EINVAL] The *nfds* argument is less than 0 or greater than FD_SETSIZE.

[EINVAL] One of the specified file descriptors refers to a STREAM or multiplexer that is linked (directly or indirectly) downstream from a multiplexer.

EXAMPLES

None.

APPLICATION USAGE

None.

RATIONALE

In earlier versions of the Single UNIX Specification, the *select()* function was defined in the **<sys/time.h>** header. This is now changed to **<sys/select.h>**. The rationale for this change was as follows: the introduction of the *pselect()* function included the **<sys/select.h>** header and the **<sys/select.h>** header defines all the related definitions for the *pselect()* and *select()* functions. Backwards-compatibility to existing XSI implementations is handled by allowing **<sys/time.h>** to include **<sys/select.h>**.

Code which wants to avoid the ambiguity of the signal mask for thread cancellation handlers can install an additional cancellation handler which resets the signal mask to the expected value.

```
void cleanup(void *arg)
{
    sigset_t *ss = (sigset_t *) arg;
    pthread_sigmask(SIG_SETMASK, ss, NULL);
}

int call_pselect(int nfds, fd_set *readfds, fd_set *writefds,
    fd_set errorfds, const struct timespec *timeout,
    const sigset_t *sigmask)
{
    sigset_t oldmask;
    int result;
    pthread_sigmask(SIG_SETMASK, NULL, &oldmask);
    pthread_cleanup_push(cleanup, &oldmask);
    result = pselect(nfds, readfds, writefds, errorfds, timeout, sigmask);
    pthread_cleanup_pop(0);
    return result;
}
```

FUTURE DIRECTIONS

None.

SEE ALSO

accept(), *alarm()*, *connect()*, *fcntl()*, *getitimer()*, *poll()*, *read()*, *recvmsg()*, *sendmsg()*, *write()*

XBD **<sys/select.h>**, **<sys/time.h>**

CHANGE HISTORY

First released in Issue 4, Version 2.

Issue 5

Moved from X/OPEN UNIX extension to BASE.

In the ERRORS section, the text has been changed to indicate that [EINVAL] is returned when *nfds* is less than 0 or greater than FD_SETSIZE. It previously stated less than 0, or greater than or equal to FD_SETSIZE.

Text about *timeout* is moved from the APPLICATION USAGE section to the DESCRIPTION.

Issue 6

The Open Group Corrigendum U026/6 is applied, changing the occurrences of *readfs* and *writefs* in the *select()* DESCRIPTION to be *readfds* and *writefds*.

Text referring to sockets is added to the DESCRIPTION.

The DESCRIPTION and ERRORS sections are updated so that references to STREAMS are marked as part of the XSI STREAMS Option Group.

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- These functions are now mandatory.

The *pselect()* function is added for alignment with IEEE Std 1003.1g-2000 and additional detail related to sockets semantics is added to the DESCRIPTION.

The *select()* function now requires inclusion of **<sys/select.h>**.

The **restrict** keyword is added to the *select()* prototype for alignment with the ISO/IEC 9899:1999 standard.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/70 is applied, updating the DESCRIPTION to reference the signal mask in terms of the calling thread rather than the process.

Issue 7

SD5-XSH-ERN-122 is applied, adding text to the DESCRIPTION for when a thread is canceled during a call to *pselect()*, and adding example code to the RATIONALE.

Functionality relating to the XSI STREAMS option is marked obsolescent.

Functionality relating to the Threads option is moved to the Base.

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0446 [372] is applied.

+

NAME

psiginfo, psignal — print signal information to standard error

SYNOPSIS

```
CX      #include <signal.h>

void psiginfo(const siginfo_t *pinfo, const char *message);
void psignal(int signum, const char *message);
```

DESCRIPTION

The *psiginfo()* and *psignal()* functions shall print a message out on *stderr* associated with a signal number. If *message* is not null and is not the empty string, then the string pointed to by the *message* argument shall be printed first, followed by a <colon>, a <space>, and the signal description string indicated by *signum*, or by the signal associated with *pinfo*. If the *message* argument is null or points to an empty string, then only the signal description shall be printed. For *psiginfo()*, the argument *pinfo* references a valid **siginfo_t** structure. For *psignal()*, if *signum* is not a valid signal number, the behavior is implementation-defined.

The *psiginfo()* and *psignal()* functions shall not change the orientation of the standard error stream.

The *psiginfo()* and *psignal()* functions shall mark for update the last data modification and last file status change timestamps of the file associated with the standard error stream at some time between their successful completion and *exit()*, *abort()*, or the completion of *fflush()* or *fclose()* on *stderr*.

The *psiginfo()* and *psignal()* functions shall not change the setting of *errno* if successful. +

On error, the *psiginfo()* and *psignal()* functions shall set the error indicator for the stream to + which *stderr* points, and shall set *errno* to indicate the error. +

Since no value is returned, an application wishing to check for error situations should set *errno* + to 0, then call *psiginfo()* or *psignal()*, then check *errno*. +

RETURN VALUE

These functions shall not return a value.

ERRORS

Refer to *fputc()*.

EXAMPLES

None.

APPLICATION USAGE

As an alternative to setting *errno* to zero before the call and checking if it is non-zero afterwards, applications can use *ferror()* to detect whether *psiginfo()* or *psignal()* encountered an error. |

An application wishing to use this method to check for error situations should call *clearerr(stderr)* before calling *psiginfo()* or *psignal()*, then if *ferror(stderr)* returns non-zero, the value of *errno* indicates which error occurred. |

RATIONALE

System V historically has *psignal()* and *psiginfo()* in **<siginfo.h>**. However, the **<siginfo.h>** header is not specified in the Base Definitions volume of POSIX.1-2008, and the type **siginfo_t** is defined in **<signal.h>**.

50209 FUTURE DIRECTIONS

50210 None.

50211 SEE ALSO

50212 *fputc()*, *perror()*, *strsignal()*

50213 XBD <signal.h>

50214 CHANGE HISTORY

50215 First released in Issue 7.

50216 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0447 [399,428], XSH/TC1-2008/0448 +
50217 [399], and XSH/TC1-2008/0449 [399,401] are applied.

NAME

pthread_atfork — register fork handlers

SYNOPSIS

```
#include <pthread.h>

int pthread_atfork(void (*prepare)(void), void (*parent)(void),
void (*child)(void));
```

DESCRIPTION

The *pthread_atfork()* function shall declare fork handlers to be called before and after *fork()*, in the context of the thread that called *fork()*. The *prepare* fork handler shall be called before *fork()* processing commences. The *parent* fork handler shall be called after *fork()* processing completes in the parent process. The *child* fork handler shall be called after *fork()* processing completes in the child process. If no handling is desired at one or more of these three points, the corresponding fork handler address(es) may be set to NULL.

The order of calls to *pthread_atfork()* is significant. The *parent* and *child* fork handlers shall be called in the order in which they were established by calls to *pthread_atfork()*. The *prepare* fork handlers shall be called in the opposite order.

RETURN VALUE

Upon successful completion, *pthread_atfork()* shall return a value of zero; otherwise, an error number shall be returned to indicate the error.

ERRORS

The *pthread_atfork()* function shall fail if:

[ENOMEM] Insufficient table space exists to record the fork handler addresses.

The *pthread_atfork()* function shall not return an error code of [EINTR].

EXAMPLES

None.

APPLICATION USAGE

None.

RATIONALE

There are at least two serious problems with the semantics of *fork()* in a multi-threaded program. One problem has to do with state (for example, memory) covered by mutexes. Consider the case where one thread has a mutex locked and the state covered by that mutex is inconsistent while another thread calls *fork()*. In the child, the mutex is in the locked state (locked by a nonexistent thread and thus can never be unlocked). Having the child simply reinitialize the mutex is unsatisfactory since this approach does not resolve the question about how to correct or otherwise deal with the inconsistent state in the child.

It is suggested that programs that use *fork()* call an *exec* function very soon afterwards in the child process, thus resetting all states. In the meantime, only a short list of async-signal-safe library routines are promised to be available.

Unfortunately, this solution does not address the needs of multi-threaded libraries. Application programs may not be aware that a multi-threaded library is in use, and they feel free to call any number of library routines between the *fork()* and *exec* calls, just as they always have. Indeed, they may be extant single-threaded programs and cannot, therefore, be expected to obey new restrictions imposed by the threads library.

On the other hand, the multi-threaded library needs a way to protect its internal state during *fork()* in case it is re-entered later in the child process. The problem arises especially in multi-

threaded I/O libraries, which are almost sure to be invoked between the *fork()* and *exec* calls to effect I/O redirection. The solution may require locking mutex variables during *fork()*, or it may entail simply resetting the state in the child after the *fork()* processing completes.

The *pthread_atfork()* function was intended to provide multi-threaded libraries with a means to protect themselves from innocent application programs that call *fork()*, and to provide multi-threaded application programs with a standard mechanism for protecting themselves from *fork()* calls in a library routine or the application itself.

The expected usage was that the prepare handler would acquire all mutex locks and the other two fork handlers would release them.

For example, an application could have supplied a prepare routine that acquires the necessary mutexes the library maintains and supplied child and parent routines that release those mutexes, thus ensuring that the child would have got a consistent snapshot of the state of the library (and that no mutexes would have been left stranded). This is good in theory, but in reality not practical. Each and every mutex and lock in the process must be located and locked. Every component of a program including third-party components must participate and they must agree who is responsible for which mutex or lock. This is especially problematic for mutexes and locks in dynamically allocated memory. All mutexes and locks internal to the implementation must be locked, too. This possibly delays the thread calling *fork()* for a long time or even indefinitely since uses of these synchronization objects may not be under control of the application. A final problem to mention here is the problem of locking streams. At least the streams under control of the system (like *stdin*, *stdout*, *stderr*) must be protected by locking the stream with *flockfile()*. But the application itself could have done that, possibly in the same thread calling *fork()*. In this case, the process will deadlock.

Alternatively, some libraries might have been able to supply just a *child* routine that reinitializes the mutexes in the library and all associated states to some known value (for example, what it was when the image was originally executed). This approach is not possible, though, because implementations are allowed to fail **_init()* and **_destroy()* calls for mutexes and locks if the mutex or lock is still locked. In this case, the *child* routine is not able to reinitialize the mutexes and locks.

When *fork()* is called, only the calling thread is duplicated in the child process. Synchronization variables remain in the same state in the child as they were in the parent at the time *fork()* was called. Thus, for example, mutex locks may be held by threads that no longer exist in the child process, and any associated states may be inconsistent. The intention was that the parent process could have avoided this by explicit code that acquires and releases locks critical to the child via *pthread_atfork()*. In addition, any critical threads would have needed to be recreated and reinitialized to the proper state in the child (also via *pthread_atfork()*).

A higher-level package may acquire locks on its own data structures before invoking lower-level packages. Under this scenario, the order specified for fork handler calls allows a simple rule of initialization for avoiding package deadlock: a package initializes all packages on which it depends before it calls the *pthread_atfork()* function for itself.

As explained, there is no suitable solution for functionality which requires non-atomic operations to be protected through mutexes and locks. This is why the POSIX.1 standard since the 1996 release requires that the child process after *fork()* in a multi-threaded process only calls *async-signal-safe* interfaces.

50307 **FUTURE DIRECTIONS**

50308 None.

50309 **SEE ALSO**50310 *atexit()*, *exec*, *fork()*50311 XBD **<pthread.h>**, **<sys/types.h>**50312 **CHANGE HISTORY**

50313 First released in Issue 5. Derived from the POSIX Threads Extension.

50314 IEEE PASC Interpretation 1003.1c #4 is applied.

50315 **Issue 6**50316 The *pthread_atfork()* function is marked as part of the Threads option.50317 The **<pthread.h>** header is added to the SYNOPSIS.50318 **Issue 7**50319 The *pthread_atfork()* function is moved from the Threads option to the Base.

50320 SD5-XSH-ERN-145 is applied, updating the RATIONALE.

NAME

pthread_attr_destroy, pthread_attr_init — destroy and initialize the thread attributes object

SYNOPSIS

```
#include <pthread.h>
```

```
int pthread_attr_destroy(pthread_attr_t *attr);
```

```
int pthread_attr_init(pthread_attr_t *attr);
```

DESCRIPTION

The *pthread_attr_destroy()* function shall destroy a thread attributes object. An implementation may cause *pthread_attr_destroy()* to set *attr* to an implementation-defined invalid value. A destroyed *attr* attributes object can be reinitialized using *pthread_attr_init()*; the results of otherwise referencing the object after it has been destroyed are undefined.

The *pthread_attr_init()* function shall initialize a thread attributes object *attr* with the default value for all of the individual attributes used by a given implementation.

The resulting attributes object (possibly modified by setting individual attribute values) when used by *pthread_create()* defines the attributes of the thread created. A single attributes object can be used in multiple simultaneous calls to *pthread_create()*. Results are undefined if *pthread_attr_init()* is called specifying an already initialized *attr* attributes object.

The behavior is undefined if the value specified by the *attr* argument to *pthread_attr_destroy()* does not refer to an initialized thread attributes object.

RETURN VALUE

Upon successful completion, *pthread_attr_destroy()* and *pthread_attr_init()* shall return a value of 0; otherwise, an error number shall be returned to indicate the error.

ERRORS

The *pthread_attr_init()* function shall fail if:

[ENOMEM] Insufficient memory exists to initialize the thread attributes object.

These functions shall not return an error code of [EINTR].

EXAMPLES

None.

APPLICATION USAGE

None.

RATIONALE

Attributes objects are provided for threads, mutexes, and condition variables as a mechanism to support probable future standardization in these areas without requiring that the function itself be changed.

Attributes objects provide clean isolation of the configurable aspects of threads. For example, “stack size” is an important attribute of a thread, but it cannot be expressed portably. When porting a threaded program, stack sizes often need to be adjusted. The use of attributes objects can help by allowing the changes to be isolated in a single place, rather than being spread across every instance of thread creation.

Attributes objects can be used to set up “classes” of threads with similar attributes; for example, “threads with large stacks and high priority” or “threads with minimal stacks”. These classes can be defined in a single place and then referenced wherever threads need to be created. Changes to “class” decisions become straightforward, and detailed analysis of each *pthread_create()* call is not required.

The attributes objects are defined as opaque types as an aid to extensibility. If these objects had been specified as structures, adding new attributes would force recompilation of all multi-threaded programs when the attributes objects are extended; this might not be possible if different program components were supplied by different vendors.

Additionally, opaque attributes objects present opportunities for improving performance. Argument validity can be checked once when attributes are set, rather than each time a thread is created. Implementations often need to cache kernel objects that are expensive to create. Opaque attributes objects provide an efficient mechanism to detect when cached objects become invalid due to attribute changes.

Since assignment is not necessarily defined on a given opaque type, implementation-defined default values cannot be defined in a portable way. The solution to this problem is to allow attributes objects to be initialized dynamically by attributes object initialization functions, so that default values can be supplied automatically by the implementation.

The following proposal was provided as a suggested alternative to the supplied attributes:

1. Maintain the style of passing a parameter formed by the bitwise-inclusive OR of flags to the initialization routines (*pthread_create()*, *pthread_mutex_init()*, *pthread_cond_init()*). The parameter containing the flags should be an opaque type for extensibility. If no flags are set in the parameter, then the objects are created with default characteristics. An implementation may specify implementation-defined flag values and associated behavior.
2. If further specialization of mutexes and condition variables is necessary, implementations may specify additional procedures that operate on the **pthread_mutex_t** and **pthread_cond_t** objects (instead of on attributes objects).

The difficulties with this solution are:

1. A bitmask is not opaque if bits have to be set into bitvector attributes objects using explicitly-coded bitwise-inclusive OR operations. If the set of options exceeds an **int**, application programmers need to know the location of each bit. If bits are set or read by encapsulation (that is, *get* and *set* functions), then the bitmask is merely an implementation of attributes objects as currently defined and should not be exposed to the programmer.
2. Many attributes are not Boolean or very small integral values. For example, scheduling policy may be placed in 3-bit or 4-bit, but priority requires 5-bit or more, thereby taking up at least 8 bits out of a possible 16 bits on machines with 16-bit integers. Because of this, the bitmask can only reasonably control whether particular attributes are set or not, and it cannot serve as the repository of the value itself. The value needs to be specified as a function parameter (which is non-extensible), or by setting a structure field (which is non-opaque), or by *get* and *set* functions (making the bitmask a redundant addition to the attributes objects).

Stack size is defined as an optional attribute because the very notion of a stack is inherently machine-dependent. Some implementations may not be able to change the size of the stack, for example, and others may not need to because stack pages may be discontinuous and can be allocated and released on demand.

The attribute mechanism has been designed in large measure for extensibility. Future extensions to the attribute mechanism or to any attributes object defined in this volume of POSIX.1-2008 has to be done with care so as not to affect binary-compatibility.

Attributes objects, even if allocated by means of dynamic allocation functions such as *malloc()*,

may have their size fixed at compile time. This means, for example, a *pthread_create()* in an implementation with extensions to **pthread_attr_t** cannot look beyond the area that the binary application assumes is valid. This suggests that implementations should maintain a size field in the attributes object, as well as possibly version information, if extensions in different directions (possibly by different vendors) are to be accommodated.

If an implementation detects that the value specified by the *attr* argument to *pthread_attr_destroy()* does not refer to an initialized thread attributes object, it is recommended that the function should fail and report an [EINVAL] error.

If an implementation detects that the value specified by the *attr* argument to *pthread_attr_init()* refers to an already initialized thread attributes object, it is recommended that the function should fail and report an [EBUSY] error.

FUTURE DIRECTIONS

None.

SEE ALSO

pthread_attr_getstacksize(), *pthread_attr_getdetachstate()*, *pthread_create()*

XBD <pthread.h>

CHANGE HISTORY

First released in Issue 5. Included for alignment with the POSIX Threads Extension.

Issue 6

The *pthread_attr_destroy()* and *pthread_attr_init()* functions are marked as part of the Threads option.

IEEE PASC Interpretation 1003.1 #107 is applied, noting that the effect of initializing an already initialized thread attributes object is undefined.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/71 is applied, updating the ERRORS section to add the optional [EINVAL] error for the *pthread_attr_destroy()* function, and the optional [EBUSY] error for the *pthread_attr_init()* function.

Issue 7

The *pthread_attr_destroy()* and *pthread_attr_init()* functions are moved from the Threads option to the Base.

The [EINVAL] error for an uninitialized thread attributes object is removed; this condition results in undefined behavior.

The [EBUSY] error for an already initialized thread attributes object is removed; this condition results in undefined behavior.

NAME

pthread_attr_getdetachstate, pthread_attr_setdetachstate — get and set the detachstate attribute

SYNOPSIS

```
#include <pthread.h>

int pthread_attr_getdetachstate(const pthread_attr_t *attr,
                               int *detachstate);
int pthread_attr_setdetachstate(pthread_attr_t *attr, int detachstate);
```

DESCRIPTION

The *detachstate* attribute controls whether the thread is created in a detached state. If the thread is created detached, then use of the ID of the newly created thread by the *pthread_detach()* or *pthread_join()* function is an error.

The *pthread_attr_getdetachstate()* and *pthread_attr_setdetachstate()* functions, respectively, shall get and set the *detachstate* attribute in the *attr* object.

For *pthread_attr_getdetachstate()*, *detachstate* shall be set to either PTHREAD_CREATE_DETACHED or PTHREAD_CREATE_JOINABLE.

For *pthread_attr_setdetachstate()*, the application shall set *detachstate* to either PTHREAD_CREATE_DETACHED or PTHREAD_CREATE_JOINABLE.

A value of PTHREAD_CREATE_DETACHED shall cause all threads created with *attr* to be in the detached state, whereas using a value of PTHREAD_CREATE_JOINABLE shall cause all threads created with *attr* to be in the joinable state. The default value of the *detachstate* attribute shall be PTHREAD_CREATE_JOINABLE.

The behavior is undefined if the value specified by the *attr* argument to *pthread_attr_getdetachstate()* or *pthread_attr_setdetachstate()* does not refer to an initialized thread attributes object.

RETURN VALUE

Upon successful completion, *pthread_attr_getdetachstate()* and *pthread_attr_setdetachstate()* shall return a value of 0; otherwise, an error number shall be returned to indicate the error.

The *pthread_attr_getdetachstate()* function stores the value of the *detachstate* attribute in *detachstate* if successful.

ERRORS

The *pthread_attr_setdetachstate()* function shall fail if:

[EINVAL] The value of *detachstate* was not valid

These functions shall not return an error code of [EINTR].

EXAMPLES**Retrieving the detachstate Attribute**

This example shows how to obtain the *detachstate* attribute of a thread attribute object.

```
#include <pthread.h>

pthread_attr_t thread_attr;
int detachstate;
int rc;

/* code initializing thread_attr */
...
```

```

50486     rc = pthread_attr_getdetachstate (&thread_attr, &detachstate);
50487     if (rc!=0) {
50488         /* handle error */
50489         ...
50490     }
50491     else {
50492         /* legal values for detachstate are:
50493          * PTHREAD_CREATE_DETACHED or PTHREAD_CREATE_JOINABLE
50494          */
50495         ...
50496     }

```

APPLICATION USAGE

None.

RATIONALE

If an implementation detects that the value specified by the *attr* argument to *pthread_attr_getdetachstate()* or *pthread_attr_setdetachstate()* does not refer to an initialized thread attributes object, it is recommended that the function should fail and report an [EINVAL] error.

FUTURE DIRECTIONS

None.

SEE ALSO

pthread_attr_destroy(), *pthread_attr_getstacksize()*, *pthread_create()*

XBD <pthread.h>

CHANGE HISTORY

First released in Issue 5. Included for alignment with the POSIX Threads Extension.

Issue 6

The *pthread_attr_setdetachstate()* and *pthread_attr_getdetachstate()* functions are marked as part of the Threads option.

The normative text is updated to avoid use of the term “must” for application requirements.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/72 is applied, adding the example to the EXAMPLES section.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/73 is applied, updating the ERRORS section to include the optional [EINVAL] error.

Issue 7

The *pthread_attr_setdetachstate()* and *pthread_attr_getdetachstate()* functions are moved from the Threads option to the Base.

The [EINVAL] error for an uninitialized thread attributes object is removed; this condition results in undefined behavior.

NAME

pthread_attr_getguardsize, pthread_attr_setguardsize — get and set the thread guardsize attribute

SYNOPSIS

```
#include <pthread.h>

int pthread_attr_getguardsize(const pthread_attr_t *restrict attr,
                             size_t *restrict guardsize);
int pthread_attr_setguardsize(pthread_attr_t *attr,
                             size_t guardsize);
```

DESCRIPTION

The *pthread_attr_getguardsize()* function shall get the *guardsize* attribute in the *attr* object. This attribute shall be returned in the *guardsize* parameter.

The *pthread_attr_setguardsize()* function shall set the *guardsize* attribute in the *attr* object. The new value of this attribute shall be obtained from the *guardsize* parameter. If *guardsize* is zero, a guard area shall not be provided for threads created with *attr*. If *guardsize* is greater than zero, a guard area of at least size *guardsize* bytes shall be provided for each thread created with *attr*.

The *guardsize* attribute controls the size of the guard area for the created thread's stack. The *guardsize* attribute provides protection against overflow of the stack pointer. If a thread's stack is created with guard protection, the implementation allocates extra memory at the overflow end of the stack as a buffer against stack overflow of the stack pointer. If an application overflows into this buffer an error shall result (possibly in a SIGSEGV signal being delivered to the thread).

A conforming implementation may round up the value contained in *guardsize* to a multiple of the configurable system variable {PAGESIZE} (see <sys/mman.h>). If an implementation rounds up the value of *guardsize* to a multiple of {PAGESIZE}, a call to *pthread_attr_getguardsize()* specifying *attr* shall store in the *guardsize* parameter the guard size specified by the previous *pthread_attr_setguardsize()* function call.

The default value of the *guardsize* attribute is implementation-defined.

If the *stackaddr* attribute has been set (that is, the caller is allocating and managing its own thread stacks), the *guardsize* attribute shall be ignored and no protection shall be provided by the implementation. It is the responsibility of the application to manage stack overflow along with stack allocation and management in this case.

The behavior is undefined if the value specified by the *attr* argument to *pthread_attr_getguardsize()* or *pthread_attr_setguardsize()* does not refer to an initialized thread attributes object.

RETURN VALUE

If successful, the *pthread_attr_getguardsize()* and *pthread_attr_setguardsize()* functions shall return zero; otherwise, an error number shall be returned to indicate the error.

ERRORS

These functions shall fail if:

[EINVAL] The parameter *guardsize* is invalid.

These functions shall not return an error code of [EINTR].

EXAMPLES**Retrieving the guardsize Attribute**

This example shows how to obtain the *guardsize* attribute of a thread attribute object.

```
#include <pthread.h>

pthread_attr_t thread_attr;
size_t guardsize;
int rc;

/* code initializing thread_attr */
...

rc = pthread_attr_getguardsize (&thread_attr, &guardsize);
if (rc != 0) {
    /* handle error */
    ...
}
else {
    if (guardsize > 0) {
        /* a guard area of at least guardsize bytes is provided */
        ...
    }
    else {
        /* no guard area provided */
        ...
    }
}
```

APPLICATION USAGE

None.

RATIONALE

The *guardsize* attribute is provided to the application for two reasons:

1. Overflow protection can potentially result in wasted system resources. An application that creates a large number of threads, and which knows its threads never overflow their stack, can save system resources by turning off guard areas.
2. When threads allocate large data structures on the stack, large guard areas may be needed to detect stack overflow.

The default size of the guard area is left implementation-defined since on systems supporting very large page sizes, the overhead might be substantial if at least one guard page is required by default.

If an implementation detects that the value specified by the *attr* argument to *pthread_attr_getguardsize()* or *pthread_attr_setguardsize()* does not refer to an initialized thread attributes object, it is recommended that the function should fail and report an [EINVAL] error.

FUTURE DIRECTIONS

None.

SEE ALSO

XBD [<pthread.h>](#), [<sys/mman.h>](#)

CHANGE HISTORY

First released in Issue 5.

Issue 6

In the ERRORS section, a third [EINVAL] error condition is removed as it is covered by the second error condition.

The **restrict** keyword is added to the *pthread_attr_getguardsize()* prototype for alignment with the ISO/IEC 9899:1999 standard.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/74 is applied, updating the ERRORS section to remove the [EINVAL] error ("The attribute *attr* is invalid."), and replacing it with the optional [EINVAL] error.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/76 is applied, adding the example to the EXAMPLES section.

Issue 7

SD5-XSH-ERN-111 is applied, removing the reference to the *stack* attribute in the DESCRIPTION.

SD5-XSH-ERN-175 is applied, updating the DESCRIPTION to note that the default size of the guard area is implementation-defined.

The *pthread_attr_getguardsize()* and *pthread_attr_setguardsize()* functions are moved from the XSI option to the Base.

The [EINVAL] error for an uninitialized thread attributes object is removed; this condition results in undefined behavior.

NAME

pthread_attr_getinheritsched, pthread_attr_setinheritsched — get and set the inheritsched attribute (**REALTIME THREADS**)

SYNOPSIS

```
#include <pthread.h>

int pthread_attr_getinheritsched(const pthread_attr_t *restrict attr,
    int *restrict inheritsched);
int pthread_attr_setinheritsched(pthread_attr_t *attr,
    int inheritsched);
```

DESCRIPTION

The *pthread_attr_getinheritsched()* and *pthread_attr_setinheritsched()* functions, respectively, shall get and set the *inheritsched* attribute in the *attr* argument.

When the attributes objects are used by *pthread_create()*, the *inheritsched* attribute determines how the other scheduling attributes of the created thread shall be set.

The supported values of *inheritsched* shall be:

PTHREAD_INHERIT_SCHED

Specifies that the thread scheduling attributes shall be inherited from the creating thread, and the scheduling attributes in this *attr* argument shall be ignored.

PTHREAD_EXPLICIT_SCHED

Specifies that the thread scheduling attributes shall be set to the corresponding values from this attributes object.

The symbols **PTHREAD_INHERIT_SCHED** and **PTHREAD_EXPLICIT_SCHED** are defined in the **<pthread.h>** header.

The following thread scheduling attributes defined by POSIX.1-2008 are affected by the *inheritsched* attribute: scheduling policy (*schedpolicy*), scheduling parameters (*schedparam*), and scheduling contention scope (*contentionscope*).

The behavior is undefined if the value specified by the *attr* argument to *pthread_attr_getinheritsched()* or *pthread_attr_setinheritsched()* does not refer to an initialized thread attributes object.

RETURN VALUE

If successful, the *pthread_attr_getinheritsched()* and *pthread_attr_setinheritsched()* functions shall return zero; otherwise, an error number shall be returned to indicate the error.

ERRORS

The *pthread_attr_setinheritsched()* function shall fail if: +

[ENOTSUP] An attempt was made to set the attribute to an unsupported value. +

The *pthread_attr_setinheritsched()* function may fail if:

[EINVAL] The value of *inheritsched* is not valid. -

These functions shall not return an error code of [EINTR].

EXAMPLES

None.

APPLICATION USAGE

After these attributes have been set, a thread can be created with the specified attributes using *pthread_create()*. Using these routines does not affect the current running thread.

See [Section 2.9.4](#) (on page 514) for further details on thread scheduling attributes and their default settings.

RATIONALE

If an implementation detects that the value specified by the *attr* argument to *pthread_attr_getinheritsched()* or *pthread_attr_setinheritsched()* does not refer to an initialized thread attributes object, it is recommended that the function should fail and report an [EINVAL] error.

FUTURE DIRECTIONS

None.

SEE ALSO

pthread_attr_destroy(), *pthread_attr_getscope()*, *pthread_attr_getschedpolicy()*,
pthread_attr_getschedparam(), *pthread_create()*

XBD [<pthread.h>](#), [<sched.h>](#)

CHANGE HISTORY

First released in Issue 5. Included for alignment with the POSIX Threads Extension.

Marked as part of the Realtime Threads Feature Group.

Issue 6

The *pthread_attr_getinheritsched()* and *pthread_attr_setinheritsched()* functions are marked as part of the Threads and Thread Execution Scheduling options.

The [ENOSYS] error condition has been removed as stubs need not be provided if an implementation does not support the Thread Execution Scheduling option.

The **restrict** keyword is added to the *pthread_attr_getinheritsched()* prototype for alignment with the ISO/IEC 9899:1999 standard.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/75 is applied, clarifying the values of *inheritsched* in the DESCRIPTION and adding two optional [EINVAL] errors to the ERRORS section for checking when *attr* refers to an uninitialized thread attribute object.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/77 is applied, adding a reference to [Section 2.9.4](#) (on page 514) in the APPLICATION USAGE section.

Issue 7

The *pthread_attr_getinheritsched()* and *pthread_attr_setinheritsched()* functions are moved from the Threads option.

The [EINVAL] error for an uninitialized thread attributes object is removed; this condition results in undefined behavior.

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0450 [314] is applied.

+

NAME

pthread_attr_getschedparam, pthread_attr_setschedparam — get and set the schedparam attribute

SYNOPSIS

```
#include <pthread.h>

int pthread_attr_getschedparam(const pthread_attr_t *restrict attr,
                               struct sched_param *restrict param);
int pthread_attr_setschedparam(pthread_attr_t *restrict attr,
                               const struct sched_param *restrict param);
```

DESCRIPTION

The *pthread_attr_getschedparam()* and *pthread_attr_setschedparam()* functions, respectively, shall get and set the scheduling parameter attributes in the *attr* argument. The contents of the *param* structure are defined in the **<sched.h>** header. For the SCHED_FIFO and SCHED_RR policies, the only required member of *param* is *sched_priority*.

TSP

For the SCHED_SPORADIC policy, the required members of the *param* structure are *sched_priority*, *sched_ss_low_priority*, *sched_ss_repl_period*, *sched_ss_init_budget*, and *sched_ss_max_repl*. The specified *sched_ss_repl_period* must be greater than or equal to the specified *sched_ss_init_budget* for the function to succeed; if it is not, then the function shall fail. The value of *sched_ss_max_repl* shall be within the inclusive range [1, {SS_REPL_MAX}] for the function to succeed; if not, the function shall fail. It is unspecified whether the *sched_ss_repl_period* and *sched_ss_init_budget* values are stored as provided by this function or are rounded to align with the resolution of the clock being used.

The behavior is undefined if the value specified by the *attr* argument to *pthread_attr_getschedparam()* or *pthread_attr_setschedparam()* does not refer to an initialized thread attributes object.

RETURN VALUE

If successful, the *pthread_attr_getschedparam()* and *pthread_attr_setschedparam()* functions shall return zero; otherwise, an error number shall be returned to indicate the error.

ERRORS

The *pthread_attr_setschedparam()* function shall fail if:

[ENOTSUP] An attempt was made to set the attribute to an unsupported value.

The *pthread_attr_setschedparam()* function may fail if:

[EINVAL] The value of *param* is not valid.

These functions shall not return an error code of [EINTR].

EXAMPLES

None.

APPLICATION USAGE

After these attributes have been set, a thread can be created with the specified attributes using *pthread_create()*. Using these routines does not affect the current running thread.

RATIONALE

If an implementation detects that the value specified by the *attr* argument to *pthread_attr_getschedparam()* or *pthread_attr_setschedparam()* does not refer to an initialized thread attributes object, it is recommended that the function should fail and report an [EINVAL] error.

FUTURE DIRECTIONS

None.

SEE ALSO

pthread_attr_destroy(), *pthread_attr_getscope()*, *pthread_attr_getinheritsched()*,
pthread_attr_getschedpolicy(), *pthread_create()*

XBD <pthread.h>, <sched.h>

CHANGE HISTORY

First released in Issue 5. Included for alignment with the POSIX Threads Extension.

Issue 6

The *pthread_attr_getschedparam()* and *pthread_attr_setschedparam()* functions are marked as part of the Threads option.

The SCHED_SPORADIC scheduling policy is added for alignment with IEEE Std 1003.1d-1999.

The **restrict** keyword is added to the *pthread_attr_getschedparam()* and *pthread_attr_setschedparam()* prototypes for alignment with the ISO/IEC 9899:1999 standard.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/78 is applied, updating the ERRORS section to include optional errors for the case when *attr* refers to an uninitialized thread attribute object.

Issue 7

The *pthread_attr_getschedparam()* and *pthread_attr_setschedparam()* functions are moved from the Threads option to the Base.

Austin Group Interpretation 1003.1-2001 #119 is applied, clarifying the accuracy requirements for the *sched_ss_repl_period* and *sched_ss_init_budget* values.

The [EINVAL] error for an uninitialized thread attributes object is removed; this condition results in undefined behavior.

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0451 [314] is applied.

+

50772 **NAME**

50773 pthread_attr_getschedpolicy, pthread_attr_setschedpolicy — get and set the schedpolicy
 50774 attribute (**REALTIME THREADS**)

50775 **SYNOPSIS**

```
50776 TPS    #include <pthread.h>
50777
50777     int pthread_attr_getschedpolicy(const pthread_attr_t *restrict attr,
50778                                     int *restrict policy);
50779     int pthread_attr_setschedpolicy(pthread_attr_t *attr, int policy);
```

50780 **DESCRIPTION**

50781 The *pthread_attr_getschedpolicy()* and *pthread_attr_setschedpolicy()* functions, respectively, shall
 50782 get and set the *schedpolicy* attribute in the *attr* argument.

50783 The supported values of *policy* shall include SCHED_FIFO, SCHED_RR, and SCHED_OTHER,
 50784 which are defined in the **<sched.h>** header. When threads executing with the scheduling policy
 50785 TSP SCHED_FIFO, SCHED_RR, or SCHED_SPORADIC are waiting on a mutex, they shall acquire
 50786 the mutex in priority order when the mutex is unlocked.

50787 The behavior is undefined if the value specified by the *attr* argument to
 50788 *pthread_attr_getschedpolicy()* or *pthread_attr_setschedpolicy()* does not refer to an initialized thread
 50789 attributes object.

50790 **RETURN VALUE**

50791 If successful, the *pthread_attr_getschedpolicy()* and *pthread_attr_setschedpolicy()* functions shall
 50792 return zero; otherwise, an error number shall be returned to indicate the error.

50793 **ERRORS**

50794 The *pthread_attr_setschedpolicy()* function shall fail if: +

50795 [ENOTSUP] An attempt was made to set the attribute to an unsupported value. +

50796 The *pthread_attr_setschedpolicy()* function may fail if:

50797 [EINVAL] The value of *policy* is not valid. -

50798 These functions shall not return an error code of [EINTR].

50799 **EXAMPLES**

50800 None.

50801 **APPLICATION USAGE**

50802 After these attributes have been set, a thread can be created with the specified attributes using
 50803 *pthread_create()*. Using these routines does not affect the current running thread.

50804 See [Section 2.9.4](#) (on page 514) for further details on thread scheduling attributes and their
 50805 default settings.

50806 **RATIONALE**

50807 If an implementation detects that the value specified by the *attr* argument to
 50808 *pthread_attr_getschedpolicy()* or *pthread_attr_setschedpolicy()* does not refer to an initialized thread
 50809 attributes object, it is recommended that the function should fail and report an [EINVAL] error.

50810 **FUTURE DIRECTIONS**

50811 None.

SEE ALSO

pthread_attr_destroy(), *pthread_attr_getscope()*, *pthread_attr_getinheritsched()*,
pthread_attr_getschedparam(), *pthread_create()*

XBD <pthread.h>, <sched.h>

CHANGE HISTORY

First released in Issue 5. Included for alignment with the POSIX Threads Extension.

Marked as part of the Realtime Threads Feature Group.

Issue 6

The *pthread_attr_getschedpolicy()* and *pthread_attr_setschedpolicy()* functions are marked as part of the Threads and Thread Execution Scheduling options.

The [ENOSYS] error condition has been removed as stubs need not be provided if an implementation does not support the Thread Execution Scheduling option.

The SCHED_SPORADIC scheduling policy is added for alignment with IEEE Std 1003.1d-1999.

The **restrict** keyword is added to the *pthread_attr_getschedpolicy()* prototype for alignment with the ISO/IEC 9899: 1999 standard.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/79 is applied, adding a reference to [Section 2.9.4](#) (on page 514) in the APPLICATION USAGE section.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/80 is applied, updating the ERRORS section to include optional errors for the case when *attr* refers to an uninitialized thread attribute object.

Issue 7

The *pthread_attr_getschedpolicy()* and *pthread_attr_setschedpolicy()* functions are moved from the Threads option.

The [EINVAL] error for an uninitialized thread attributes object is removed; this condition results in undefined behavior.

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0452 [314] is applied.

+

NAME

pthread_attr_getscope, pthread_attr_setscope — get and set the contentionscope attribute (REALTIME THREADS)

SYNOPSIS

```

#include <pthread.h>

int pthread_attr_getscope(const pthread_attr_t *restrict attr,
    int *restrict contentionscope);
int pthread_attr_setscope(pthread_attr_t *attr, int contentionscope);

```

DESCRIPTION

The *pthread_attr_getscope()* and *pthread_attr_setscope()* functions, respectively, shall get and set the *contentionscope* attribute in the *attr* object.

The *contentionscope* attribute may have the values PTHREAD_SCOPE_SYSTEM, signifying system scheduling contention scope, or PTHREAD_SCOPE_PROCESS, signifying process scheduling contention scope. The symbols PTHREAD_SCOPE_SYSTEM and PTHREAD_SCOPE_PROCESS are defined in the **<pthread.h>** header.

The behavior is undefined if the value specified by the *attr* argument to *pthread_attr_getscope()* or *pthread_attr_setscope()* does not refer to an initialized thread attributes object.

RETURN VALUE

If successful, the *pthread_attr_getscope()* and *pthread_attr_setscope()* functions shall return zero; otherwise, an error number shall be returned to indicate the error.

ERRORS

The *pthread_attr_setscope()* function shall fail if:

[ENOTSUP] An attempt was made to set the attribute to an unsupported value.

The *pthread_attr_setscope()* function may fail if:

[EINVAL] The value of *contentionscope* is not valid.

These functions shall not return an error code of [EINTR].

EXAMPLES

None.

APPLICATION USAGE

After these attributes have been set, a thread can be created with the specified attributes using *pthread_create()*. Using these routines does not affect the current running thread.

See [Section 2.9.4](#) (on page 514) for further details on thread scheduling attributes and their default settings.

RATIONALE

If an implementation detects that the value specified by the *attr* argument to *pthread_attr_getscope()* or *pthread_attr_setscope()* does not refer to an initialized thread attributes object, it is recommended that the function should fail and report an [EINVAL] error.

FUTURE DIRECTIONS

None.

SEE ALSO

pthread_attr_destroy(), *pthread_attr_getinheritsched()*, *pthread_attr_getschedpolicy()*,
pthread_attr_getschedparam(), *pthread_create()*

XBD <pthread.h>, <sched.h>

CHANGE HISTORY

First released in Issue 5. Included for alignment with the POSIX Threads Extension.

Marked as part of the Realtime Threads Feature Group.

Issue 6

The *pthread_attr_getscope()* and *pthread_attr_setscope()* functions are marked as part of the Threads and Thread Execution Scheduling options.

The [ENOSYS] error condition has been removed as stubs need not be provided if an implementation does not support the Thread Execution Scheduling option.

The **restrict** keyword is added to the *pthread_attr_getscope()* prototype for alignment with the ISO/IEC 9899:1999 standard.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/81 is applied, adding a reference to [Section 2.9.4](#) (on page 514) in the APPLICATION USAGE section.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/82 is applied, updating the ERRORS section to include optional errors for the case when *attr* refers to an uninitialized thread attribute object.

Issue 7

The *pthread_attr_getscope()* and *pthread_attr_setscope()* functions are moved from the Threads option.

The [EINVAL] error for an uninitialized thread attributes object is removed; this condition results in undefined behavior.

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0453 [314] is applied.

+

NAME

pthread_attr_getstack, pthread_attr_setstack — get and set stack attributes

SYNOPSIS

```
TSA TSS #include <pthread.h>
```

```
int pthread_attr_getstack(const pthread_attr_t *restrict attr,
    void **restrict stackaddr, size_t *restrict stacksize);
int pthread_attr_setstack(pthread_attr_t *attr, void *stackaddr,
    size_t stacksize);
```

DESCRIPTION

The *pthread_attr_getstack()* and *pthread_attr_setstack()* functions, respectively, shall get and set the thread creation stack attributes *stackaddr* and *stacksize* in the *attr* object.

The stack attributes specify the area of storage to be used for the created thread's stack. The base (lowest addressable byte) of the storage shall be *stackaddr*, and the size of the storage shall be *stacksize* bytes. The *stacksize* shall be at least {PTHREAD_STACK_MIN}. The *pthread_attr_setstack()* function may fail with [EINVAL] if *stackaddr* does not meet implementation-defined alignment requirements. All pages within the stack described by *stackaddr* and *stacksize* shall be both readable and writable by the thread.

If the *pthread_attr_getstack()* function is called before the *stackaddr* attribute has been set, the behavior is unspecified.

The behavior is undefined if the value specified by the *attr* argument to *pthread_attr_getstack()* or *pthread_attr_setstack()* does not refer to an initialized thread attributes object.

RETURN VALUE

Upon successful completion, these functions shall return a value of 0; otherwise, an error number shall be returned to indicate the error.

The *pthread_attr_getstack()* function shall store the stack attribute values in *stackaddr* and *stacksize* if successful.

ERRORS

The *pthread_attr_setstack()* function shall fail if:

[EINVAL] The value of *stacksize* is less than {PTHREAD_STACK_MIN} or exceeds an implementation-defined limit.

The *pthread_attr_setstack()* function may fail if:

[EINVAL] The value of *stackaddr* does not have proper alignment to be used as a stack, or ((**char** *)*stackaddr* + *stacksize*) lacks proper alignment.

[EACCES] The stack page(s) described by *stackaddr* and *stacksize* are not both readable and writable by the thread.

These functions shall not return an error code of [EINTR].

EXAMPLES

None.

APPLICATION USAGE

These functions are appropriate for use by applications in an environment where the stack for a thread must be placed in some particular region of memory.

While it might seem that an application could detect stack overflow by providing a protected page outside the specified stack region, this cannot be done portably. Implementations are free to place the thread's initial stack pointer anywhere within the specified region to accommodate the machine's stack pointer behavior and allocation requirements. Furthermore, on some architectures, such as the IA-64, "overflow" might mean that two separate stack pointers allocated within the region will overlap somewhere in the middle of the region.

After a successful call to *pthread_attr_setstack()*, the storage area specified by the *stackaddr* parameter is under the control of the implementation, as described in [Section 2.9.8](#) (on page 521).

The specification of the *stackaddr* attribute presents several ambiguities that make portable use of these functions impossible. For example, the standard allows implementations to impose arbitrary alignment requirements on *stackaddr*. Applications cannot assume that a buffer obtained from *malloc()* is suitably aligned. Note that although the *stacksize* value passed to *pthread_attr_setstack()* must satisfy alignment requirements, the same is not true for *pthread_attr_setstacksize()* where the implementation must increase the specified size if necessary to achieve the proper alignment.

RATIONALE

If an implementation detects that the value specified by the *attr* argument to *pthread_attr_getstack()* or *pthread_attr_setstack()* does not refer to an initialized thread attributes object, it is recommended that the function should fail and report an [EINVAL] error.

FUTURE DIRECTIONS

None.

SEE ALSO

pthread_attr_destroy(), *pthread_attr_getdetachstate()*, *pthread_attr_getstacksize()*, *pthread_create()*

XBD [<limits.h>](#), [<pthread.h>](#)

CHANGE HISTORY

First released in Issue 6. Developed as part of the XSI option and brought into the BASE by IEEE PASC Interpretation 1003.1 #101.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/83 is applied, updating the APPLICATION USAGE section to refer to [Section 2.9.8](#) (on page 521).

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC/D6/84 is applied, updating the ERRORS section to include optional errors for the case when *attr* refers to an uninitialized thread attribute object.

Issue 7

SD5-XSH-ERN-66 is applied, correcting the use of *attr* in the [EINVAL] error condition.

Austin Group Interpretation 1003.1-2001 #057 is applied, clarifying the behavior if the function is called before the *stackaddr* attribute is set.

SD5-XSH-ERN-157 is applied, updating the APPLICATION USAGE section.

The description of the *stackaddr* attribute is updated in the DESCRIPTION and APPLICATION USAGE sections.

50982
50983

The [EINVAL] error for an uninitialized thread attributes object is removed; this condition results in undefined behavior.

50984 **NAME**

50985 pthread_attr_getstacksize, pthread_attr_setstacksize — get and set the stacksize attribute

50986 **SYNOPSIS**

```

50987 TSS      #include <pthread.h>
50988
50988      int pthread_attr_getstacksize(const pthread_attr_t *restrict attr,
50989                                   size_t *restrict stacksize);
50990      int pthread_attr_setstacksize(pthread_attr_t *attr, size_t stacksize);

```

50991 **DESCRIPTION**

50992 The *pthread_attr_getstacksize()* and *pthread_attr_setstacksize()* functions, respectively, shall get and
 50993 set the thread creation *stacksize* attribute in the *attr* object.

50994 The *stacksize* attribute shall define the minimum stack size (in bytes) allocated for the created
 50995 threads stack.

50996 The behavior is undefined if the value specified by the *attr* argument to
 50997 *pthread_attr_getstacksize()* or *pthread_attr_setstacksize()* does not refer to an initialized thread
 50998 attributes object.

50999 **RETURN VALUE**

51000 Upon successful completion, *pthread_attr_getstacksize()* and *pthread_attr_setstacksize()* shall
 51001 return a value of 0; otherwise, an error number shall be returned to indicate the error.

51002 The *pthread_attr_getstacksize()* function stores the *stacksize* attribute value in *stacksize* if
 51003 successful.

51004 **ERRORS**

51005 The *pthread_attr_setstacksize()* function shall fail if:

51006 [EINVAL] The value of *stacksize* is less than {PTHREAD_STACK_MIN} or exceeds a
 51007 system-imposed limit.

51008 These functions shall not return an error code of [EINTR].

51009 **EXAMPLES**

51010 None.

51011 **APPLICATION USAGE**

51012 None.

51013 **RATIONALE**

51014 If an implementation detects that the value specified by the *attr* argument to
 51015 *pthread_attr_getstacksize()* or *pthread_attr_setstacksize()* does not refer to an initialized thread
 51016 attributes object, it is recommended that the function should fail and report an [EINVAL] error.

51017 **FUTURE DIRECTIONS**

51018 None.

51019 **SEE ALSO**

51020 [*pthread_attr_destroy\(\)*](#), [*pthread_attr_getdetachstate\(\)*](#), [*pthread_create\(\)*](#)

51021 XBD [*<limits.h>*](#), [*<pthread.h>*](#)

51022 **CHANGE HISTORY**

51023 First released in Issue 5. Included for alignment with the POSIX Threads Extension.

51024 Issue 6

51025 The *pthread_attr_getstacksize()* and *pthread_attr_setstacksize()* functions are marked as part of the
51026 Threads and Thread Stack Size Attribute options.

51027 The **restrict** keyword is added to the *pthread_attr_getstacksize()* prototype for alignment with the
51028 ISO/IEC 9899:1999 standard.

51029 IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/43 is applied, correcting the margin code
51030 in the SYNOPSIS from TSA to TSS and updating the CHANGE HISTORY from “Thread Stack
51031 Address Attribute” option to “Thread Stack Size Attribute” option.

51032 IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/87 is applied, updating the ERRORS
51033 section to include optional errors for the case when *attr* refers to an uninitialized thread attribute
51034 object.

51035 Issue 7

51036 The *pthread_attr_getstacksize()* and *pthread_attr_setstacksize()* functions are moved from the
51037 Threads option.

51038 The [EINVAL] error for an uninitialized thread attributes object is removed; this condition
51039 results in undefined behavior.

51040 **NAME**

51041 pthread_attr_init — initialize the thread attributes object

51042 **SYNOPSIS**

51043 #include <pthread.h>

51044 int pthread_attr_init(pthread_attr_t *attr);

51045 **DESCRIPTION**51046 Refer to *pthread_attr_destroy()*.

51047 **NAME**

51048 pthread_attr_setdetachstate — set the detachstate attribute

51049 **SYNOPSIS**

51050 #include <pthread.h>

51051 int pthread_attr_setdetachstate(pthread_attr_t *attr, int detachstate);

51052 **DESCRIPTION**

51053 Refer to *pthread_attr_getdetachstate()*.

51054 **NAME**

51055 pthread_attr_setguardsize — set the thread guardsize attribute

51056 **SYNOPSIS**

51057 #include <pthread.h>

51058 int pthread_attr_setguardsize(pthread_attr_t *attr,
51059 size_t guardsize);51060 **DESCRIPTION**51061 Refer to *pthread_attr_getguardsize()*.

51062 NAME

51063 pthread_attr_setinheritsched — set the inheritsched attribute (**REALTIME THREADS**)

51064 SYNOPSIS

```
51065 TPS    #include <pthread.h>
51066
51066    int pthread_attr_setinheritsched(pthread_attr_t *attr,
51067    int inheritsched);
```

51068 DESCRIPTION

51069 Refer to *pthread_attr_getinheritsched()*.

51070 **NAME**

51071 pthread_attr_setschedparam — set the schedparam attribute

51072 **SYNOPSIS**

51073 #include <pthread.h>

```
51074 int pthread_attr_setschedparam(pthread_attr_t *restrict attr,  
51075                               const struct sched_param *restrict param);
```

51076 **DESCRIPTION**51077 Refer to *pthread_attr_getschedparam()*.

51078 NAME

51079 pthread_attr_setschedpolicy — set the schedpolicy attribute (**REALTIME THREADS**)

51080 SYNOPSIS

```
51081 TPS    #include <pthread.h>
51082        int pthread_attr_setschedpolicy(pthread_attr_t *attr, int policy);
```

51083 DESCRIPTION

51084 Refer to *pthread_attr_getschedpolicy()*.

51085 **NAME**51086 pthread_attr_setscope — set the contention scope attribute (**REALTIME THREADS**)51087 **SYNOPSIS**

```
51088 TPS    #include <pthread.h>
51089        int pthread_attr_setscope(pthread_attr_t *attr, int contention_scope);
```

51090 **DESCRIPTION**51091 Refer to *pthread_attr_getscope()*.

51092 NAME

51093 pthread_attr_setstack — set the stack attribute

51094 SYNOPSIS

51095 TSA TSS `#include <pthread.h>`

```
51096 int pthread_attr_setstack(pthread_attr_t *attr, void *stackaddr,  
51097                          size_t stacksize);
```

51098 DESCRIPTION

51099 Refer to *pthread_attr_getstack()*.

51100 **NAME**

51101 pthread_attr_setstacksize — set the stacksize attribute

51102 **SYNOPSIS**51103 TSS `#include <pthread.h>`51104 `int pthread_attr_setstacksize(pthread_attr_t *attr, size_t stacksize);`51105 **DESCRIPTION**51106 Refer to *pthread_attr_getstacksize()*.

51107 **NAME**

51108 pthread_barrier_destroy, pthread_barrier_init — destroy and initialize a barrier object

51109 **SYNOPSIS**

```
51110 #include <pthread.h>
51111 int pthread_barrier_destroy(pthread_barrier_t *barrier);
51112 int pthread_barrier_init(pthread_barrier_t *restrict barrier,
51113     const pthread_barrierattr_t *restrict attr, unsigned count);
```

51114 **DESCRIPTION**

51115 The *pthread_barrier_destroy()* function shall destroy the barrier referenced by *barrier* and release
 51116 any resources used by the barrier. The effect of subsequent use of the barrier is undefined until
 51117 the barrier is reinitialized by another call to *pthread_barrier_init()*. An implementation may use
 51118 this function to set *barrier* to an invalid value. The results are undefined if
 51119 *pthread_barrier_destroy()* is called when any thread is blocked on the barrier, or if this function is
 51120 called with an uninitialized barrier.

51121 The *pthread_barrier_init()* function shall allocate any resources required to use the barrier
 51122 referenced by *barrier* and shall initialize the barrier with attributes referenced by *attr*. If *attr* is
 51123 NULL, the default barrier attributes shall be used; the effect is the same as passing the address of
 51124 a default barrier attributes object. The results are undefined if *pthread_barrier_init()* is called
 51125 when any thread is blocked on the barrier (that is, has not returned from the
 51126 *pthread_barrier_wait()* call). The results are undefined if a barrier is used without first being
 51127 initialized. The results are undefined if *pthread_barrier_init()* is called specifying an already
 51128 initialized barrier.

51129 The *count* argument specifies the number of threads that must call *pthread_barrier_wait()* before
 51130 any of them successfully return from the call. The value specified by *count* must be greater than
 51131 zero.

51132 If the *pthread_barrier_init()* function fails, the barrier shall not be initialized and the contents of
 51133 *barrier* are undefined.

51134 Only the object referenced by *barrier* may be used for performing synchronization. The result of
 51135 referring to copies of that object in calls to *pthread_barrier_destroy()* or *pthread_barrier_wait()* is
 51136 undefined.

51137 **RETURN VALUE**

51138 Upon successful completion, these functions shall return zero; otherwise, an error number shall
 51139 be returned to indicate the error.

51140 **ERRORS**

51141 The *pthread_barrier_init()* function shall fail if:

51142 [EAGAIN] The system lacks the necessary resources to initialize another barrier.

51143 [EINVAL] The value specified by *count* is equal to zero.

51144 [ENOMEM] Insufficient memory exists to initialize the barrier.

51145 These functions shall not return an error code of [EINTR].

EXAMPLES

None.

APPLICATION USAGE

None.

RATIONALE

If an implementation detects that the value specified by the *barrier* argument to *pthread_barrier_destroy()* does not refer to an initialized barrier object, it is recommended that the function should fail and report an [EINVAL] error.

If an implementation detects that the value specified by the *attr* argument to *pthread_barrier_init()* does not refer to an initialized barrier attributes object, it is recommended that the function should fail and report an [EINVAL] error.

If an implementation detects that the value specified by the *barrier* argument to *pthread_barrier_destroy()* or *pthread_barrier_init()* refers to a barrier that is in use (for example, in a *pthread_barrier_wait()* call) by another thread, or detects that the value specified by the *barrier* argument to *pthread_barrier_init()* refers to an already initialized barrier object, it is recommended that the function should fail and report an [EBUSY] error.

FUTURE DIRECTIONS

None.

SEE ALSO

pthread_barrier_wait()

XBD <pthread.h>

CHANGE HISTORY

First released in Issue 6. Derived from IEEE Std 1003.1j-2000.

Issue 7

The *pthread_barrier_destroy()* and *pthread_barrier_init()* functions are moved from the Barriers option to the Base.

The [EINVAL] error for an uninitialized barrier object and an uninitialized barrier attributes object is removed; this condition results in undefined behavior.

The [EBUSY] error for a barrier that is in use or an already initialized barrier object is removed; this condition results in undefined behavior.

51176 **NAME**

51177 pthread_barrier_wait — synchronize at a barrier

51178 **SYNOPSIS**

51179 #include <pthread.h>

51180 int pthread_barrier_wait(pthread_barrier_t *barrier);

51181 **DESCRIPTION**

51182 The *pthread_barrier_wait()* function shall synchronize participating threads at the barrier
51183 referenced by *barrier*. The calling thread shall block until the required number of threads have
51184 called *pthread_barrier_wait()* specifying the barrier.

51185 When the required number of threads have called *pthread_barrier_wait()* specifying the barrier,
51186 the constant PTHREAD_BARRIER_SERIAL_THREAD shall be returned to one unspecified
51187 thread and zero shall be returned to each of the remaining threads. At this point, the barrier shall
51188 be reset to the state it had as a result of the most recent *pthread_barrier_init()* function that
51189 referenced it.

51190 The constant PTHREAD_BARRIER_SERIAL_THREAD is defined in <pthread.h> and its value
51191 shall be distinct from any other value returned by *pthread_barrier_wait()*.

51192 The results are undefined if this function is called with an uninitialized barrier.

51193 If a signal is delivered to a thread blocked on a barrier, upon return from the signal handler the
51194 thread shall resume waiting at the barrier if the barrier wait has not completed (that is, if the
51195 required number of threads have not arrived at the barrier during the execution of the signal
51196 handler); otherwise, the thread shall continue as normal from the completed barrier wait. Until
51197 the thread in the signal handler returns from it, it is unspecified whether other threads may
51198 proceed past the barrier once they have all reached it.

51199 A thread that has blocked on a barrier shall not prevent any unblocked thread that is eligible to
51200 use the same processing resources from eventually making forward progress in its execution.
51201 Eligibility for processing resources shall be determined by the scheduling policy.

51202 **RETURN VALUE**

51203 Upon successful completion, the *pthread_barrier_wait()* function shall return
51204 PTHREAD_BARRIER_SERIAL_THREAD for a single (arbitrary) thread synchronized at the
51205 barrier and zero for each of the other threads. Otherwise, an error number shall be returned to
51206 indicate the error.

51207 **ERRORS**

51208 This function shall not return an error code of [EINTR].

51209 **EXAMPLES**

51210 None.

51211 **APPLICATION USAGE**

51212 Applications using this function may be subject to priority inversion, as discussed in XBD
51213 [Section 3.287](#) (on page 79).

51214 **RATIONALE**

51215 If an implementation detects that the value specified by the *barrier* argument to
51216 *pthread_barrier_wait()* does not refer to an initialized barrier object, it is recommended that the
51217 function should fail and report an [EINVAL] error.

51218 **FUTURE DIRECTIONS**

51219 None.

51220 **SEE ALSO**51221 [*pthread_barrier_destroy\(\)*](#)51222 XBD [Section 3.287](#) (on page 79), [Section 4.11](#) (on page 110), [**<pthread.h>**](#)51223 **CHANGE HISTORY**

51224 First released in Issue 6. Derived from IEEE Std 1003.1j-2000.

51225 In the SYNOPSIS, the inclusion of **<sys/types.h>** is no longer required.51226 **Issue 7**51227 The *pthread_barrier_wait()* function is moved from the Barriers option to the Base.51228 The [EINVAL] error for an uninitialized barrier object is removed; this condition results in
51229 undefined behavior.

NAME

pthread_barrierattr_destroy, pthread_barrierattr_init — destroy and initialize the barrier attributes object

SYNOPSIS

```
#include <pthread.h>

int pthread_barrierattr_destroy(pthread_barrierattr_t *attr);
int pthread_barrierattr_init(pthread_barrierattr_t *attr);
```

DESCRIPTION

The *pthread_barrierattr_destroy()* function shall destroy a barrier attributes object. A destroyed *attr* attributes object can be reinitialized using *pthread_barrierattr_init()*; the results of otherwise referencing the object after it has been destroyed are undefined. An implementation may cause *pthread_barrierattr_destroy()* to set the object referenced by *attr* to an invalid value.

The *pthread_barrierattr_init()* function shall initialize a barrier attributes object *attr* with the default value for all of the attributes defined by the implementation.

If *pthread_barrierattr_init()* is called specifying an already initialized *attr* attributes object, the results are undefined.

After a barrier attributes object has been used to initialize one or more barriers, any function affecting the attributes object (including destruction) shall not affect any previously initialized barrier.

The behavior is undefined if the value specified by the *attr* argument to *pthread_barrierattr_destroy()* does not refer to an initialized barrier attributes object.

RETURN VALUE

If successful, the *pthread_barrierattr_destroy()* and *pthread_barrierattr_init()* functions shall return zero; otherwise, an error number shall be returned to indicate the error.

ERRORS

The *pthread_barrierattr_init()* function shall fail if:

[ENOMEM] Insufficient memory exists to initialize the barrier attributes object.

These functions shall not return an error code of [EINTR].

EXAMPLES

None.

APPLICATION USAGE

None.

RATIONALE

If an implementation detects that the value specified by the *attr* argument to *pthread_barrierattr_destroy()* does not refer to an initialized barrier attributes object, it is recommended that the function should fail and report an [EINVAL] error.

FUTURE DIRECTIONS

None.

SEE ALSO

pthread_barrierattr_getpshared()

XBD <pthread.h>

CHANGE HISTORY

51271 First released in Issue 6. Derived from IEEE Std 1003.1j-2000.

51272 In the SYNOPSIS, the inclusion of **<sys/types.h>** is no longer required.

Issue 7

51274 The *pthread_barrierattr_destroy()* and *pthread_barrierattr_init()* functions are moved from the Barriers option to the Base.

51275 The [EINVAL] error for an uninitialized barrier attributes object is removed; this condition
51276 results in undefined behavior.

NAME

pthread_barrierattr_getpshared, pthread_barrierattr_setpshared — get and set the process-shared attribute of the barrier attributes object

SYNOPSIS

```
TSH #include <pthread.h>

int pthread_barrierattr_getpshared(const pthread_barrierattr_t
    *restrict attr, int *restrict pshared);
int pthread_barrierattr_setpshared(pthread_barrierattr_t *attr,
    int pshared);
```

DESCRIPTION

The *pthread_barrierattr_getpshared()* function shall obtain the value of the *process-shared* attribute from the attributes object referenced by *attr*. The *pthread_barrierattr_setpshared()* function shall set the *process-shared* attribute in an initialized attributes object referenced by *attr*.

The *process-shared* attribute is set to `PTHREAD_PROCESS_SHARED` to permit a barrier to be operated upon by any thread that has access to the memory where the barrier is allocated. If the *process-shared* attribute is `PTHREAD_PROCESS_PRIVATE`, the barrier shall only be operated upon by threads created within the same process as the thread that initialized the barrier; if threads of different processes attempt to operate on such a barrier, the behavior is undefined. The default value of the attribute shall be `PTHREAD_PROCESS_PRIVATE`. Both constants `PTHREAD_PROCESS_SHARED` and `PTHREAD_PROCESS_PRIVATE` are defined in `<pthread.h>`.

Additional attributes, their default values, and the names of the associated functions to get and set those attribute values are implementation-defined.

The behavior is undefined if the value specified by the *attr* argument to *pthread_barrierattr_getpshared()* or *pthread_barrierattr_setpshared()* does not refer to an initialized barrier attributes object.

RETURN VALUE

If successful, the *pthread_barrierattr_getpshared()* function shall return zero and store the value of the *process-shared* attribute of *attr* into the object referenced by the *pshared* parameter. Otherwise, an error number shall be returned to indicate the error.

If successful, the *pthread_barrierattr_setpshared()* function shall return zero; otherwise, an error number shall be returned to indicate the error.

ERRORS

The *pthread_barrierattr_setpshared()* function may fail if:

[EINVAL] The new value specified for the *process-shared* attribute is not one of the legal values `PTHREAD_PROCESS_SHARED` or `PTHREAD_PROCESS_PRIVATE`.

These functions shall not return an error code of [EINTR].

51316 EXAMPLES

51317 None.

51318 APPLICATION USAGE

51319 The *pthread_barrierattr_getpshared()* and *pthread_barrierattr_setpshared()* functions are part of the
51320 Thread Process-Shared Synchronization option and need not be provided on all
51321 implementations.

51322 RATIONALE

51323 If an implementation detects that the value specified by the *attr* argument to
51324 *pthread_barrierattr_getpshared()* or *pthread_barrierattr_setpshared()* does not refer to an initialized
51325 barrier attributes object, it is recommended that the function should fail and report an [EINVAL]
51326 error.

51327 FUTURE DIRECTIONS

51328 None.

51329 SEE ALSO

51330 *pthread_barrier_destroy()*, *pthread_barrierattr_destroy()*

51331 XBD <pthread.h>

51332 CHANGE HISTORY

51333 First released in Issue 6. Derived from IEEE Std 1003.1j-2000

51334 Issue 7

51335 The *pthread_barrierattr_getpshared()* and *pthread_barrierattr_setpshared()* functions are moved from
51336 the Barriers option.

51337 The [EINVAL] error for an uninitialized barrier attributes object is removed; this condition
51338 results in undefined behavior.

51339 NAME

51340 pthread_barrierattr_init — initialize the barrier attributes object

51341 SYNOPSIS

51342 #include <pthread.h>

51343 int pthread_barrierattr_init(pthread_barrierattr_t *attr);

51344 DESCRIPTION

51345 Refer to *pthread_barrierattr_destroy()*.

51346 **NAME**

51347 pthread_barrierattr_setpshared — set the process-shared attribute of the barrier attributes object

51348 **SYNOPSIS**

```
51349 TSH      #include <pthread.h>
51350          int pthread_barrierattr_setpshared(pthread_barrierattr_t *attr,
51351          int pshared);
```

51352 **DESCRIPTION**51353 Refer to *pthread_barrierattr_getpshared()*.

51354 **NAME**

51355 pthread_cancel — cancel execution of a thread

51356 **SYNOPSIS**

51357 #include <pthread.h>

51358 int pthread_cancel(pthread_t thread);

51359 **DESCRIPTION**

51360 The *pthread_cancel()* function shall request that *thread* be canceled. The target thread's
51361 cancelability state and type determines when the cancellation takes effect. When the cancellation
51362 is acted on, the cancellation cleanup handlers for *thread* shall be called. When the last
51363 cancellation cleanup handler returns, the thread-specific data destructor functions shall be called
51364 for *thread*. When the last destructor function returns, *thread* shall be terminated.

51365 The cancellation processing in the target thread shall run asynchronously with respect to the
51366 calling thread returning from *pthread_cancel()*.

51367 **RETURN VALUE**

51368 If successful, the *pthread_cancel()* function shall return zero; otherwise, an error number shall be
51369 returned to indicate the error.

51370 **ERRORS**51371 The *pthread_cancel()* function shall not return an error code of [EINTR].51372 **EXAMPLES**

51373 None.

51374 **APPLICATION USAGE**

51375 None.

51376 **RATIONALE**

51377 Two alternative functions were considered for sending the cancellation notification to a thread.
51378 One would be to define a new SIGCANCEL signal that had the cancellation semantics when
51379 delivered; the other was to define the new *pthread_cancel()* function, which would trigger the
51380 cancellation semantics.

51381 The advantage of a new signal was that so much of the delivery criteria were identical to that
51382 used when trying to deliver a signal that making cancellation notification a signal was seen as
51383 consistent. Indeed, many implementations implement cancellation using a special signal. On the
51384 other hand, there would be no signal functions that could be used with this signal except
51385 *pthread_kill()*, and the behavior of the delivered cancellation signal would be unlike any
51386 previously existing defined signal.

51387 The benefits of a special function include the recognition that this signal would be defined
51388 because of the similar delivery criteria and that this is the only common behavior between a
51389 cancellation request and a signal. In addition, the cancellation delivery mechanism does not
51390 have to be implemented as a signal. There are also strong, if not stronger, parallels with
51391 language exception mechanisms than with signals that are potentially obscured if the delivery
51392 mechanism is visibly closer to signals.

51393 In the end, it was considered that as there were so many exceptions to the use of the new signal
51394 with existing signals functions it would be misleading. A special function has resolved this
51395 problem. This function was carefully defined so that an implementation wishing to provide the
51396 cancellation functions on top of signals could do so. The special function also means that
51397 implementations are not obliged to implement cancellation with signals.

51398 If an implementation detects use of a thread ID after the end of its lifetime, it is recommended
51399 that the function should fail and report an [ESRCH] error.

51400 **FUTURE DIRECTIONS**

51401 None.

51402 **SEE ALSO**51403 *pthread_exit()*, *pthread_cond_timedwait()*, *pthread_join()*, *pthread_setcancelstate()*

51404 XBD <pthread.h>

51405 **CHANGE HISTORY**

51406 First released in Issue 5. Included for alignment with the POSIX Threads Extension.

51407 **Issue 6**51408 The *pthread_cancel()* function is marked as part of the Threads option.51409 **Issue 7**51410 The *pthread_cancel()* function is moved from the Threads option to the Base.

51411 Austin Group Interpretation 1003.1-2001 #142 is applied, removing the [ESRCH] error condition.

51412 **NAME**

51413 pthread_cleanup_pop, pthread_cleanup_push — establish cancellation handlers

51414 **SYNOPSIS**

```
51415 #include <pthread.h>
51416 void pthread_cleanup_pop(int execute);
51417 void pthread_cleanup_push(void (*routine)(void*), void *arg);
```

51418 **DESCRIPTION**

51419 The *pthread_cleanup_pop()* function shall remove the routine at the top of the calling thread's
 51420 cancellation cleanup stack and optionally invoke it (if *execute* is non-zero).

51421 The *pthread_cleanup_push()* function shall push the specified cancellation cleanup handler *routine*
 51422 onto the calling thread's cancellation cleanup stack. The cancellation cleanup handler shall be
 51423 popped from the cancellation cleanup stack and invoked with the argument *arg* when:

- 51424 • The thread exits (that is, calls *pthread_exit()*).
- 51425 • The thread acts upon a cancellation request.
- 51426 • The thread calls *pthread_cleanup_pop()* with a non-zero *execute* argument.

51427 These functions may be implemented as macros. The application shall ensure that they appear
 51428 as statements, and in pairs within the same lexical scope (that is, the *pthread_cleanup_push()*
 51429 macro may be thought to expand to a token list whose first token is '{' with
 51430 *pthread_cleanup_pop()* expanding to a token list whose last token is the corresponding '}').

51431 The effect of calling *longjmp()* or *siglongjmp()* is undefined if there have been any calls to
 51432 *pthread_cleanup_push()* or *pthread_cleanup_pop()* made without the matching call since the jump
 51433 buffer was filled. The effect of calling *longjmp()* or *siglongjmp()* from inside a cancellation
 51434 cleanup handler is also undefined unless the jump buffer was also filled in the cancellation
 51435 cleanup handler.

51436 The effect of the use of **return**, **break**, **continue**, and **goto** to prematurely leave a code block
 51437 described by a pair of *pthread_cleanup_push()* and *pthread_cleanup_pop()* functions calls is
 51438 undefined.

51439 **RETURN VALUE**

51440 The *pthread_cleanup_push()* and *pthread_cleanup_pop()* functions shall not return a value.

51441 **ERRORS**

51442 No errors are defined.

51443 These functions shall not return an error code of [EINTR].

51444 **EXAMPLES**

51445 The following is an example using thread primitives to implement a cancelable, writers-priority
 51446 read-write lock:

```
51447 typedef struct {
51448     pthread_mutex_t lock;
51449     pthread_cond_t rcond,
51450     wcond;
51451     int lock_count; /* < 0 .. Held by writer. */
51452                   /* > 0 .. Held by lock_count readers. */
51453                   /* = 0 .. Held by nobody. */
51454     int waiting_writers; /* Count of waiting writers. */
51455 } rwlock;
```

```

51456 void
51457 waiting_reader_cleanup(void *arg)
51458 {
51459     rwlock *l;
51460
51461     l = (rwlock *) arg;
51462     pthread_mutex_unlock(&l->lock);
51463
51464 void
51465 lock_for_read(rwlock *l)
51466 {
51467     pthread_mutex_lock(&l->lock);
51468     pthread_cleanup_push(waiting_reader_cleanup, l);
51469     while ((l->lock_count < 0) || (l->waiting_writers != 0))
51470         pthread_cond_wait(&l->rcond, &l->lock);
51471     l->lock_count++;
51472     /*
51473      * Note the pthread_cleanup_pop executes
51474      * waiting_reader_cleanup.
51475      */
51476     pthread_cleanup_pop(1);
51477
51478 void
51479 release_read_lock(rwlock *l)
51480 {
51481     pthread_mutex_lock(&l->lock);
51482     if (--l->lock_count == 0)
51483         pthread_cond_signal(&l->wcond);
51484     pthread_mutex_unlock(&l->lock);
51485
51486 void
51487 waiting_writer_cleanup(void *arg)
51488 {
51489     rwlock *l;
51490
51491     l = (rwlock *) arg;
51492     if ((--l->waiting_writers == 0) && (l->lock_count >= 0)) {
51493         /*
51494          * This only happens if we have been canceled. If the
51495          * lock is not held by a writer, there may be readers who
51496          * were blocked because waiting_writers was positive; they
51497          * can now be unblocked.
51498          */
51499         pthread_cond_broadcast(&l->rcond);
51500     }
51501     pthread_mutex_unlock(&l->lock);
51502
51503 void
51504 lock_for_write(rwlock *l)
51505 {
51506     pthread_mutex_lock(&l->lock);

```

```

51505         l->waiting_writers++;
51506         pthread_cleanup_push(waiting_writer_cleanup, l);
51507         while (l->lock_count != 0)
51508             pthread_cond_wait(&l->wcond, &l->lock);
51509         l->lock_count = -1;
51510     /*
51511      * Note the pthread_cleanup_pop executes
51512      * waiting_writer_cleanup.
51513      */
51514     pthread_cleanup_pop(1);
51515 }

51516 void
51517 release_write_lock(rwlock *l)
51518 {
51519     pthread_mutex_lock(&l->lock);
51520     l->lock_count = 0;
51521     if (l->waiting_writers == 0)
51522         pthread_cond_broadcast(&l->rcond);
51523     else
51524         pthread_cond_signal(&l->wcond);
51525     pthread_mutex_unlock(&l->lock);
51526 }

51527 /*
51528  * This function is called to initialize the read/write lock.
51529  */
51530 void
51531 initialize_rwlock(rwlock *l)
51532 {
51533     pthread_mutex_init(&l->lock, pthread_mutexattr_default);
51534     pthread_cond_init(&l->wcond, pthread_condattr_default);
51535     pthread_cond_init(&l->rcond, pthread_condattr_default);
51536     l->lock_count = 0;
51537     l->waiting_writers = 0;
51538 }

51539 reader_thread()
51540 {
51541     lock_for_read(&lock);
51542     pthread_cleanup_push(release_read_lock, &lock);
51543     /*
51544      * Thread has read lock.
51545      */
51546     pthread_cleanup_pop(1);
51547 }

51548 writer_thread()
51549 {
51550     lock_for_write(&lock);
51551     pthread_cleanup_push(release_write_lock, &lock);
51552     /*
51553      * Thread has write lock.
51554      */

```

```

51555     pthread_cleanup_pop(1);
51556 }

```

51557 APPLICATION USAGE

51558 The two routines that push and pop cancellation cleanup handlers, *pthread_cleanup_push()* and
 51559 *pthread_cleanup_pop()*, can be thought of as left and right-parentheses. They always need to be
 51560 matched.

51561 RATIONALE

51562 The restriction that the two routines that push and pop cancellation cleanup handlers,
 51563 *pthread_cleanup_push()* and *pthread_cleanup_pop()*, have to appear in the same lexical scope
 51564 allows for efficient macro or compiler implementations and efficient storage management. A
 51565 sample implementation of these routines as macros might look like this:

```

51566 #define pthread_cleanup_push(rtn,arg) { \
51567     struct _pthread_handler_rec __cleanup_handler, *__head; \
51568     __cleanup_handler.rtn = rtn; \
51569     __cleanup_handler.arg = arg; \
51570     (void) pthread_getspecific(_pthread_handler_key, &__head); \
51571     __cleanup_handler.next = *__head; \
51572     *__head = &__cleanup_handler;
51573
51574 #define pthread_cleanup_pop(ex) \
51575     *__head = __cleanup_handler.next; \
51576     if (ex) (*__cleanup_handler.rtn)(__cleanup_handler.arg); \
51577 }

```

51577 A more ambitious implementation of these routines might do even better by allowing the
 51578 compiler to note that the cancellation cleanup handler is a constant and can be expanded inline.

51579 This volume of POSIX.1-2008 currently leaves unspecified the effect of calling *longjmp()* from a
 51580 signal handler executing in a POSIX System Interfaces function. If an implementation wants to
 51581 allow this and give the programmer reasonable behavior, the *longjmp()* function has to call all
 51582 cancellation cleanup handlers that have been pushed but not popped since the time *setjmp()* was
 51583 called.

51584 Consider a multi-threaded function called by a thread that uses signals. If a signal were
 51585 delivered to a signal handler during the operation of *qsort()* and that handler were to call
 51586 *longjmp()* (which, in turn, did *not* call the cancellation cleanup handlers) the helper threads
 51587 created by the *qsort()* function would not be canceled. Instead, they would continue to execute
 51588 and write into the argument array even though the array might have been popped off the stack.

51589 Note that the specified cleanup handling mechanism is especially tied to the C language and,
 51590 while the requirement for a uniform mechanism for expressing cleanup is language-
 51591 independent, the mechanism used in other languages may be quite different. In addition, this
 51592 mechanism is really only necessary due to the lack of a real exception mechanism in the C
 51593 language, which would be the ideal solution.

51594 There is no notion of a cancellation cleanup-safe function. If an application has no cancellation
 51595 points in its signal handlers, blocks any signal whose handler may have cancellation points
 51596 while calling async-unsafe functions, or disables cancellation while calling async-unsafe
 51597 functions, all functions may be safely called from cancellation cleanup routines.

51598 **FUTURE DIRECTIONS**

51599 None.

51600 **SEE ALSO**51601 *pthread_cancel()*, *pthread_setcancelstate()*

51602 XBD <pthread.h>

51603 **CHANGE HISTORY**

51604 First released in Issue 5. Included for alignment with the POSIX Threads Extension.

51605 **Issue 6**51606 The *pthread_cleanup_pop()* and *pthread_cleanup_push()* functions are marked as part of the
51607 Threads option.

51608 The APPLICATION USAGE section is added.

51609 The normative text is updated to avoid use of the term “must” for application requirements.

51610 IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/88 is applied, updating the
51611 DESCRIPTION to describe the consequences of prematurely leaving a code block defined by the
51612 *pthread_cleanup_push()* and *pthread_cleanup_pop()* functions.51613 **Issue 7**51614 The *pthread_cleanup_pop()* and *pthread_cleanup_push()* functions are moved from the Threads
51615 option to the Base.

51616 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0454 [229] is applied.

+

NAME

pthread_cond_broadcast, pthread_cond_signal — broadcast or signal a condition

SYNOPSIS

```
#include <pthread.h>

int pthread_cond_broadcast(pthread_cond_t *cond);
int pthread_cond_signal(pthread_cond_t *cond);
```

DESCRIPTION

These functions shall unblock threads blocked on a condition variable.

The *pthread_cond_broadcast()* function shall unblock all threads currently blocked on the specified condition variable *cond*.

The *pthread_cond_signal()* function shall unblock at least one of the threads that are blocked on the specified condition variable *cond* (if any threads are blocked on *cond*).

If more than one thread is blocked on a condition variable, the scheduling policy shall determine the order in which threads are unblocked. When each thread unblocked as a result of a *pthread_cond_broadcast()* or *pthread_cond_signal()* returns from its call to *pthread_cond_wait()* or *pthread_cond_timedwait()*, the thread shall own the mutex with which it called *pthread_cond_wait()* or *pthread_cond_timedwait()*. The thread(s) that are unblocked shall contend for the mutex according to the scheduling policy (if applicable), and as if each had called *pthread_mutex_lock()*.

The *pthread_cond_broadcast()* or *pthread_cond_signal()* functions may be called by a thread whether or not it currently owns the mutex that threads calling *pthread_cond_wait()* or *pthread_cond_timedwait()* have associated with the condition variable during their waits; however, if predictable scheduling behavior is required, then that mutex shall be locked by the thread calling *pthread_cond_broadcast()* or *pthread_cond_signal()*.

The *pthread_cond_broadcast()* and *pthread_cond_signal()* functions shall have no effect if there are no threads currently blocked on *cond*.

The behavior is undefined if the value specified by the *cond* argument to *pthread_cond_broadcast()* or *pthread_cond_signal()* does not refer to an initialized condition variable.

RETURN VALUE

If successful, the *pthread_cond_broadcast()* and *pthread_cond_signal()* functions shall return zero; otherwise, an error number shall be returned to indicate the error.

ERRORS

These functions shall not return an error code of [EINTR].

EXAMPLES

None.

APPLICATION USAGE

The *pthread_cond_broadcast()* function is used whenever the shared-variable state has been changed in a way that more than one thread can proceed with its task. Consider a single producer/multiple consumer problem, where the producer can insert multiple items on a list that is accessed one item at a time by the consumers. By calling the *pthread_cond_broadcast()* function, the producer would notify all consumers that might be waiting, and thereby the application would receive more throughput on a multi-processor. In addition, *pthread_cond_broadcast()* makes it easier to implement a read-write lock. The *pthread_cond_broadcast()* function is needed in order to wake up all waiting readers when a writer releases its lock. Finally, the two-phase commit algorithm can use this broadcast function to notify all clients of an impending transaction commit.

It is not safe to use the *pthread_cond_signal()* function in a signal handler that is invoked asynchronously. Even if it were safe, there would still be a race between the test of the Boolean *pthread_cond_wait()* that could not be efficiently eliminated.

Mutexes and condition variables are thus not suitable for releasing a waiting thread by signaling from code running in a signal handler.

RATIONALE

If an implementation detects that the value specified by the *cond* argument to *pthread_cond_broadcast()* or *pthread_cond_signal()* does not refer to an initialized condition variable, it is recommended that the function should fail and report an [EINVAL] error.

Multiple Awakenings by Condition Signal

On a multi-processor, it may be impossible for an implementation of *pthread_cond_signal()* to avoid the unblocking of more than one thread blocked on a condition variable. For example, consider the following partial implementation of *pthread_cond_wait()* and *pthread_cond_signal()*, executed by two threads in the order given. One thread is trying to wait on the condition variable, another is concurrently executing *pthread_cond_signal()*, while a third thread is already waiting.

```
pthread_cond_wait(mutex, cond):
    value = cond->value; /* 1 */
    pthread_mutex_unlock(mutex); /* 2 */
    pthread_mutex_lock(cond->mutex); /* 10 */
    if (value == cond->value) { /* 11 */
        me->next_cond = cond->waiter;
        cond->waiter = me;
        pthread_mutex_unlock(cond->mutex);
        unable_to_run(me);
    } else
        pthread_mutex_unlock(cond->mutex); /* 12 */
    pthread_mutex_lock(mutex); /* 13 */

pthread_cond_signal(cond):
    pthread_mutex_lock(cond->mutex); /* 3 */
    cond->value++; /* 4 */
    if (cond->waiter) { /* 5 */
        sleeper = cond->waiter; /* 6 */
        cond->waiter = sleeper->next_cond; /* 7 */
        able_to_run(sleeper); /* 8 */
    }
    pthread_mutex_unlock(cond->mutex); /* 9 */
```

The effect is that more than one thread can return from its call to *pthread_cond_wait()* or *pthread_cond_timedwait()* as a result of one call to *pthread_cond_signal()*. This effect is called “spurious wakeup”. Note that the situation is self-correcting in that the number of threads that are so awakened is finite; for example, the next thread to call *pthread_cond_wait()* after the sequence of events above blocks.

While this problem could be resolved, the loss of efficiency for a fringe condition that occurs only rarely is unacceptable, especially given that one has to check the predicate associated with a condition variable anyway. Correcting this problem would unnecessarily reduce the degree of concurrency in this basic building block for all higher-level synchronization operations.

An added benefit of allowing spurious wakeups is that applications are forced to code a

51710 predicate-testing-loop around the condition wait. This also makes the application tolerate
51711 superfluous condition broadcasts or signals on the same condition variable that may be coded in
51712 some other part of the application. The resulting applications are thus more robust. Therefore,
51713 POSIX.1-2008 explicitly documents that spurious wakeups may occur.

51714 FUTURE DIRECTIONS

51715 None.

51716 SEE ALSO

51717 [*pthread_cond_destroy\(\)*](#), [*pthread_cond_timedwait\(\)*](#)

51718 XBD [Section 4.11](#) (on page 110), [**<pthread.h>**](#)

51719 CHANGE HISTORY

51720 First released in Issue 5. Included for alignment with the POSIX Threads Extension.

51721 Issue 6

51722 The *pthread_cond_broadcast()* and *pthread_cond_signal()* functions are marked as part of the
51723 Threads option.

51724 The APPLICATION USAGE section is added.

51725 Issue 7

51726 The *pthread_cond_broadcast()* and *pthread_cond_signal()* functions are moved from the Threads
51727 option to the Base.

51728 The [EINVAL] error for an uninitialized condition variable is removed; this condition results in
51729 undefined behavior.

NAME

pthread_cond_destroy, pthread_cond_init — destroy and initialize condition variables

SYNOPSIS

```
#include <pthread.h>

int pthread_cond_destroy(pthread_cond_t *cond);
int pthread_cond_init(pthread_cond_t *restrict cond,
    const pthread_condattr_t *restrict attr);
pthread_cond_t cond = PTHREAD_COND_INITIALIZER;
```

DESCRIPTION

The *pthread_cond_destroy()* function shall destroy the given condition variable specified by *cond*; the object becomes, in effect, uninitialized. An implementation may cause *pthread_cond_destroy()* to set the object referenced by *cond* to an invalid value. A destroyed condition variable object can be reinitialized using *pthread_cond_init()*; the results of otherwise referencing the object after it has been destroyed are undefined.

It shall be safe to destroy an initialized condition variable upon which no threads are currently blocked. Attempting to destroy a condition variable upon which other threads are currently blocked results in undefined behavior.

The *pthread_cond_init()* function shall initialize the condition variable referenced by *cond* with attributes referenced by *attr*. If *attr* is NULL, the default condition variable attributes shall be used; the effect is the same as passing the address of a default condition variable attributes object. Upon successful initialization, the state of the condition variable shall become initialized.

Only *cond* itself may be used for performing synchronization. The result of referring to copies of *cond* in calls to *pthread_cond_wait()*, *pthread_cond_timedwait()*, *pthread_cond_signal()*, *pthread_cond_broadcast()*, and *pthread_cond_destroy()* is undefined.

Attempting to initialize an already initialized condition variable results in undefined behavior.

In cases where default condition variable attributes are appropriate, the macro PTHREAD_COND_INITIALIZER can be used to initialize condition variables. The effect shall be equivalent to dynamic initialization by a call to *pthread_cond_init()* with parameter *attr* specified as NULL, except that no error checks are performed.

The behavior is undefined if the value specified by the *cond* argument to *pthread_cond_destroy()* does not refer to an initialized condition variable.

The behavior is undefined if the value specified by the *attr* argument to *pthread_cond_init()* does not refer to an initialized condition variable attributes object.

RETURN VALUE

If successful, the *pthread_cond_destroy()* and *pthread_cond_init()* functions shall return zero; otherwise, an error number shall be returned to indicate the error.

ERRORS

The *pthread_cond_init()* function shall fail if:

[EAGAIN] The system lacked the necessary resources (other than memory) to initialize another condition variable.

[ENOMEM] Insufficient memory exists to initialize the condition variable.

These functions shall not return an error code of [EINTR].

EXAMPLES

A condition variable can be destroyed immediately after all the threads that are blocked on it are awakened. For example, consider the following code:

```

struct list {
    pthread_mutex_t lm;
    ...
}

struct elt {
    key k;
    int busy;
    pthread_cond_t notbusy;
    ...
}

/* Find a list element and reserve it. */
struct elt *
list_find(struct list *lp, key k)
{
    struct elt *ep;

    pthread_mutex_lock(&lp->lm);
    while ((ep = find_elt(l, k) != NULL) && ep->busy)
        pthread_cond_wait(&ep->notbusy, &lp->lm);
    if (ep != NULL)
        ep->busy = 1;
    pthread_mutex_unlock(&lp->lm);
    return(ep);
}

delete_elt(struct list *lp, struct elt *ep)
{
    pthread_mutex_lock(&lp->lm);
    assert(ep->busy);
    ... remove ep from list ...
    ep->busy = 0; /* Paranoid. */
    (A) pthread_cond_broadcast(&ep->notbusy);
    pthread_mutex_unlock(&lp->lm);
    (B) pthread_cond_destroy(&ep->notbusy);
    free(ep);
}

```

In this example, the condition variable and its list element may be freed (line B) immediately after all threads waiting for it are awakened (line A), since the mutex and the code ensure that no other thread can touch the element to be deleted.

APPLICATION USAGE

None.

RATIONALE

If an implementation detects that the value specified by the *cond* argument to *pthread_cond_destroy()* does not refer to an initialized condition variable, it is recommended that the function should fail and report an [EINVAL] error.

If an implementation detects that the value specified by the *cond* argument to

51819 *pthread_cond_destroy()* or *pthread_cond_init()* refers to a condition variable that is in use (for
51820 example, in a *pthread_cond_wait()* call) by another thread, or detects that the value specified by
51821 the *cond* argument to *pthread_cond_init()* refers to an already initialized condition variable, it is
51822 recommended that the function should fail and report an [EBUSY] error.

51823 If an implementation detects that the value specified by the *attr* argument to *pthread_cond_init()*
51824 does not refer to an initialized condition variable attributes object, it is recommended that the
51825 function should fail and report an [EINVAL] error.

51826 See also *pthread_mutex_destroy()*.

51827 FUTURE DIRECTIONS

51828 None.

51829 SEE ALSO

51830 *pthread_cond_broadcast()*, *pthread_cond_timedwait()*, *pthread_mutex_destroy()*

51831 XBD <pthread.h>

51832 CHANGE HISTORY

51833 First released in Issue 5. Included for alignment with the POSIX Threads Extension.

51834 Issue 6

51835 The *pthread_cond_destroy()* and *pthread_cond_init()* functions are marked as part of the Threads
51836 option.

51837 IEEE PASC Interpretation 1003.1c #34 is applied, updating the DESCRIPTION.

51838 The **restrict** keyword is added to the *pthread_cond_init()* prototype for alignment with the
51839 ISO/IEC 9899:1999 standard.

51840 Issue 7

51841 The *pthread_cond_destroy()* and *pthread_cond_init()* functions are moved from the Threads option
51842 to the Base.

51843 The [EINVAL] error for an uninitialized condition variable and an uninitialized condition
51844 variable attributes object is removed; this condition results in undefined behavior.

51845 The [EBUSY] error for a condition variable already in use or an already initialized condition
51846 variable is removed; this condition results in undefined behavior.

51847 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0455 [70] is applied.

+

51848 **NAME**

51849 pthread_cond_signal — signal a condition

51850 **SYNOPSIS**

51851 #include <pthread.h>

51852 int pthread_cond_signal(pthread_cond_t *cond);

51853 **DESCRIPTION**51854 Refer to *pthread_cond_broadcast()*.

NAME

pthread_cond_timedwait, pthread_cond_wait — wait on a condition

SYNOPSIS

```
#include <pthread.h>

int pthread_cond_timedwait(pthread_cond_t *restrict cond,
    pthread_mutex_t *restrict mutex,
    const struct timespec *restrict abstime);
int pthread_cond_wait(pthread_cond_t *restrict cond,
    pthread_mutex_t *restrict mutex);
```

DESCRIPTION

The *pthread_cond_timedwait()* and *pthread_cond_wait()* functions shall block on a condition variable. The application shall ensure that these functions are called with *mutex* locked by the calling thread; otherwise, an error (for PTHREAD_MUTEX_ERRORCHECK and robust mutexes) or undefined behavior (for other mutexes) results.

These functions atomically release *mutex* and cause the calling thread to block on the condition variable *cond*; atomically here means “atomically with respect to access by another thread to the mutex and then the condition variable”. That is, if another thread is able to acquire the mutex after the about-to-block thread has released it, then a subsequent call to *pthread_cond_broadcast()* or *pthread_cond_signal()* in that thread shall behave as if it were issued after the about-to-block thread has blocked.

Upon successful return, the mutex shall have been locked and shall be owned by the calling thread. If *mutex* is a robust mutex where an owner terminated while holding the lock and the state is recoverable, the mutex shall be acquired even though the function returns an error code.

When using condition variables there is always a Boolean predicate involving shared variables associated with each condition wait that is true if the thread should proceed. Spurious wakeups from the *pthread_cond_timedwait()* or *pthread_cond_wait()* functions may occur. Since the return from *pthread_cond_timedwait()* or *pthread_cond_wait()* does not imply anything about the value of this predicate, the predicate should be re-evaluated upon such return.

When a thread waits on a condition variable, having specified a particular mutex to either the *pthread_cond_timedwait()* or the *pthread_cond_wait()* operation, a dynamic binding is formed between that mutex and condition variable that remains in effect as long as at least one thread is blocked on the condition variable. During this time, the effect of an attempt by any thread to wait on that condition variable using a different mutex is undefined. Once all waiting threads have been unblocked (as by the *pthread_cond_broadcast()* operation), the next wait operation on that condition variable shall form a new dynamic binding with the mutex specified by that wait operation. Even though the dynamic binding between condition variable and mutex may be removed or replaced between the time a thread is unblocked from a wait on the condition variable and the time that it returns to the caller or begins cancellation cleanup, the unblocked thread shall always re-acquire the mutex specified in the condition wait operation call from which it is returning.

A condition wait (whether timed or not) is a cancellation point. When the cancelability type of a thread is set to PTHREAD_CANCEL_DEFERRED, a side-effect of acting upon a cancellation request while in a condition wait is that the mutex is (in effect) re-acquired before calling the first cancellation cleanup handler. The effect is as if the thread were unblocked, allowed to execute up to the point of returning from the call to *pthread_cond_timedwait()* or *pthread_cond_wait()*, but at that point notices the cancellation request and instead of returning to the caller of *pthread_cond_timedwait()* or *pthread_cond_wait()*, starts the thread cancellation activities, which includes calling cancellation cleanup handlers.

A thread that has been unblocked because it has been canceled while blocked in a call to *pthread_cond_timedwait()* or *pthread_cond_wait()* shall not consume any condition signal that may be directed concurrently at the condition variable if there are other threads blocked on the condition variable.

The *pthread_cond_timedwait()* function shall be equivalent to *pthread_cond_wait()*, except that an error is returned if the absolute time specified by *abstime* passes (that is, system time equals or exceeds *abstime*) before the condition *cond* is signaled or broadcasted, or if the absolute time specified by *abstime* has already been passed at the time of the call. When such timeouts occur, *pthread_cond_timedwait()* shall nonetheless release and re-acquire the mutex referenced by *mutex*, and may consume a condition signal directed concurrently at the condition variable.

The condition variable shall have a clock attribute which specifies the clock that shall be used to measure the time specified by the *abstime* argument. The *pthread_cond_timedwait()* function is also a cancellation point.

If a signal is delivered to a thread waiting for a condition variable, upon return from the signal handler the thread resumes waiting for the condition variable as if it was not interrupted, or it shall return zero due to spurious wakeup.

The behavior is undefined if the value specified by the *cond* or *mutex* argument to these functions does not refer to an initialized condition variable or an initialized mutex object, respectively.

51922 RETURN VALUE

Except in the case of [ETIMEDOUT], all these error checks shall act as if they were performed immediately at the beginning of processing for the function and shall cause an error return, in effect, prior to modifying the state of the mutex specified by *mutex* or the condition variable specified by *cond*.

Upon successful completion, a value of zero shall be returned; otherwise, an error number shall be returned to indicate the error.

51929 ERRORS

These functions shall fail if:

[ENOTRECOVERABLE]

The state protected by the mutex is not recoverable.

[EOWNERDEAD]

The mutex is a robust mutex and the process containing the previous owning thread terminated while holding the mutex lock. The mutex lock shall be acquired by the calling thread and it is up to the new owner to make the state consistent.

[EPERM]

The mutex type is PTHREAD_MUTEX_ERRORCHECK or the mutex is a robust mutex, and the current thread does not own the mutex.

The *pthread_cond_timedwait()* function shall fail if:

[ETIMEDOUT] The time specified by *abstime* to *pthread_cond_timedwait()* has passed.

[EINVAL] The *abstime* argument specified a nanosecond value less than zero or greater than or equal to 1000 million.

51944 These functions may fail if:

51945 [EOWNERDEAD]

51946 The mutex is a robust mutex and the previous owning thread terminated
51947 while holding the mutex lock. The mutex lock shall be acquired by the calling
51948 thread and it is up to the new owner to make the state consistent.

51949 These functions shall not return an error code of [EINTR].

51950 EXAMPLES

51951 None.

51952 APPLICATION USAGE

51953 Applications that have assumed that non-zero return values are errors will need updating for
51954 use with robust mutexes, since a valid return for a thread acquiring a mutex which is protecting
51955 a currently inconsistent state is [EOWNERDEAD]. Applications that do not check the error
51956 returns, due to ruling out the possibility of such errors arising, should not use robust mutexes. If
51957 an application is supposed to work with normal and robust mutexes, it should check all return
51958 values for error conditions and if necessary take appropriate action.

51959 RATIONALE

51960 If an implementation detects that the value specified by the *cond* argument to
51961 *pthread_cond_timedwait()* or *pthread_cond_wait()* does not refer to an initialized condition
51962 variable, or detects that the value specified by the *mutex* argument to *pthread_cond_timedwait()* or
51963 *pthread_cond_wait()* does not refer to an initialized mutex object, it is recommended that the
51964 function should fail and report an [EINVAL] error.

51965 Condition Wait Semantics

51966 It is important to note that when *pthread_cond_wait()* and *pthread_cond_timedwait()* return
51967 without error, the associated predicate may still be false. Similarly, when
51968 *pthread_cond_timedwait()* returns with the timeout error, the associated predicate may be true
51969 due to an unavoidable race between the expiration of the timeout and the predicate state change.

51970 The application needs to recheck the predicate on any return because it cannot be sure there is
51971 another thread waiting on the thread to handle the signal, and if there is not then the signal is
51972 lost. The burden is on the application to check the predicate.

51973 Some implementations, particularly on a multi-processor, may sometimes cause multiple
51974 threads to wake up when the condition variable is signaled simultaneously on different
51975 processors.

51976 In general, whenever a condition wait returns, the thread has to re-evaluate the predicate
51977 associated with the condition wait to determine whether it can safely proceed, should wait
51978 again, or should declare a timeout. A return from the wait does not imply that the associated
51979 predicate is either true or false.

51980 It is thus recommended that a condition wait be enclosed in the equivalent of a “while loop”
51981 that checks the predicate.

Timed Wait Semantics

An absolute time measure was chosen for specifying the timeout parameter for two reasons. First, a relative time measure can be easily implemented on top of a function that specifies absolute time, but there is a race condition associated with specifying an absolute timeout on top of a function that specifies relative timeouts. For example, assume that `clock_gettime()` returns the current time and `cond_relative_timed_wait()` uses relative timeouts:

```
clock_gettime(CLOCK_REALTIME, &now)
reltime = sleep_til_this_absolute_time - now;
cond_relative_timed_wait(c, m, &reltime);
```

If the thread is preempted between the first statement and the last statement, the thread blocks for too long. Blocking, however, is irrelevant if an absolute timeout is used. An absolute timeout also need not be recomputed if it is used multiple times in a loop, such as that enclosing a condition wait.

For cases when the system clock is advanced discontinuously by an operator, it is expected that implementations process any timed wait expiring at an intervening time as if that time had actually occurred.

Cancellation and Condition Wait

A condition wait, whether timed or not, is a cancellation point. That is, the functions `pthread_cond_wait()` or `pthread_cond_timedwait()` are points where a pending (or concurrent) cancellation request is noticed. The reason for this is that an indefinite wait is possible at these points—whatever event is being waited for, even if the program is totally correct, might never occur; for example, some input data being awaited might never be sent. By making condition wait a cancellation point, the thread can be canceled and perform its cancellation cleanup handler even though it may be stuck in some indefinite wait.

A side-effect of acting on a cancellation request while a thread is blocked on a condition variable is to re-acquire the mutex before calling any of the cancellation cleanup handlers. This is done in order to ensure that the cancellation cleanup handler is executed in the same state as the critical code that lies both before and after the call to the condition wait function. This rule is also required when interfacing to POSIX threads from languages, such as Ada or C++, which may choose to map cancellation onto a language exception; this rule ensures that each exception handler guarding a critical section can always safely depend upon the fact that the associated mutex has already been locked regardless of exactly where within the critical section the exception was raised. Without this rule, there would not be a uniform rule that exception handlers could follow regarding the lock, and so coding would become very cumbersome.

Therefore, since *some* statement has to be made regarding the state of the lock when a cancellation is delivered during a wait, a definition has been chosen that makes application coding most convenient and error free.

When acting on a cancellation request while a thread is blocked on a condition variable, the implementation is required to ensure that the thread does not consume any condition signals directed at that condition variable if there are any other threads waiting on that condition variable. This rule is specified in order to avoid deadlock conditions that could occur if these two independent requests (one acting on a thread and the other acting on the condition variable) were not processed independently.

Performance of Mutexes and Condition Variables

Mutexes are expected to be locked only for a few instructions. This practice is almost automatically enforced by the desire of programmers to avoid long serial regions of execution (which would reduce total effective parallelism).

When using mutexes and condition variables, one tries to ensure that the usual case is to lock the mutex, access shared data, and unlock the mutex. Waiting on a condition variable should be a relatively rare situation. For example, when implementing a read-write lock, code that acquires a read-lock typically needs only to increment the count of readers (under mutual-exclusion) and return. The calling thread would actually wait on the condition variable only when there is already an active writer. So the efficiency of a synchronization operation is bounded by the cost of mutex lock/unlock and not by condition wait. Note that in the usual case there is no context switch.

This is not to say that the efficiency of condition waiting is unimportant. Since there needs to be at least one context switch per Ada rendezvous, the efficiency of waiting on a condition variable is important. The cost of waiting on a condition variable should be little more than the minimal cost for a context switch plus the time to unlock and lock the mutex.

Features of Mutexes and Condition Variables

It had been suggested that the mutex acquisition and release be decoupled from condition wait. This was rejected because it is the combined nature of the operation that, in fact, facilitates realtime implementations. Those implementations can atomically move a high-priority thread between the condition variable and the mutex in a manner that is transparent to the caller. This can prevent extra context switches and provide more deterministic acquisition of a mutex when the waiting thread is signaled. Thus, fairness and priority issues can be dealt with directly by the scheduling discipline. Furthermore, the current condition wait operation matches existing practice.

Scheduling Behavior of Mutexes and Condition Variables

Synchronization primitives that attempt to interfere with scheduling policy by specifying an ordering rule are considered undesirable. Threads waiting on mutexes and condition variables are selected to proceed in an order dependent upon the scheduling policy rather than in some fixed order (for example, FIFO or priority). Thus, the scheduling policy determines which thread(s) are awakened and allowed to proceed.

Timed Condition Wait

The `pthread_cond_timedwait()` function allows an application to give up waiting for a particular condition after a given amount of time. An example of its use follows:

```
(void) pthread_mutex_lock(&t.mn);
    t.waiters++;
    clock_gettime(CLOCK_REALTIME, &ts);
    ts.tv_sec += 5;
    rc = 0;
    while (! mypredicate(&t) && rc == 0)
        rc = pthread_cond_timedwait(&t.cond, &t.mn, &ts);
    t.waiters--;
    if (rc == 0 || mypredicate(&t))
        setmystate(&t);
(void) pthread_mutex_unlock(&t.mn);
```

By making the timeout parameter absolute, it does not need to be recomputed each time the program checks its blocking predicate. If the timeout was relative, it would have to be recomputed before each call. This would be especially difficult since such code would need to take into account the possibility of extra wakeups that result from extra broadcasts or signals on the condition variable that occur before either the predicate is true or the timeout is due.

FUTURE DIRECTIONS

None.

SEE ALSO

[*pthread_cond_broadcast\(\)*](#)

XBD Section 4.11 (on page 110), [**<pthread.h>**](#)

CHANGE HISTORY

First released in Issue 5. Included for alignment with the POSIX Threads Extension.

Issue 6

The *pthread_cond_timedwait()* and *pthread_cond_wait()* functions are marked as part of the Threads option.

The Open Group Corrigendum U021/9 is applied, correcting the prototype for the *pthread_cond_wait()* function.

The DESCRIPTION is updated for alignment with IEEE Std 1003.1j-2000 by adding semantics for the Clock Selection option.

The ERRORS section has an additional case for [EPERM] in response to IEEE PASC Interpretation 1003.1c #28.

The **restrict** keyword is added to the *pthread_cond_timedwait()* and *pthread_cond_wait()* prototypes for alignment with the ISO/IEC 9899:1999 standard.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/89 is applied, updating the DESCRIPTION for consistency with the *pthread_cond_destroy()* function that states it is safe to destroy an initialized condition variable upon which no threads are currently blocked.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/90 is applied, updating words in the DESCRIPTION from “the cancelability enable state” to “the cancelability type”.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/91 is applied, updating the ERRORS section to remove the error case related to *abstime* from the *pthread_cond_wait()* function, and to make the error case related to *abstime* mandatory for *pthread_cond_timedwait()* for consistency with other functions.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/92 is applied, adding a new paragraph to the RATIONALE section stating that an application should check the predicate on any return from this function.

Issue 7

SD5-XSH-ERN-44 is applied, changing the definition of the “shall fail” case of the [EINVAL] error.

Changes are made from The Open Group Technical Standard, 2006, Extended API Set Part 3.

The *pthread_cond_timedwait()* and *pthread_cond_wait()* functions are moved from the Threads option to the Base.

The [EINVAL] error for an uninitialized condition variable or uninitialized mutex object is removed; this condition results in undefined behavior"

- 52113 The [EPERM] error is revised and moved to the “shall fail” list of error conditions for the
52114 *pthread_cond_timedwait()* function.
- 52115 The DESCRIPTION is updated to clarify the behavior when *mutex* is a robust mutex.
- 52116 The ERRORS section is updated to include “shall fail” cases for
52117 PTHREAD_MUTEX_ERRORCHECK mutexes.
- 52118 The DESCRIPTION is rewritten to clarify that undefined behavior occurs only for mutexes
52119 where the [EPERM] error is not mandated.
- 52120 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0456 [91,286,437] and +
52121 XSH/TC1-2008/0457 [239] are applied.

NAME

pthread_condattr_destroy, pthread_condattr_init — destroy and initialize the condition variable attributes object

SYNOPSIS

```
#include <pthread.h>

int pthread_condattr_destroy(pthread_condattr_t *attr);
int pthread_condattr_init(pthread_condattr_t *attr);
```

DESCRIPTION

The *pthread_condattr_destroy()* function shall destroy a condition variable attributes object; the object becomes, in effect, uninitialized. An implementation may cause *pthread_condattr_destroy()* to set the object referenced by *attr* to an invalid value. A destroyed *attr* attributes object can be reinitialized using *pthread_condattr_init()*; the results of otherwise referencing the object after it has been destroyed are undefined.

The *pthread_condattr_init()* function shall initialize a condition variable attributes object *attr* with the default value for all of the attributes defined by the implementation.

Results are undefined if *pthread_condattr_init()* is called specifying an already initialized *attr* attributes object.

After a condition variable attributes object has been used to initialize one or more condition variables, any function affecting the attributes object (including destruction) shall not affect any previously initialized condition variables.

This volume of POSIX.1-2008 requires two attributes, the *clock* attribute and the *process-shared* attribute.

Additional attributes, their default values, and the names of the associated functions to get and set those attribute values are implementation-defined.

The behavior is undefined if the value specified by the *attr* argument to *pthread_condattr_destroy()* does not refer to an initialized condition variable attributes object.

RETURN VALUE

If successful, the *pthread_condattr_destroy()* and *pthread_condattr_init()* functions shall return zero; otherwise, an error number shall be returned to indicate the error.

ERRORS

The *pthread_condattr_init()* function shall fail if:

[ENOMEM] Insufficient memory exists to initialize the condition variable attributes object.

These functions shall not return an error code of [EINTR].

EXAMPLES

None.

APPLICATION USAGE

None.

RATIONALE

A *process-shared* attribute has been defined for condition variables for the same reason it has been defined for mutexes.

If an implementation detects that the value specified by the *attr* argument to *pthread_condattr_destroy()* does not refer to an initialized condition variable attributes object, it is recommended that the function should fail and report an [EINVAL] error.

52165 See also *pthread_attr_destroy()* and *pthread_mutex_destroy()*.

52166 FUTURE DIRECTIONS

52167 None.

52168 SEE ALSO

52169 *pthread_attr_destroy()*, *pthread_cond_destroy()*, *pthread_condattr_getpshared()*, *pthread_create()*,
52170 *pthread_mutex_destroy()*

52171 XBD <pthread.h>

52172 CHANGE HISTORY

52173 First released in Issue 5. Included for alignment with the POSIX Threads Extension.

52174 Issue 6

52175 The *pthread_condattr_destroy()* and *pthread_condattr_init()* functions are marked as part of the
52176 Threads option.

52177 Issue 7

52178 The *pthread_condattr_destroy()* and *pthread_condattr_init()* functions are moved from the Threads
52179 option to the Base.

52180 The [EINVAL] error for an uninitialized condition variable attributes object is removed; this
52181 condition results in undefined behavior.

52182 **NAME**

52183 pthread_condattr_getclock, pthread_condattr_setclock — get and set the clock selection
 52184 condition variable attribute

52185 **SYNOPSIS**

```
52186 #include <pthread.h>

52187 int pthread_condattr_getclock(const pthread_condattr_t *restrict attr,
52188 clockid_t *restrict clock_id);
52189 int pthread_condattr_setclock(pthread_condattr_t *attr,
52190 clockid_t clock_id);
```

52191 **DESCRIPTION**

52192 The *pthread_condattr_getclock()* function shall obtain the value of the *clock* attribute from the
 52193 attributes object referenced by *attr*.

52194 The *pthread_condattr_setclock()* function shall set the *clock* attribute in an initialized attributes
 52195 object referenced by *attr*. If *pthread_condattr_setclock()* is called with a *clock_id* argument that
 52196 refers to a CPU-time clock, the call shall fail.

52197 The *clock* attribute is the clock ID of the clock that shall be used to measure the timeout service of
 52198 *pthread_cond_timedwait()*. The default value of the *clock* attribute shall refer to the system clock.

52199 The behavior is undefined if the value specified by the *attr* argument to
 52200 *pthread_condattr_getclock()* or *pthread_condattr_setclock()* does not refer to an initialized condition
 52201 variable attributes object.

52202 **RETURN VALUE**

52203 If successful, the *pthread_condattr_getclock()* function shall return zero and store the value of the
 52204 clock attribute of *attr* into the object referenced by the *clock_id* argument. Otherwise, an error
 52205 number shall be returned to indicate the error.

52206 If successful, the *pthread_condattr_setclock()* function shall return zero; otherwise, an error
 52207 number shall be returned to indicate the error.

52208 **ERRORS**

52209 The *pthread_condattr_setclock()* function may fail if:

52210 [EINVAL] The value specified by *clock_id* does not refer to a known clock, or is a CPU-
 52211 time clock.

52212 These functions shall not return an error code of [EINTR].

52213 **EXAMPLES**

52214 None.

52215 **APPLICATION USAGE**

52216 None.

52217 **RATIONALE**

52218 If an implementation detects that the value specified by the *attr* argument to
 52219 *pthread_condattr_getclock()* or *pthread_condattr_setclock()* does not refer to an initialized condition
 52220 variable attributes object, it is recommended that the function should fail and report an
 52221 [EINVAL] error.

52222 **FUTURE DIRECTIONS**

52223 None.

SEE ALSO

pthread_cond_destroy(), *pthread_cond_timedwait()*, *pthread_condattr_destroy()*,
pthread_condattr_getpshared(), *pthread_create()*, *pthread_mutex_destroy()*

XBD <pthread.h>

CHANGE HISTORY

First released in Issue 6. Derived from IEEE Std 1003.1j-2000.

Issue 7

The *pthread_condattr_getclock()* and *pthread_condattr_setclock()* functions are moved from the Clock Selection option to the Base.

The [EINVAL] error for an uninitialized condition variable attributes object is removed; this condition results in undefined behavior.

NAME

pthread_condattr_getpshared, pthread_condattr_setpshared — get and set the process-shared condition variable attributes

SYNOPSIS

```
TSH #include <pthread.h>

int pthread_condattr_getpshared(const pthread_condattr_t *restrict attr,
    int *restrict pshared);
int pthread_condattr_setpshared(pthread_condattr_t *attr,
    int pshared);
```

DESCRIPTION

The *pthread_condattr_getpshared()* function shall obtain the value of the *process-shared* attribute from the attributes object referenced by *attr*.

The *pthread_condattr_setpshared()* function shall set the *process-shared* attribute in an initialized attributes object referenced by *attr*.

The *process-shared* attribute is set to `PTHREAD_PROCESS_SHARED` to permit a condition variable to be operated upon by any thread that has access to the memory where the condition variable is allocated, even if the condition variable is allocated in memory that is shared by multiple processes. If the *process-shared* attribute is `PTHREAD_PROCESS_PRIVATE`, the condition variable shall only be operated upon by threads created within the same process as the thread that initialized the condition variable; if threads of differing processes attempt to operate on such a condition variable, the behavior is undefined. The default value of the attribute is `PTHREAD_PROCESS_PRIVATE`.

The behavior is undefined if the value specified by the *attr* argument to *pthread_condattr_getpshared()* or *pthread_condattr_setpshared()* does not refer to an initialized condition variable attributes object.

RETURN VALUE

If successful, the *pthread_condattr_setpshared()* function shall return zero; otherwise, an error number shall be returned to indicate the error.

If successful, the *pthread_condattr_getpshared()* function shall return zero and store the value of the *process-shared* attribute of *attr* into the object referenced by the *pshared* parameter. Otherwise, an error number shall be returned to indicate the error.

ERRORS

The *pthread_condattr_setpshared()* function may fail if:

[EINVAL] The new value specified for the attribute is outside the range of legal values for that attribute.

These functions shall not return an error code of [EINTR].

EXAMPLES

None.

APPLICATION USAGE

None.

RATIONALE

If an implementation detects that the value specified by the *attr* argument to *pthread_condattr_getpshared()* or *pthread_condattr_setpshared()* does not refer to an initialized condition variable attributes object, it is recommended that the function should fail and report an [EINVAL] error.

FUTURE DIRECTIONS

None.

SEE ALSO

pthread_create(), *pthread_cond_destroy()*, *pthread_condattr_destroy()*, *pthread_mutex_destroy()*

XBD <pthread.h>

CHANGE HISTORY

First released in Issue 5. Included for alignment with the POSIX Threads Extension.

Issue 6

The *pthread_condattr_getpshared()* and *pthread_condattr_setpshared()* functions are marked as part of the Threads and Thread Process-Shared Synchronization options.

The **restrict** keyword is added to the *pthread_condattr_getpshared()* prototype for alignment with the ISO/IEC 9899:1999 standard.

Issue 7

The *pthread_condattr_getpshared()* and *pthread_condattr_setpshared()* functions are moved from the Threads option.

The [EINVAL] error for an uninitialized condition variable attributes object is removed; this condition results in undefined behavior.

52297 **NAME**

52298 pthread_condattr_init — initialize the condition variable attributes object

52299 **SYNOPSIS**

52300 #include <pthread.h>

52301 int pthread_condattr_init(pthread_condattr_t *attr);

52302 **DESCRIPTION**52303 Refer to *pthread_condattr_destroy()*.

52304 NAME

52305 pthread_condattr_setclock — set the clock selection condition variable attribute

52306 SYNOPSIS

52307 #include <pthread.h>

52308 int pthread_condattr_setclock(pthread_condattr_t *attr,
52309 clockid_t clock_id);

52310 DESCRIPTION

52311 Refer to *pthread_condattr_getclock()*.

52312 **NAME**

52313 pthread_condattr_setpshared — set the process-shared condition variable attribute

52314 **SYNOPSIS**

```
52315 TSH      #include <pthread.h>
52316          int pthread_condattr_setpshared(pthread_condattr_t *attr,
52317          int pshared);
```

52318 **DESCRIPTION**52319 Refer to *pthread_condattr_getpshared()*.

52320 **NAME**

52321 pthread_create — thread creation

52322 **SYNOPSIS**

52323 #include <pthread.h>

```
52324 int pthread_create(pthread_t *restrict thread,
52325                   const pthread_attr_t *restrict attr,
52326                   void *(*start_routine)(void*), void *restrict arg);
```

52327 **DESCRIPTION**

52328 The *pthread_create()* function shall create a new thread, with attributes specified by *attr*, within a
 52329 process. If *attr* is NULL, the default attributes shall be used. If the attributes specified by *attr* are
 52330 modified later, the thread's attributes shall not be affected. Upon successful completion,
 52331 *pthread_create()* shall store the ID of the created thread in the location referenced by *thread*.

52332 The thread is created executing *start_routine* with *arg* as its sole argument. If the *start_routine*
 52333 returns, the effect shall be as if there was an implicit call to *pthread_exit()* using the return value
 52334 of *start_routine* as the exit status. Note that the thread in which *main()* was originally invoked
 52335 differs from this. When it returns from *main()*, the effect shall be as if there was an implicit call to
 52336 *exit()* using the return value of *main()* as the exit status.

52337 The signal state of the new thread shall be initialized as follows:

- 52338 • The signal mask shall be inherited from the creating thread.
- 52339 • The set of signals pending for the new thread shall be empty.

52340 XSI The thread-local current locale and the alternate stack shall not be inherited. +

52341 The floating-point environment shall be inherited from the creating thread.

52342 If *pthread_create()* fails, no new thread is created and the contents of the location referenced by
 52343 *thread* are undefined.

52344 TCT If _POSIX_THREAD_CPUTIME is defined, the new thread shall have a CPU-time clock
 52345 accessible, and the initial value of this clock shall be set to zero.

52346 The behavior is undefined if the value specified by the *attr* argument to *pthread_create()* does not
 52347 refer to an initialized thread attributes object.

52348 **RETURN VALUE**

52349 If successful, the *pthread_create()* function shall return zero; otherwise, an error number shall be
 52350 returned to indicate the error.

52351 **ERRORS**

52352 The *pthread_create()* function shall fail if:

52353 [EAGAIN] The system lacked the necessary resources to create another thread, or the
 52354 system-imposed limit on the total number of threads in a process
 52355 {PTHREAD_THREADS_MAX} would be exceeded.

52356 [EPERM] The caller does not have appropriate privileges to set the required scheduling
 52357 parameters or scheduling policy.

52358 The *pthread_create()* function shall not return an error code of [EINTR].

EXAMPLES

None.

APPLICATION USAGE

There is no requirement on the implementation that the ID of the created thread be available before the newly created thread starts executing. The calling thread can obtain the ID of the created thread through the return value of the *pthread_create()* function, and the newly created thread can obtain its ID by a call to *pthread_self()*.

RATIONALE

A suggested alternative to *pthread_create()* would be to define two separate operations: create and start. Some applications would find such behavior more natural. Ada, in particular, separates the “creation” of a task from its “activation”.

Splitting the operation was rejected by the standard developers for many reasons:

- The number of calls required to start a thread would increase from one to two and thus place an additional burden on applications that do not require the additional synchronization. The second call, however, could be avoided by the additional complication of a start-up state attribute.
- An extra state would be introduced: “created but not started”. This would require the standard to specify the behavior of the thread operations when the target has not yet started executing.
- For those applications that require such behavior, it is possible to simulate the two separate steps with the facilities that are currently provided. The *start_routine()* can synchronize by waiting on a condition variable that is signaled by the start operation.

An Ada implementor can choose to create the thread at either of two points in the Ada program: when the task object is created, or when the task is activated (generally at a “begin”). If the first approach is adopted, the *start_routine()* needs to wait on a condition variable to receive the order to begin “activation”. The second approach requires no such condition variable or extra synchronization. In either approach, a separate Ada task control block would need to be created when the task object is created to hold rendezvous queues, and so on.

An extension of the preceding model would be to allow the state of the thread to be modified between the create and start. This would allow the thread attributes object to be eliminated. This has been rejected because:

- All state in the thread attributes object has to be able to be set for the thread. This would require the definition of functions to modify thread attributes. There would be no reduction in the number of function calls required to set up the thread. In fact, for an application that creates all threads using identical attributes, the number of function calls required to set up the threads would be dramatically increased. Use of a thread attributes object permits the application to make one set of attribute setting function calls. Otherwise, the set of attribute setting function calls needs to be made for each thread creation.
- Depending on the implementation architecture, functions to set thread state would require kernel calls, or for other implementation reasons would not be able to be implemented as macros, thereby increasing the cost of thread creation.
- The ability for applications to segregate threads by class would be lost.

Another suggested alternative uses a model similar to that for process creation, such as “thread fork”. The fork semantics would provide more flexibility and the “create” function can be implemented simply by doing a thread fork followed immediately by a call to the desired “start

routine” for the thread. This alternative has these problems:

- For many implementations, the entire stack of the calling thread would need to be duplicated, since in many architectures there is no way to determine the size of the calling frame.
- Efficiency is reduced since at least some part of the stack has to be copied, even though in most cases the thread never needs the copied context, since it merely calls the desired start routine.

If an implementation detects that the value specified by the *attr* argument to *pthread_create()* does not refer to an initialized thread attributes object, it is recommended that the function should fail and report an [EINVAL] error.

FUTURE DIRECTIONS

None.

SEE ALSO

fork(), *pthread_exit()*, *pthread_join()*

XBD Section 4.11 (on page 110), **<pthread.h>**

CHANGE HISTORY

First released in Issue 5. Included for alignment with the POSIX Threads Extension.

Issue 6

The *pthread_create()* function is marked as part of the Threads option.

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The [EPERM] mandatory error condition is added.

The thread CPU-time clock semantics are added for alignment with IEEE Std 1003.1d-1999.

The **restrict** keyword is added to the *pthread_create()* prototype for alignment with the ISO/IEC 9899:1999 standard.

The DESCRIPTION is updated to make it explicit that the floating-point environment is inherited from the creating thread.

IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/44 is applied, adding text that the alternate stack is not inherited.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/93 is applied, updating the ERRORS section to remove the mandatory [EINVAL] error (“The value specified by *attr* is invalid”), and adding the optional [EINVAL] error (“The attributes specified by *attr* are invalid”).

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/94 is applied, adding the APPLICATION USAGE section.

Issue 7

The *pthread_create()* function is moved from the Threads option to the Base.

The [EINVAL] error for an uninitialized thread attributes object is removed; this condition results in undefined behavior.

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0458 [302] is applied.

+

52444 **NAME**

52445 pthread_detach — detach a thread

52446 **SYNOPSIS**

52447 #include <pthread.h>

52448 int pthread_detach(pthread_t thread);

52449 **DESCRIPTION**

52450 The *pthread_detach()* function shall indicate to the implementation that storage for the thread
52451 *thread* can be reclaimed when that thread terminates. If *thread* has not terminated,
52452 *pthread_detach()* shall not cause it to terminate.

52453 The behavior is undefined if the value specified by the *thread* argument to *pthread_detach()* does
52454 not refer to a joinable thread.

52455 **RETURN VALUE**

52456 If the call succeeds, *pthread_detach()* shall return 0; otherwise, an error number shall be returned
52457 to indicate the error.

52458 **ERRORS**52459 The *pthread_detach()* function shall not return an error code of [EINTR].52460 **EXAMPLES**

52461 None.

52462 **APPLICATION USAGE**

52463 None.

52464 **RATIONALE**

52465 The *pthread_join()* or *pthread_detach()* functions should eventually be called for every thread that
52466 is created so that storage associated with the thread may be reclaimed.

52467 It has been suggested that a “detach” function is not necessary; the *detachstate* thread creation
52468 attribute is sufficient, since a thread need never be dynamically detached. However, need arises
52469 in at least two cases:

- 52470 1. In a cancellation handler for a *pthread_join()* it is nearly essential to have a
52471 *pthread_detach()* function in order to detach the thread on which *pthread_join()* was
52472 waiting. Without it, it would be necessary to have the handler do another *pthread_join()* to
52473 attempt to detach the thread, which would both delay the cancellation processing for an
52474 unbounded period and introduce a new call to *pthread_join()*, which might itself need a
52475 cancellation handler. A dynamic detach is nearly essential in this case.
- 52476 2. In order to detach the “initial thread” (as may be desirable in processes that set up server
52477 threads).

52478 If an implementation detects that the value specified by the *thread* argument to *pthread_detach()*
52479 does not refer to a joinable thread, it is recommended that the function should fail and report an
52480 [EINVAL] error.

52481 If an implementation detects use of a thread ID after the end of its lifetime, it is recommended
52482 that the function should fail and report an [ESRCH] error.

52483 **FUTURE DIRECTIONS**

52484 None.

52485 **SEE ALSO**52486 *pthread_join()*

52487 XBD <pthread.h>

52488 **CHANGE HISTORY**

52489 First released in Issue 5. Included for alignment with the POSIX Threads Extension.

52490 **Issue 6**52491 The *pthread_detach()* function is marked as part of the Threads option.52492 IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/95 is applied, updating the ERRORS
52493 section so that the [EINVAL] and [ESRCH] error cases become optional.52494 **Issue 7**52495 The *pthread_detach()* function is moved from the Threads option to the Base.

52496 Austin Group Interpretation 1003.1-2001 #142 is applied, removing the [ESRCH] error condition.

52497 The [EINVAL] error for a non-joinable thread is removed; this condition results in undefined
52498 behavior.

52499 **NAME**

52500 pthread_equal — compare thread IDs

52501 **SYNOPSIS**

52502 #include <pthread.h>

52503 int pthread_equal(pthread_t t1, pthread_t t2);

52504 **DESCRIPTION**52505 This function shall compare the thread IDs *t1* and *t2*.52506 **RETURN VALUE**52507 The *pthread_equal()* function shall return a non-zero value if *t1* and *t2* are equal; otherwise, zero
52508 shall be returned.52509 If either *t1* or *t2* are not valid thread IDs, the behavior is undefined.52510 **ERRORS**

52511 No errors are defined.

52512 The *pthread_equal()* function shall not return an error code of [EINTR].52513 **EXAMPLES**

52514 None.

52515 **APPLICATION USAGE**

52516 None.

52517 **RATIONALE**52518 Implementations may choose to define a thread ID as a structure. This allows additional
52519 flexibility and robustness over using an **int**. For example, a thread ID could include a sequence
52520 number that allows detection of “dangling IDs” (copies of a thread ID that has been detached).
52521 Since the C language does not support comparison on structure types, the *pthread_equal()*
52522 function is provided to compare thread IDs.52523 **FUTURE DIRECTIONS**

52524 None.

52525 **SEE ALSO**52526 *pthread_create()*, *pthread_self()*

52527 XBD <pthread.h>

52528 **CHANGE HISTORY**

52529 First released in Issue 5. Included for alignment with the POSIX Threads Extension.

52530 **Issue 6**52531 The *pthread_equal()* function is marked as part of the Threads option.52532 **Issue 7**52533 The *pthread_equal()* function is moved from the Threads option to the Base.

NAME

pthread_exit — thread termination

SYNOPSIS

```
#include <pthread.h>
```

```
void pthread_exit(void *value_ptr);
```

DESCRIPTION

The *pthread_exit()* function shall terminate the calling thread and make the value *value_ptr* available to any successful join with the terminating thread. Any cancellation cleanup handlers that have been pushed and not yet popped shall be popped in the reverse order that they were pushed and then executed. After all cancellation cleanup handlers have been executed, if the thread has any thread-specific data, appropriate destructor functions shall be called in an unspecified order. Thread termination does not release any application visible process resources, including, but not limited to, mutexes and file descriptors, nor does it perform any process-level cleanup actions, including, but not limited to, calling any *atexit()* routines that may exist.

An implicit call to *pthread_exit()* is made when a thread other than the thread in which *main()* was first invoked returns from the start routine that was used to create it. The function's return value shall serve as the thread's exit status.

The behavior of *pthread_exit()* is undefined if called from a cancellation cleanup handler or destructor function that was invoked as a result of either an implicit or explicit call to *pthread_exit()*.

After a thread has terminated, the result of access to local (auto) variables of the thread is undefined. Thus, references to local variables of the exiting thread should not be used for the *pthread_exit()* *value_ptr* parameter value.

The process shall exit with an exit status of 0 after the last thread has been terminated. The behavior shall be as if the implementation called *exit()* with a zero argument at thread termination time.

RETURN VALUE

The *pthread_exit()* function cannot return to its caller.

ERRORS

No errors are defined.

EXAMPLES

None.

APPLICATION USAGE

None.

RATIONALE

The normal mechanism by which a thread terminates is to return from the routine that was specified in the *pthread_create()* call that started it. The *pthread_exit()* function provides the capability for a thread to terminate without requiring a return from the start routine of that thread, thereby providing a function analogous to *exit()*.

Regardless of the method of thread termination, any cancellation cleanup handlers that have been pushed and not yet popped are executed, and the destructors for any existing thread-specific data are executed. This volume of POSIX.1-2008 requires that cancellation cleanup handlers be popped and called in order. After all cancellation cleanup handlers have been executed, thread-specific data destructors are called, in an unspecified order, for each item of thread-specific data that exists in the thread. This ordering is necessary because cancellation cleanup handlers may rely on thread-specific data.

52580 As the meaning of the status is determined by the application (except when the thread has been
52581 canceled, in which case it is PTHREAD_CANCELED), the implementation has no idea what an
52582 illegal status value is, which is why no address error checking is done.

52583 **FUTURE DIRECTIONS**

52584 None.

52585 **SEE ALSO**

52586 *exit()*, *pthread_create()*, *pthread_join()*

52587 XBD <pthread.h>

52588 **CHANGE HISTORY**

52589 First released in Issue 5. Included for alignment with the POSIX Threads Extension.

52590 **Issue 6**

52591 The *pthread_exit()* function is marked as part of the Threads option.

52592 **Issue 7**

52593 The *pthread_exit()* function is moved from the Threads option to the Base.

NAME

pthread_getconcurrency, pthread_setconcurrency — get and set the level of concurrency

SYNOPSIS

```
OB XSI #include <pthread.h>

int pthread_getconcurrency(void);
int pthread_setconcurrency(int new_level);
```

DESCRIPTION

Unbound threads in a process may or may not be required to be simultaneously active. By default, the threads implementation ensures that a sufficient number of threads are active so that the process can continue to make progress. While this conserves system resources, it may not produce the most effective level of concurrency.

The *pthread_setconcurrency()* function allows an application to inform the threads implementation of its desired concurrency level, *new_level*. The actual level of concurrency provided by the implementation as a result of this function call is unspecified.

If *new_level* is zero, it causes the implementation to maintain the concurrency level at its discretion as if *pthread_setconcurrency()* had never been called.

The *pthread_getconcurrency()* function shall return the value set by a previous call to the *pthread_setconcurrency()* function. If the *pthread_setconcurrency()* function was not previously called, this function shall return zero to indicate that the implementation is maintaining the concurrency level.

A call to *pthread_setconcurrency()* shall inform the implementation of its desired concurrency level. The implementation shall use this as a hint, not a requirement.

If an implementation does not support multiplexing of user threads on top of several kernel-scheduled entities, the *pthread_setconcurrency()* and *pthread_getconcurrency()* functions are provided for source code compatibility but they shall have no effect when called. To maintain the function semantics, the *new_level* parameter is saved when *pthread_setconcurrency()* is called so that a subsequent call to *pthread_getconcurrency()* shall return the same value.

RETURN VALUE

If successful, the *pthread_setconcurrency()* function shall return zero; otherwise, an error number shall be returned to indicate the error.

The *pthread_getconcurrency()* function shall always return the concurrency level set by a previous call to *pthread_setconcurrency()*. If the *pthread_setconcurrency()* function has never been called, *pthread_getconcurrency()* shall return zero.

ERRORS

The *pthread_setconcurrency()* function shall fail if:

[EINVAL] The value specified by *new_level* is negative.

[EAGAIN] The value specified by *new_level* would cause a system resource to be exceeded.

The *pthread_setconcurrency()* function shall not return an error code of [EINTR].

52633 EXAMPLES

52634 None.

52635 APPLICATION USAGE

52636 Application developers should note that an implementation can always ignore any calls to
52637 *pthread_setconcurrency()* and return a constant for *pthread_getconcurrency()*. For this reason, it is
52638 not recommended that portable applications use this function.

52639 RATIONALE

52640 None.

52641 FUTURE DIRECTIONS

52642 These functions may be removed in a future version.

52643 SEE ALSO

52644 XBD <pthread.h>

52645 CHANGE HISTORY

52646 First released in Issue 5.

52647 Issue 7

52648 SD5-XSH-ERN-184 is applied.

52649 The *pthread_getconcurrency()* and *pthread_setconcurrency()* functions are marked obsolescent.

NAME

pthread_getcpuclockid — access a thread CPU-time clock (**ADVANCED REALTIME THREADS**)

SYNOPSIS

```
TCT    #include <pthread.h>
        #include <time.h>

        int pthread_getcpuclockid(pthread_t thread_id, clockid_t *clock_id);
```

DESCRIPTION

The *pthread_getcpuclockid()* function shall return in *clock_id* the clock ID of the CPU-time clock of the thread specified by *thread_id*, if the thread specified by *thread_id* exists.

RETURN VALUE

Upon successful completion, *pthread_getcpuclockid()* shall return zero; otherwise, an error number shall be returned to indicate the error.

ERRORS

No errors are defined.

EXAMPLES

None.

APPLICATION USAGE

The *pthread_getcpuclockid()* function is part of the Thread CPU-Time Clocks option and need not be provided on all implementations.

RATIONALE

If an implementation detects use of a thread ID after the end of its lifetime, it is recommended that the function should fail and report an [ESRCH] error.

FUTURE DIRECTIONS

None.

SEE ALSO

clock_getcpuclockid(), *clock_getres()*, *timer_create()*

XBD **<pthread.h>**, **<time.h>**

CHANGE HISTORY

First released in Issue 6. Derived from IEEE Std 1003.1d-1999.

In the SYNOPSIS, the inclusion of **<sys/types.h>** is no longer required.

Issue 7

The *pthread_getcpuclockid()* function is moved from the Threads option.

Austin Group Interpretation 1003.1-2001 #142 is applied, removing the [ESRCH] error condition.

NAME

pthread_getschedparam, pthread_setschedparam — dynamic thread scheduling parameters access (**REALTIME THREADS**)

SYNOPSIS

```

#include <pthread.h>

int pthread_getschedparam(pthread_t thread, int *restrict policy,
    struct sched_param *restrict param);
int pthread_setschedparam(pthread_t thread, int policy,
    const struct sched_param *param);

```

DESCRIPTION

The *pthread_getschedparam()* and *pthread_setschedparam()* functions shall, respectively, get and set the scheduling policy and parameters of individual threads within a multi-threaded process to be retrieved and set. For SCHED_FIFO and SCHED_RR, the only required member of the **sched_param** structure is the priority *sched_priority*. For SCHED_OTHER, the affected scheduling parameters are implementation-defined.

The *pthread_getschedparam()* function shall retrieve the scheduling policy and scheduling parameters for the thread whose thread ID is given by *thread* and shall store those values in *policy* and *param*, respectively. The priority value returned from *pthread_getschedparam()* shall be the value specified by the most recent *pthread_setschedparam()*, *pthread_setschedprio()*, or *pthread_create()* call affecting the target thread. It shall not reflect any temporary adjustments to its priority as a result of any priority inheritance or ceiling functions. The *pthread_setschedparam()* function shall set the scheduling policy and associated scheduling parameters for the thread whose thread ID is given by *thread* to the policy and associated parameters provided in *policy* and *param*, respectively.

The *policy* parameter may have the value SCHED_OTHER, SCHED_FIFO, or SCHED_RR. The scheduling parameters for the SCHED_OTHER policy are implementation-defined. The SCHED_FIFO and SCHED_RR policies shall have a single scheduling parameter, *priority*.

If **_POSIX_THREAD_SPORADIC_SERVER** is defined, then the *policy* argument may have the value SCHED_SPORADIC, with the exception for the *pthread_setschedparam()* function that if the scheduling policy was not SCHED_SPORADIC at the time of the call, it is implementation-defined whether the function is supported; in other words, the implementation need not allow the application to dynamically change the scheduling policy to SCHED_SPORADIC. The sporadic server scheduling policy has the associated parameters *sched_ss_low_priority*, *sched_ss_repl_period*, *sched_ss_init_budget*, *sched_priority*, and *sched_ss_max_repl*. The specified *sched_ss_repl_period* shall be greater than or equal to the specified *sched_ss_init_budget* for the function to succeed; if it is not, then the function shall fail. The value of *sched_ss_max_repl* shall be within the inclusive range [1,{SS_REPL_MAX}] for the function to succeed; if not, the function shall fail. It is unspecified whether the *sched_ss_repl_period* and *sched_ss_init_budget* values are stored as provided by this function or are rounded to align with the resolution of the clock being used.

If the *pthread_setschedparam()* function fails, the scheduling parameters shall not be changed for the target thread.

RETURN VALUE

If successful, the *pthread_getschedparam()* and *pthread_setschedparam()* functions shall return zero; otherwise, an error number shall be returned to indicate the error.

ERRORS

The *pthread_setschedparam()* function shall fail if:

[ENOTSUP] An attempt was made to set the policy or scheduling parameters to an unsupported value.

[ENOTSUP] An attempt was made to dynamically change the scheduling policy to SCHED_SPORADIC, and the implementation does not support this change.

The *pthread_setschedparam()* function may fail if:

[EINVAL] The value specified by *policy* or one of the scheduling parameters associated with the scheduling policy *policy* is invalid.

[EPERM] The caller does not have appropriate privileges to set either the scheduling parameters or the scheduling policy of the specified thread.

[EPERM] The implementation does not allow the application to modify one of the parameters to the value specified.

These functions shall not return an error code of [EINTR].

EXAMPLES

None.

APPLICATION USAGE

None.

RATIONALE

If an implementation detects use of a thread ID after the end of its lifetime, it is recommended that the function should fail and report an [ESRCH] error.

FUTURE DIRECTIONS

None.

SEE ALSO

pthread_setschedprio(), *sched_getparam()*, *sched_getscheduler()*

XBD <pthread.h>, <sched.h>

CHANGE HISTORY

First released in Issue 5. Included for alignment with the POSIX Threads Extension.

Issue 6

The *pthread_getschedparam()* and *pthread_setschedparam()* functions are marked as part of the Threads and Thread Execution Scheduling options.

The [ENOSYS] error condition has been removed as stubs need not be provided if an implementation does not support the Thread Execution Scheduling option.

The Open Group Corrigendum U026/2 is applied, correcting the prototype for the *pthread_setschedparam()* function so that its second argument is of type **int**.

The SCHED_SPORADIC scheduling policy is added for alignment with IEEE Std 1003.1d-1999.

The **restrict** keyword is added to the *pthread_getschedparam()* prototype for alignment with the ISO/IEC 9899:1999 standard.

The Open Group Corrigendum U047/1 is applied.

IEEE PASC Interpretation 1003.1 #96 is applied, noting that priority values can also be set by a call to the *pthread_setschedprio()* function.

52770 **Issue 7**

52771 The *pthread_getschedparam()* and *pthread_setschedparam()* functions are moved from the Threads
52772 option.

52773 Austin Group Interpretation 1003.1-2001 #119 is applied, clarifying the accuracy requirements
52774 for the *sched_ss_repl_period* and *sched_ss_init_budget* values.

52775 Austin Group Interpretation 1003.1-2001 #142 is applied, removing the [ESRCH] error condition.

52776 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0459 [314] is applied.

+

NAME

pthread_getspecific, pthread_setspecific — thread-specific data management

SYNOPSIS

```
#include <pthread.h>
```

```
void *pthread_getspecific(pthread_key_t key);
int pthread_setspecific(pthread_key_t key, const void *value);
```

DESCRIPTION

The *pthread_getspecific()* function shall return the value currently bound to the specified *key* on behalf of the calling thread.

The *pthread_setspecific()* function shall associate a thread-specific *value* with a *key* obtained via a previous call to *pthread_key_create()*. Different threads may bind different values to the same key. These values are typically pointers to blocks of dynamically allocated memory that have been reserved for use by the calling thread.

The effect of calling *pthread_getspecific()* or *pthread_setspecific()* with a *key* value not obtained from *pthread_key_create()* or after *key* has been deleted with *pthread_key_delete()* is undefined.

Both *pthread_getspecific()* and *pthread_setspecific()* may be called from a thread-specific data destructor function. A call to *pthread_getspecific()* for the thread-specific data key being destroyed shall return the value NULL, unless the value is changed (after the destructor starts) by a call to *pthread_setspecific()*. Calling *pthread_setspecific()* from a thread-specific data destructor routine may result either in lost storage (after at least PTHREAD_DESTRUCTOR_ITERATIONS attempts at destruction) or in an infinite loop.

Both functions may be implemented as macros.

RETURN VALUE

The *pthread_getspecific()* function shall return the thread-specific data value associated with the given *key*. If no thread-specific data value is associated with *key*, then the value NULL shall be returned.

If successful, the *pthread_setspecific()* function shall return zero; otherwise, an error number shall be returned to indicate the error.

ERRORS

No errors are returned from *pthread_getspecific()*.

The *pthread_setspecific()* function shall fail if:

[ENOMEM] Insufficient memory exists to associate the non-NULL value with the key.

The *pthread_setspecific()* function shall not return an error code of [EINTR].

EXAMPLES

None.

APPLICATION USAGE

None.

RATIONALE

Performance and ease-of-use of *pthread_getspecific()* are critical for functions that rely on maintaining state in thread-specific data. Since no errors are required to be detected by it, and since the only error that could be detected is the use of an invalid key, the function to *pthread_getspecific()* has been designed to favor speed and simplicity over error reporting.

If an implementation detects that the value specified by the *key* argument to *pthread_setspecific()* does not refer to a key value obtained from *pthread_key_create()* or refers to a key that has been

52821 deleted with *pthread_key_delete()*, it is recommended that the function should fail and report an
52822 [EINVAL] error.

52823 FUTURE DIRECTIONS

52824 None.

52825 SEE ALSO

52826 *pthread_key_create()*

52827 XBD <pthread.h>

52828 CHANGE HISTORY

52829 First released in Issue 5. Included for alignment with the POSIX Threads Extension.

52830 Issue 6

52831 The *pthread_getspecific()* and *pthread_setspecific()* functions are marked as part of the Threads
52832 option.

52833 IEEE PASC Interpretation 1003.1c #3 (Part 6) is applied, updating the DESCRIPTION.

52834 IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/96 is applied, updating the ERRORS
52835 section so that the [ENOMEM] error case is changed from “to associate the value with the key”
52836 to “to associate the non-NULL value with the key”.

52837 Issue 7

52838 Austin Group Interpretation 1003.1-2001 #063 is applied, updating the ERRORS section.

52839 The *pthread_getspecific()* and *pthread_setspecific()* functions are moved from the Threads option to
52840 the Base.

52841 The [EINVAL] error for a key value not obtained from *pthread_key_create()* or a key deleted with
52842 *pthread_key_delete()* is removed; this condition results in undefined behavior.

52843 **NAME**

52844 pthread_join — wait for thread termination

52845 **SYNOPSIS**

52846 #include <pthread.h>

52847 int pthread_join(pthread_t thread, void **value_ptr);

52848 **DESCRIPTION**

52849 The *pthread_join()* function shall suspend execution of the calling thread until the target *thread*
 52850 terminates, unless the target *thread* has already terminated. On return from a successful
 52851 *pthread_join()* call with a non-NULL *value_ptr* argument, the value passed to *pthread_exit()* by
 52852 the terminating thread shall be made available in the location referenced by *value_ptr*. When a
 52853 *pthread_join()* returns successfully, the target thread has been terminated. The results of multiple
 52854 simultaneous calls to *pthread_join()* specifying the same target thread are undefined. If the
 52855 thread calling *pthread_join()* is canceled, then the target thread shall not be detached.

52856 It is unspecified whether a thread that has exited but remains unjoined counts against
 52857 {PTHREAD_THREADS_MAX}.

52858 The behavior is undefined if the value specified by the *thread* argument to *pthread_join()* does not
 52859 refer to a joinable thread.

52860 The behavior is undefined if the value specified by the *thread* argument to *pthread_join()* refers to
 52861 the calling thread.

52862 **RETURN VALUE**

52863 If successful, the *pthread_join()* function shall return zero; otherwise, an error number shall be
 52864 returned to indicate the error.

52865 **ERRORS**52866 The *pthread_join()* function may fail if:

52867 [EDEADLK] A deadlock was detected.

52868 The *pthread_join()* function shall not return an error code of [EINTR].52869 **EXAMPLES**

52870 An example of thread creation and deletion follows:

```

52871 typedef struct {
52872     int *ar;
52873     long n;
52874 } subarray;

52875 void *
52876 incer(void *arg)
52877 {
52878     long i;

52879     for (i = 0; i < ((subarray *)arg)->n; i++)
52880         ((subarray *)arg)->ar[i]++;
52881 }

52882 int main(void)
52883 {
52884     int          ar[1000000];
52885     pthread_t    th1, th2;
52886     subarray     sb1, sb2;

```

```

52887     sb1.ar = &ar[0];
52888     sb1.n  = 500000;
52889     (void) pthread_create(&th1, NULL, incer, &sb1);

52890     sb2.ar = &ar[500000];
52891     sb2.n  = 500000;
52892     (void) pthread_create(&th2, NULL, incer, &sb2);

52893     (void) pthread_join(th1, NULL);
52894     (void) pthread_join(th2, NULL);
52895     return 0;
52896 }

```

APPLICATION USAGE

None.

RATIONALE

The *pthread_join()* function is a convenience that has proven useful in multi-threaded applications. It is true that a programmer could simulate this function if it were not provided by passing extra state as part of the argument to the *start_routine()*. The terminating thread would set a flag to indicate termination and broadcast a condition that is part of that state; a joining thread would wait on that condition variable. While such a technique would allow a thread to wait on more complex conditions (for example, waiting for multiple threads to terminate), waiting on individual thread termination is considered widely useful. Also, including the *pthread_join()* function in no way precludes a programmer from coding such complex waits. Thus, while not a primitive, including *pthread_join()* in this volume of POSIX.1-2008 was considered valuable.

The *pthread_join()* function provides a simple mechanism allowing an application to wait for a thread to terminate. After the thread terminates, the application may then choose to clean up resources that were used by the thread. For instance, after *pthread_join()* returns, any application-provided stack storage could be reclaimed.

The *pthread_join()* or *pthread_detach()* function should eventually be called for every thread that is created with the *detachstate* attribute set to *PTHREAD_CREATE_JOINABLE* so that storage associated with the thread may be reclaimed.

The interaction between *pthread_join()* and cancellation is well-defined for the following reasons:

- The *pthread_join()* function, like all other non-async-cancel-safe functions, can only be called with deferred cancelability type.
- Cancellation cannot occur in the disabled cancelability state.

Thus, only the default cancelability state need be considered. As specified, either the *pthread_join()* call is canceled, or it succeeds, but not both. The difference is obvious to the application, since either a cancellation handler is run or *pthread_join()* returns. There are no race conditions since *pthread_join()* was called in the deferred cancelability state.

If an implementation detects that the value specified by the *thread* argument to *pthread_join()* does not refer to a joinable thread, it is recommended that the function should fail and report an [EINVAL] error.

If an implementation detects that the value specified by the *thread* argument to *pthread_join()* refers to the calling thread, it is recommended that the function should fail and report an [EDEADLK] error.

If an implementation detects use of a thread ID after the end of its lifetime, it is recommended

52932 that the function should fail and report an [ESRCH] error.

52933 **FUTURE DIRECTIONS**

52934 None.

52935 **SEE ALSO**

52936 *pthread_create()*, *wait()*

52937 XBD Section 4.11 (on page 110), *<pthread.h>*

52938 **CHANGE HISTORY**

52939 First released in Issue 5. Included for alignment with the POSIX Threads Extension.

52940 **Issue 6**

52941 The *pthread_join()* function is marked as part of the Threads option.

52942 IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/97 is applied, updating the ERRORS
52943 section so that the [EINVAL] error is made optional and the words “the implementation has
52944 detected” are removed from it.

52945 **Issue 7**

52946 The *pthread_join()* function is moved from the Threads option to the Base.

52947 Austin Group Interpretation 1003.1-2001 #142 is applied, removing the [ESRCH] error condition.

52948 The [EINVAL] error for a non-joinable thread is removed; this condition results in undefined
52949 behavior.

52950 The [EDEADLK] error for the calling thread is removed; this condition results in undefined
52951 behavior.

NAME

pthread_key_create — thread-specific data key creation

SYNOPSIS

```
#include <pthread.h>
```

```
int pthread_key_create(pthread_key_t *key, void (*destructor)(void*));
```

DESCRIPTION

The *pthread_key_create()* function shall create a thread-specific data key visible to all threads in the process. Key values provided by *pthread_key_create()* are opaque objects used to locate thread-specific data. Although the same key value may be used by different threads, the values bound to the key by *pthread_setspecific()* are maintained on a per-thread basis and persist for the life of the calling thread.

Upon key creation, the value NULL shall be associated with the new key in all active threads. Upon thread creation, the value NULL shall be associated with all defined keys in the new thread.

An optional destructor function may be associated with each key value. At thread exit, if a key value has a non-NULL destructor pointer, and the thread has a non-NULL value associated with that key, the value of the key is set to NULL, and then the function pointed to is called with the previously associated value as its sole argument. The order of destructor calls is unspecified if more than one destructor exists for a thread when it exits.

If, after all the destructors have been called for all non-NULL values with associated destructors, there are still some non-NULL values with associated destructors, then the process is repeated. If, after at least {PTHREAD_DESTRUCTOR_ITERATIONS} iterations of destructor calls for outstanding non-NULL values, there are still some non-NULL values with associated destructors, implementations may stop calling destructors, or they may continue calling destructors until no non-NULL values with associated destructors exist, even though this might result in an infinite loop.

RETURN VALUE

If successful, the *pthread_key_create()* function shall store the newly created key value at **key* and shall return zero. Otherwise, an error number shall be returned to indicate the error.

ERRORS

The *pthread_key_create()* function shall fail if:

[EAGAIN] The system lacked the necessary resources to create another thread-specific data key, or the system-imposed limit on the total number of keys per process {PTHREAD_KEYS_MAX} has been exceeded.

[ENOMEM] Insufficient memory exists to create the key.

The *pthread_key_create()* function shall not return an error code of [EINTR].

EXAMPLES

The following example demonstrates a function that initializes a thread-specific data key when it is first called, and associates a thread-specific object with each calling thread, initializing this object when necessary.

```
static pthread_key_t key;
static pthread_once_t key_once = PTHREAD_ONCE_INIT;

static void
make_key()
{
```

```

52997         (void) pthread_key_create(&key, NULL);
52998     }
52999     func()
53000     {
53001         void *ptr;
53002         (void) pthread_once(&key_once, make_key);
53003         if ((ptr = pthread_getspecific(key)) == NULL) {
53004             ptr = malloc(OBJECT_SIZE);
53005             ...
53006             (void) pthread_setspecific(key, ptr);
53007         }
53008         ...
53009     }

```

Note that the key has to be initialized before *pthread_getspecific()* or *pthread_setspecific()* can be used. The *pthread_key_create()* call could either be explicitly made in a module initialization routine, or it can be done implicitly by the first call to a module as in this example. Any attempt to use the key before it is initialized is a programming error, making the code below incorrect.

```

53014     static pthread_key_t key;
53015     func()
53016     {
53017         void *ptr;
53018         /* KEY NOT INITIALIZED!!! THIS WON'T WORK!!! */
53019         if ((ptr = pthread_getspecific(key)) == NULL &&
53020             pthread_setspecific(key, NULL) != 0) {
53021             pthread_key_create(&key, NULL);
53022             ...
53023         }
53024     }

```

APPLICATION USAGE

None.

RATIONALE

Destructor Functions

Normally, the value bound to a key on behalf of a particular thread is a pointer to storage allocated dynamically on behalf of the calling thread. The destructor functions specified with *pthread_key_create()* are intended to be used to free this storage when the thread exits. Thread cancellation cleanup handlers cannot be used for this purpose because thread-specific data may persist outside the lexical scope in which the cancellation cleanup handlers operate.

If the value associated with a key needs to be updated during the lifetime of the thread, it may be necessary to release the storage associated with the old value before the new value is bound. Although the *pthread_setspecific()* function could do this automatically, this feature is not needed often enough to justify the added complexity. Instead, the programmer is responsible for freeing the stale storage:

```

53039     pthread_getspecific(key, &old);
53040     new = allocate();
53041     destructor(old);

```

53042 pthread_setspecific(key, new);

53043 **Note:** The above example could leak storage if run with asynchronous cancellation enabled. No such
 53044 problems occur in the default cancellation state if no cancellation points occur between the get
 53045 and set.

53046 There is no notion of a destructor-safe function. If an application does not call *pthread_exit()*
 53047 from a signal handler, or if it blocks any signal whose handler may call *pthread_exit()* while
 53048 calling async-unsafe functions, all functions may be safely called from destructors.

53049 Non-Idempotent Data Key Creation

53050 There were requests to make *pthread_key_create()* idempotent with respect to a given *key* address
 53051 parameter. This would allow applications to call *pthread_key_create()* multiple times for a given
 53052 *key* address and be guaranteed that only one key would be created. Doing so would require the
 53053 key value to be previously initialized (possibly at compile time) to a known null value and
 53054 would require that implicit mutual-exclusion be performed based on the address and contents of
 53055 the *key* parameter in order to guarantee that exactly one key would be created.

53056 Unfortunately, the implicit mutual-exclusion would not be limited to only *pthread_key_create()*.
 53057 On many implementations, implicit mutual-exclusion would also have to be performed by
 53058 *pthread_getspecific()* and *pthread_setspecific()* in order to guard against using incompletely stored
 53059 or not-yet-visible key values. This could significantly increase the cost of important operations,
 53060 particularly *pthread_getspecific()*.

53061 Thus, this proposal was rejected. The *pthread_key_create()* function performs no implicit
 53062 synchronization. It is the responsibility of the programmer to ensure that it is called exactly once
 53063 per key before use of the key. Several straightforward mechanisms can already be used to
 53064 accomplish this, including calling explicit module initialization functions, using mutexes, and
 53065 using *pthread_once()*. This places no significant burden on the programmer, introduces no
 53066 possibly confusing *ad hoc* implicit synchronization mechanism, and potentially allows
 53067 commonly used thread-specific data operations to be more efficient.

53068 FUTURE DIRECTIONS

53069 None.

53070 SEE ALSO

53071 *pthread_getspecific()*, *pthread_key_delete()*

53072 XBD <pthread.h>

53073 CHANGE HISTORY

53074 First released in Issue 5. Included for alignment with the POSIX Threads Extension.

53075 Issue 6

53076 The *pthread_key_create()* function is marked as part of the Threads option.

53077 IEEE PASC Interpretation 1003.1c #8 is applied, updating the DESCRIPTION.

53078 Issue 7

53079 The *pthread_key_create()* function is moved from the Threads option to the Base.

NAME

pthread_key_delete — thread-specific data key deletion

SYNOPSIS

```
#include <pthread.h>
```

```
int pthread_key_delete(pthread_key_t key);
```

DESCRIPTION

The *pthread_key_delete()* function shall delete a thread-specific data key previously returned by *pthread_key_create()*. The thread-specific data values associated with *key* need not be NULL at the time *pthread_key_delete()* is called. It is the responsibility of the application to free any application storage or perform any cleanup actions for data structures related to the deleted key or associated thread-specific data in any threads; this cleanup can be done either before or after *pthread_key_delete()* is called. Any attempt to use *key* following the call to *pthread_key_delete()* results in undefined behavior.

The *pthread_key_delete()* function shall be callable from within destructor functions. No destructor functions shall be invoked by *pthread_key_delete()*. Any destructor function that may have been associated with *key* shall no longer be called upon thread exit.

RETURN VALUE

If successful, the *pthread_key_delete()* function shall return zero; otherwise, an error number shall be returned to indicate the error.

ERRORS

The *pthread_key_delete()* function shall not return an error code of [EINTR].

EXAMPLES

None.

APPLICATION USAGE

None.

RATIONALE

A thread-specific data key deletion function has been included in order to allow the resources associated with an unused thread-specific data key to be freed. Unused thread-specific data keys can arise, among other scenarios, when a dynamically loaded module that allocated a key is unloaded.

Conforming applications are responsible for performing any cleanup actions needed for data structures associated with the key to be deleted, including data referenced by thread-specific data values. No such cleanup is done by *pthread_key_delete()*. In particular, destructor functions are not called. There are several reasons for this division of responsibility:

1. The associated destructor functions used to free thread-specific data at thread exit time are only guaranteed to work correctly when called in the thread that allocated the thread-specific data. (Destructors themselves may utilize thread-specific data.) Thus, they cannot be used to free thread-specific data in other threads at key deletion time. Attempting to have them called by other threads at key deletion time would require other threads to be asynchronously interrupted. But since interrupted threads could be in an arbitrary state, including holding locks necessary for the destructor to run, this approach would fail. In general, there is no safe mechanism whereby an implementation could free thread-specific data at key deletion time.
2. Even if there were a means of safely freeing thread-specific data associated with keys to be deleted, doing so would require that implementations be able to enumerate the threads with non-NULL data and potentially keep them from creating more thread-

specific data while the key deletion is occurring. This special case could cause extra synchronization in the normal case, which would otherwise be unnecessary.

For an application to know that it is safe to delete a key, it has to know that all the threads that might potentially ever use the key do not attempt to use it again. For example, it could know this if all the client threads have called a cleanup procedure declaring that they are through with the module that is being shut down, perhaps by setting a reference count to zero.

If an implementation detects that the value specified by the *key* argument to *pthread_key_delete()* does not refer to a key value obtained from *pthread_key_create()* or refers to a key that has been deleted with *pthread_key_delete()*, it is recommended that the function should fail and report an [EINVAL] error.

FUTURE DIRECTIONS

None.

SEE ALSO

pthread_key_create()

XBD <pthread.h>

CHANGE HISTORY

First released in Issue 5. Included for alignment with the POSIX Threads Extension.

Issue 6

The *pthread_key_delete()* function is marked as part of the Threads option.

Issue 7

The *pthread_key_delete()* function is moved from the Threads option to the Base.

The [EINVAL] error for a key value not obtained from *pthread_key_create()* or a key deleted with *pthread_key_delete()* is removed; this condition results in undefined behavior.

NAME

pthread_kill — send a signal to a thread

SYNOPSIS

```
#include <signal.h>
```

```
int pthread_kill(pthread_t thread, int sig);
```

DESCRIPTION

The *pthread_kill()* function shall request that a signal be delivered to the specified thread.

As in *kill()*, if *sig* is zero, error checking shall be performed but no signal shall actually be sent.

RETURN VALUE

Upon successful completion, the function shall return a value of zero. Otherwise, the function shall return an error number. If the *pthread_kill()* function fails, no signal shall be sent.

ERRORS

The *pthread_kill()* function shall fail if:

[EINVAL] The value of the *sig* argument is an invalid or unsupported signal number.

The *pthread_kill()* function shall not return an error code of [EINTR].

EXAMPLES

None.

APPLICATION USAGE

The *pthread_kill()* function provides a mechanism for asynchronously directing a signal at a thread in the calling process. This could be used, for example, by one thread to affect broadcast delivery of a signal to a set of threads.

Note that *pthread_kill()* only causes the signal to be handled in the context of the given thread; the signal action (termination or stopping) affects the process as a whole.

RATIONALE

If an implementation detects use of a thread ID after the end of its lifetime, it is recommended that the function should fail and report an [ESRCH] error.

FUTURE DIRECTIONS

None.

SEE ALSO

kill(), *pthread_self()*, *raise()*

XBD <signal.h>

CHANGE HISTORY

First released in Issue 5. Included for alignment with the POSIX Threads Extension.

Issue 6

The *pthread_kill()* function is marked as part of the Threads option.

The APPLICATION USAGE section is added.

Issue 7

The *pthread_kill()* function is moved from the Threads option to the Base.

Austin Group Interpretation 1003.1-2001 #142 is applied, removing the [ESRCH] error condition.

53188 NAME

53189 pthread_mutex_consistent — mark state protected by robust mutex as consistent

53190 SYNOPSIS

53191 #include <pthread.h>

53192 int pthread_mutex_consistent(pthread_mutex_t *mutex);

53193 DESCRIPTION

53194 If *mutex* is a robust mutex in an inconsistent state, the *pthread_mutex_consistent()* function can be
 53195 used to mark the state protected by the mutex referenced by *mutex* as consistent again.

53196 If an owner of a robust mutex terminates while holding the mutex, the mutex becomes
 53197 inconsistent and the next thread that acquires the mutex lock shall be notified of the state by the
 53198 return value [EOWNERDEAD]. In this case, the mutex does not become normally usable again
 53199 until the state is marked consistent.

53200 If the thread which acquired the mutex lock with the return value [EOWNERDEAD] terminates
 53201 before calling either *pthread_mutex_consistent()* or *pthread_mutex_unlock()*, the next thread that
 53202 acquires the mutex lock shall be notified about the state of the mutex by the return value
 53203 [EOWNERDEAD].

53204 The behavior is undefined if the value specified by the *mutex* argument to
 53205 *pthread_mutex_consistent()* does not refer to an initialized mutex.

53206 RETURN VALUE

53207 Upon successful completion, the *pthread_mutex_consistent()* function shall return zero.
 53208 Otherwise, an error value shall be returned to indicate the error.

53209 ERRORS

53210 The *pthread_mutex_consistent()* function shall fail if:

53211 [EINVAL] The mutex object referenced by *mutex* is not robust or does not protect an
 53212 inconsistent state.

53213 These functions shall not return an error code of [EINTR].

53214 EXAMPLES

53215 None.

53216 APPLICATION USAGE

53217 The *pthread_mutex_consistent()* function is only responsible for notifying the implementation that
 53218 the state protected by the mutex has been recovered and that normal operations with the mutex
 53219 can be resumed. It is the responsibility of the application to recover the state so it can be reused.
 53220 If the application is not able to perform the recovery, it can notify the implementation that the
 53221 situation is unrecoverable by a call to *pthread_mutex_unlock()* without a prior call to
 53222 *pthread_mutex_consistent()*, in which case subsequent threads that attempt to lock the mutex will
 53223 fail to acquire the lock and be returned [ENOTRECOVERABLE].

53224 RATIONALE

53225 If an implementation detects that the value specified by the *mutex* argument to
 53226 *pthread_mutex_consistent()* does not refer to an initialized mutex, it is recommended that the
 53227 function should fail and report an [EINVAL] error.

53228 FUTURE DIRECTIONS

53229 None.

53230 **SEE ALSO**

53231 [pthread_mutex_lock\(\)](#), [pthread_mutexattr_getrobust\(\)](#)

53232 XBD [<pthread.h>](#)

53233 **CHANGE HISTORY**

53234 First released in Issue 7.

NAME

pthread_mutex_destroy, pthread_mutex_init — destroy and initialize a mutex

SYNOPSIS

```
#include <pthread.h>

int pthread_mutex_destroy(pthread_mutex_t *mutex);
int pthread_mutex_init(pthread_mutex_t *restrict mutex,
    const pthread_mutexattr_t *restrict attr);
pthread_mutex_t mutex = PTHREAD_MUTEX_INITIALIZER;
```

DESCRIPTION

The *pthread_mutex_destroy()* function shall destroy the mutex object referenced by *mutex*; the mutex object becomes, in effect, uninitialized. An implementation may cause *pthread_mutex_destroy()* to set the object referenced by *mutex* to an invalid value.

A destroyed mutex object can be reinitialized using *pthread_mutex_init()*; the results of otherwise referencing the object after it has been destroyed are undefined.

It shall be safe to destroy an initialized mutex that is unlocked. Attempting to destroy a locked mutex or a mutex that is referenced (for example, while being used in a *pthread_cond_timedwait()* or *pthread_cond_wait()*) by another thread results in undefined behavior.

The *pthread_mutex_init()* function shall initialize the mutex referenced by *mutex* with attributes specified by *attr*. If *attr* is NULL, the default mutex attributes are used; the effect shall be the same as passing the address of a default mutex attributes object. Upon successful initialization, the state of the mutex becomes initialized and unlocked.

Only *mutex* itself may be used for performing synchronization. The result of referring to copies of *mutex* in calls to *pthread_mutex_lock()*, *pthread_mutex_trylock()*, *pthread_mutex_unlock()*, and *pthread_mutex_destroy()* is undefined.

Attempting to initialize an already initialized mutex results in undefined behavior.

In cases where default mutex attributes are appropriate, the macro PTHREAD_MUTEX_INITIALIZER can be used to initialize mutexes. The effect shall be equivalent to dynamic initialization by a call to *pthread_mutex_init()* with parameter *attr* specified as NULL, except that no error checks are performed.

The behavior is undefined if the value specified by the *mutex* argument to *pthread_mutex_destroy()* does not refer to an initialized mutex.

The behavior is undefined if the value specified by the *attr* argument to *pthread_mutex_init()* does not refer to an initialized mutex attributes object.

RETURN VALUE

If successful, the *pthread_mutex_destroy()* and *pthread_mutex_init()* functions shall return zero; otherwise, an error number shall be returned to indicate the error.

ERRORS

The *pthread_mutex_init()* function shall fail if:

[EAGAIN]	The system lacked the necessary resources (other than memory) to initialize another mutex.
[ENOMEM]	Insufficient memory exists to initialize the mutex.
[EPERM]	The caller does not have the privilege to perform the operation.

53277 The *pthread_mutex_init()* function may fail if:
53278 [EINVAL] The attributes object referenced by *attr* has the robust mutex attribute set
53279 without the process-shared attribute being set.

53280 These functions shall not return an error code of [EINTR].

53281 EXAMPLES

53282 None.

53283 APPLICATION USAGE

53284 None.

53285 RATIONALE

53286 If an implementation detects that the value specified by the *mutex* argument to
53287 *pthread_mutex_destroy()* does not refer to an initialized mutex, it is recommended that the
53288 function should fail and report an [EINVAL] error.

53289 If an implementation detects that the value specified by the *mutex* argument to
53290 *pthread_mutex_destroy()* or *pthread_mutex_init()* refers to a locked mutex or a mutex that is
53291 referenced (for example, while being used in a *pthread_cond_timedwait()* or *pthread_cond_wait()*)
53292 by another thread, or detects that the value specified by the *mutex* argument to
53293 *pthread_mutex_init()* refers to an already initialized mutex, it is recommended that the function
53294 should fail and report an [EBUSY] error.

53295 If an implementation detects that the value specified by the *attr* argument to
53296 *pthread_mutex_init()* does not refer to an initialized mutex attributes object, it is recommended
53297 that the function should fail and report an [EINVAL] error.

53298 Alternate Implementations Possible

53299 This volume of POSIX.1-2008 supports several alternative implementations of mutexes. An
53300 implementation may store the lock directly in the object of type **pthread_mutex_t**. Alternatively,
53301 an implementation may store the lock in the heap and merely store a pointer, handle, or unique
53302 ID in the mutex object. Either implementation has advantages or may be required on certain
53303 hardware configurations. So that portable code can be written that is invariant to this choice, this
53304 volume of POSIX.1-2008 does not define assignment or equality for this type, and it uses the
53305 term “initialize” to reinforce the (more restrictive) notion that the lock may actually reside in the
53306 mutex object itself.

53307 Note that this precludes an over-specification of the type of the mutex or condition variable and
53308 motivates the opaqueness of the type.

53309 An implementation is permitted, but not required, to have *pthread_mutex_destroy()* store an
53310 illegal value into the mutex. This may help detect erroneous programs that try to lock (or
53311 otherwise reference) a mutex that has already been destroyed.

53312 Tradeoff Between Error Checks and Performance Supported

53313 Many error conditions that can occur are not required to be detected by the implementation in
53314 order to let implementations trade off performance *versus* degree of error checking according to
53315 the needs of their specific applications and execution environment. As a general rule, conditions
53316 caused by the system (such as insufficient memory) are required to be detected, but conditions
53317 caused by an erroneously coded application (such as failing to provide adequate
53318 synchronization to prevent a mutex from being deleted while in use) are specified to result in
53319 undefined behavior.

53320 A wide range of implementations is thus made possible. For example, an implementation

intended for application debugging may implement all of the error checks, but an implementation running a single, provably correct application under very tight performance constraints in an embedded computer might implement minimal checks. An implementation might even be provided in two versions, similar to the options that compilers provide: a full-checking, but slower version; and a limited-checking, but faster version. To forbid this optionality would be a disservice to users.

By carefully limiting the use of “undefined behavior” only to things that an erroneous (badly coded) application might do, and by defining that resource-not-available errors are mandatory, this volume of POSIX.1-2008 ensures that a fully-conforming application is portable across the full range of implementations, while not forcing all implementations to add overhead to check for numerous things that a correct program never does. When the behavior is undefined, no error number is specified to be returned on implementations that do detect the condition. This is because undefined behavior means *anything* can happen, which includes returning with any value (which might happen to be a valid, but different, error number). However, since the error number might be useful to application developers when diagnosing problems during application development, a recommendation is made in rationale that implementors should return a particular error number if their implementation does detect the condition.

Why No Limits are Defined

Defining symbols for the maximum number of mutexes and condition variables was considered but rejected because the number of these objects may change dynamically. Furthermore, many implementations place these objects into application memory; thus, there is no explicit maximum.

Static Initializers for Mutexes and Condition Variables

Providing for static initialization of statically allocated synchronization objects allows modules with private static synchronization variables to avoid runtime initialization tests and overhead. Furthermore, it simplifies the coding of self-initializing modules. Such modules are common in C libraries, where for various reasons the design calls for self-initialization instead of requiring an explicit module initialization function to be called. An example use of static initialization follows.

Without static initialization, a self-initializing routine *foo()* might look as follows:

```
static pthread_once_t foo_once = PTHREAD_ONCE_INIT;
static pthread_mutex_t foo_mutex;

void foo_init()
{
    pthread_mutex_init(&foo_mutex, NULL);
}

void foo()
{
    pthread_once(&foo_once, foo_init);
    pthread_mutex_lock(&foo_mutex);
    /* Do work. */
    pthread_mutex_unlock(&foo_mutex);
}
```

With static initialization, the same routine could be coded as follows:

```
static pthread_mutex_t foo_mutex = PTHREAD_MUTEX_INITIALIZER;
```

```
53366 void foo()  
53367 {  
53368     pthread_mutex_lock(&foo_mutex);  
53369     /* Do work. */  
53370     pthread_mutex_unlock(&foo_mutex);  
53371 }
```

53372 Note that the static initialization both eliminates the need for the initialization test inside
53373 *pthread_once()* and the fetch of *&foo_mutex* to learn the address to be passed to
53374 *pthread_mutex_lock()* or *pthread_mutex_unlock()*.

53375 Thus, the C code written to initialize static objects is simpler on all systems and is also faster on a
53376 large class of systems; those where the (entire) synchronization object can be stored in
53377 application memory.

53378 Yet the locking performance question is likely to be raised for machines that require mutexes to
53379 be allocated out of special memory. Such machines actually have to have mutexes and possibly
53380 condition variables contain pointers to the actual hardware locks. For static initialization to work
53381 on such machines, *pthread_mutex_lock()* also has to test whether or not the pointer to the actual
53382 lock has been allocated. If it has not, *pthread_mutex_lock()* has to initialize it before use. The
53383 reservation of such resources can be made when the program is loaded, and hence return codes
53384 have not been added to mutex locking and condition variable waiting to indicate failure to
53385 complete initialization.

53386 This runtime test in *pthread_mutex_lock()* would at first seem to be extra work; an extra test is
53387 required to see whether the pointer has been initialized. On most machines this would actually
53388 be implemented as a fetch of the pointer, testing the pointer against zero, and then using the
53389 pointer if it has already been initialized. While the test might seem to add extra work, the extra
53390 effort of testing a register is usually negligible since no extra memory references are actually
53391 done. As more and more machines provide caches, the real expenses are memory references, not
53392 instructions executed.

53393 Alternatively, depending on the machine architecture, there are often ways to eliminate *all*
53394 overhead in the most important case: on the lock operations that occur *after* the lock has been
53395 initialized. This can be done by shifting more overhead to the less frequent operation:
53396 initialization. Since out-of-line mutex allocation also means that an address has to be
53397 dereferenced to find the actual lock, one technique that is widely applicable is to have static
53398 initialization store a bogus value for that address; in particular, an address that causes a machine
53399 fault to occur. When such a fault occurs upon the first attempt to lock such a mutex, validity
53400 checks can be done, and then the correct address for the actual lock can be filled in. Subsequent
53401 lock operations incur no extra overhead since they do not “fault”. This is merely one technique
53402 that can be used to support static initialization, while not adversely affecting the performance of
53403 lock acquisition. No doubt there are other techniques that are highly machine-dependent.

53404 The locking overhead for machines doing out-of-line mutex allocation is thus similar for
53405 modules being implicitly initialized, where it is improved for those doing mutex allocation
53406 entirely inline. The inline case is thus made much faster, and the out-of-line case is not
53407 significantly worse.

53408 Besides the issue of locking performance for such machines, a concern is raised that it is possible
53409 that threads would serialize contending for initialization locks when attempting to finish
53410 initializing statically allocated mutexes. (Such finishing would typically involve taking an
53411 internal lock, allocating a structure, storing a pointer to the structure in the mutex, and releasing
53412 the internal lock.) First, many implementations would reduce such serialization by hashing on
53413 the mutex address. Second, such serialization can only occur a bounded number of times. In

particular, it can happen at most as many times as there are statically allocated synchronization objects. Dynamically allocated objects would still be initialized via *pthread_mutex_init()* or *pthread_cond_init()*.

Finally, if none of the above optimization techniques for out-of-line allocation yields sufficient performance for an application on some implementation, the application can avoid static initialization altogether by explicitly initializing all synchronization objects with the corresponding *pthread_*_init()* functions, which are supported by all implementations. An implementation can also document the tradeoffs and advise which initialization technique is more efficient for that particular implementation.

Destroying Mutexes

A mutex can be destroyed immediately after it is unlocked. For example, consider the following code:

```
struct obj {
    pthread_mutex_t om;
    int refcnt;
    ...
};

obj_done(struct obj *op)
{
    pthread_mutex_lock(&op->om);
    if (--op->refcnt == 0) {
        pthread_mutex_unlock(&op->om);
        (A) pthread_mutex_destroy(&op->om);
        (B) free(op);
    } else
        (C) pthread_mutex_unlock(&op->om);
}
```

In this case *obj* is reference counted and *obj_done()* is called whenever a reference to the object is dropped. Implementations are required to allow an object to be destroyed and freed and potentially unmapped (for example, lines A and B) immediately after the object is unlocked (line C).

Robust Mutexes

Implementations are required to provide robust mutexes for mutexes with the process-shared attribute set to *PTHREAD_PROCESS_SHARED*. Implementations are allowed, but not required, to provide robust mutexes when the process-shared attribute is set to *PTHREAD_PROCESS_PRIVATE*.

FUTURE DIRECTIONS

None.

SEE ALSO

pthread_mutex_getprioceiling(), *pthread_mutexattr_getrobust()*, *pthread_mutex_lock()*,
pthread_mutex_timedlock(), *pthread_mutexattr_getpshared()*

XBD <pthread.h>

CHANGE HISTORY

First released in Issue 5. Included for alignment with the POSIX Threads Extension.

Issue 6

The *pthread_mutex_destroy()* and *pthread_mutex_init()* functions are marked as part of the Threads option.

The *pthread_mutex_timedlock()* function is added to the SEE ALSO section for alignment with IEEE Std 1003.1d-1999.

IEEE PASC Interpretation 1003.1c #34 is applied, updating the DESCRIPTION.

The **restrict** keyword is added to the *pthread_mutex_init()* prototype for alignment with the ISO/IEC 9899:1999 standard.

Issue 7

Changes are made from The Open Group Technical Standard, 2006, Extended API Set Part 3.

The *pthread_mutex_destroy()* and *pthread_mutex_init()* functions are moved from the Threads option to the Base.

The [EINVAL] error for an uninitialized mutex or an uninitialized mutex attributes object is removed; this condition results in undefined behavior.

The [EBUSY] error for a locked mutex, a mutex that is referenced, or an already initialized mutex is removed; this condition results in undefined behavior.

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0460 [70,428] is applied.

+

53475 **NAME**

53476 pthread_mutex_getprioceiling, pthread_mutex_setprioceiling — get and set the priority ceiling
 53477 of a mutex (**REALTIME THREADS**)

53478 **SYNOPSIS**

```
53479 RPP|TPP #include <pthread.h>

53480 int pthread_mutex_getprioceiling(const pthread_mutex_t *restrict mutex,
53481 int *restrict prioceiling);
53482 int pthread_mutex_setprioceiling(pthread_mutex_t *restrict mutex,
53483 int prioceiling, int *restrict old_ceiling);
```

53484 **DESCRIPTION**

53485 The *pthread_mutex_getprioceiling()* function shall return the current priority ceiling of the mutex.

53486 The *pthread_mutex_setprioceiling()* function shall attempt to lock the mutex as if by a call to
 53487 *pthread_mutex_lock()*, except that the process of locking the mutex need not adhere to the priority
 53488 protect protocol. On acquiring the mutex it shall change the mutex's priority ceiling and then
 53489 release the mutex as if by a call to *pthread_mutex_unlock()*. When the change is successful, the
 53490 previous value of the priority ceiling shall be returned in *old_ceiling*.

53491 If the *pthread_mutex_setprioceiling()* function fails, the mutex priority ceiling shall not be
 53492 changed.

53493 **RETURN VALUE**

53494 If successful, the *pthread_mutex_getprioceiling()* and *pthread_mutex_setprioceiling()* functions shall
 53495 return zero; otherwise, an error number shall be returned to indicate the error.

53496 **ERRORS**

53497 These functions shall fail if:

53498	[EINVAL]	The protocol attribute of <i>mutex</i> is PTHREAD_PRIO_NONE.
53499	[EPERM]	The implementation requires appropriate privileges to perform the operation
53500		and the caller does not have appropriate privileges.

53501 The *pthread_mutex_setprioceiling()* function shall fail if:

53502	[EAGAIN]	The mutex could not be acquired because the maximum number of recursive
53503		locks for <i>mutex</i> has been exceeded.

53504	[EDEADLK]	The mutex type is PTHREAD_MUTEX_ERRORCHECK and the current
53505		thread already owns the mutex.

53506	[EINVAL]	The mutex was created with the protocol attribute having the value
53507		PTHREAD_PRIO_PROTECT and the calling thread's priority is higher than
53508		the mutex's current priority ceiling, and the implementation adheres to the
53509		priority protect protocol in the process of locking the mutex.

53510	[ENOTRECOVERABLE]	

53511		The mutex is a robust mutex and the state protected by the mutex is not
53512		recoverable.

53513	[EOWNERDEAD]	

53514		The mutex is a robust mutex and the process containing the previous owning
53515		thread terminated while holding the mutex lock. The mutex lock shall be
53516		acquired by the calling thread and it is up to the new owner to make the state
53517		consistent (see <i>pthread_mutex_lock()</i>).

53518 The *pthread_mutex_setprioceiling()* function may fail if:

53519 [EDEADLK] A deadlock condition was detected.

53520 [EINVAL] The priority requested by *prioceiling* is out of range.

53521 [EOWNERDEAD]

53522 The mutex is a robust mutex and the previous owning thread terminated

53523 while holding the mutex lock. The mutex lock shall be acquired by the calling

53524 thread and it is up to the new owner to make the state consistent (see

53525 *pthread_mutex_lock()*).

53526 These functions shall not return an error code of [EINTR].

53527 EXAMPLES

53528 None.

53529 APPLICATION USAGE

53530 None.

53531 RATIONALE

53532 None.

53533 FUTURE DIRECTIONS

53534 None.

53535 SEE ALSO

53536 *pthread_mutex_destroy()*, *pthread_mutex_lock()*, *pthread_mutex_timedlock()*

53537 XBD <pthread.h>

53538 CHANGE HISTORY

53539 First released in Issue 5. Included for alignment with the POSIX Threads Extension.

53540 Marked as part of the Realtime Threads Feature Group.

53541 Issue 6

53542 The *pthread_mutex_getprioceiling()* and *pthread_mutex_setprioceiling()* functions are marked as

53543 part of the Threads and Thread Priority Protection options.

53544 The [ENOSYS] error conditions have been removed.

53545 The *pthread_mutex_timedlock()* function is added to the SEE ALSO section for alignment with

53546 IEEE Std 1003.1d-1999.

53547 The **restrict** keyword is added to the *pthread_mutex_getprioceiling()* and

53548 *pthread_mutex_setprioceiling()* prototypes for alignment with the ISO/IEC 9899:1999 standard.

53549 Issue 7

53550 SD5-XSH-ERN-39 is applied.

53551 Austin Group Interpretation 1003.1-2001 #052 is applied, adding [EDEADLK] as a “may fail”

53552 error.

53553 SD5-XSH-ERN-158 is applied, updating the ERRORS section to include a “shall fail” error case

53554 for when the protocol attribute of *mutex* is PTHREAD_PRIO_NONE.

53555 The *pthread_mutex_getprioceiling()* and *pthread_mutex_setprioceiling()* functions are moved from

53556 the Threads option to require support of either the Robust Mutex Priority Protection option or

53557 the Non-Robust Mutex Priority Protection option.

53558
53559

The DESCRIPTION and ERRORS sections are updated to account properly for all of the various mutex types.

53560 NAME

53561 pthread_mutex_init — destroy and initialize a mutex

53562 SYNOPSIS

```
53563     #include <pthread.h>

53564     int pthread_mutex_init(pthread_mutex_t *restrict mutex,
53565         const pthread_mutexattr_t *restrict attr);
53566     pthread_mutex_t mutex = PTHREAD_MUTEX_INITIALIZER;
```

53567 DESCRIPTION

53568 Refer to *pthread_mutex_destroy()*.

NAME

pthread_mutex_lock, pthread_mutex_trylock, pthread_mutex_unlock — lock and unlock a mutex

SYNOPSIS

```
#include <pthread.h>

int pthread_mutex_lock(pthread_mutex_t *mutex);
int pthread_mutex_trylock(pthread_mutex_t *mutex);
int pthread_mutex_unlock(pthread_mutex_t *mutex);
```

DESCRIPTION

The mutex object referenced by *mutex* shall be locked by a call to *pthread_mutex_lock()* that returns zero or [EOWNERDEAD]. If the mutex is already locked by another thread, the calling thread shall block until the mutex becomes available. This operation shall return with the mutex object referenced by *mutex* in the locked state with the calling thread as its owner. If a thread attempts to relock a mutex that it has already locked, *pthread_mutex_lock()* shall behave as described in the **Relock** column of the following table. If a thread attempts to unlock a mutex that it has not locked or a mutex which is unlocked, *pthread_mutex_unlock()* shall behave as described in the **Unlock When Not Owner** column of the following table.

Mutex Type	Robustness	Relock	Unlock When Not Owner
NORMAL	non-robust	deadlock	undefined behavior
NORMAL	robust	deadlock	error returned
ERRORCHECK	either	error returned	error returned
RECURSIVE	either	recursive (see below)	error returned
DEFAULT	non-robust	undefined behavior†	undefined behavior†
DEFAULT	robust	undefined behavior†	error returned

† If the mutex type is PTHREAD_MUTEX_DEFAULT, the behavior of *pthread_mutex_lock()* may correspond to one of the three other standard mutex types as described in the table above. If it does not correspond to one of those three, the behavior is undefined for the cases marked †.

Where the table indicates recursive behavior, the mutex shall maintain the concept of a lock count. When a thread successfully acquires a mutex for the first time, the lock count shall be set to one. Every time a thread relocks this mutex, the lock count shall be incremented by one. Each time the thread unlocks the mutex, the lock count shall be decremented by one. When the lock count reaches zero, the mutex shall become available for other threads to acquire.

The *pthread_mutex_trylock()* function shall be equivalent to *pthread_mutex_lock()*, except that if the mutex object referenced by *mutex* is currently locked (by any thread, including the current thread), the call shall return immediately. If the mutex type is PTHREAD_MUTEX_RECURSIVE and the mutex is currently owned by the calling thread, the mutex lock count shall be incremented by one and the *pthread_mutex_trylock()* function shall immediately return success.

The *pthread_mutex_unlock()* function shall release the mutex object referenced by *mutex*. The manner in which a mutex is released is dependent upon the mutex's type attribute. If there are threads blocked on the mutex object referenced by *mutex* when *pthread_mutex_unlock()* is called, resulting in the mutex becoming available, the scheduling policy shall determine which thread shall acquire the mutex.

53615 (In the case of PTHREAD_MUTEX_RECURSIVE mutexes, the mutex shall become available
53616 when the count reaches zero and the calling thread no longer has any locks on this mutex.)

53617 If a signal is delivered to a thread waiting for a mutex, upon return from the signal handler the
53618 thread shall resume waiting for the mutex as if it was not interrupted.

53619 If *mutex* is a robust mutex and the process containing the owning thread terminated while
53620 holding the mutex lock, a call to *pthread_mutex_lock()* shall return the error value
53621 [EOWNERDEAD]. If *mutex* is a robust mutex and the owning thread terminated while holding
53622 the mutex lock, a call to *pthread_mutex_lock()* may return the error value [EOWNERDEAD] even
53623 if the process in which the owning thread resides has not terminated. In these cases, the mutex is
53624 locked by the thread but the state it protects is marked as inconsistent. The application should
53625 ensure that the state is made consistent for reuse and when that is complete call
53626 *pthread_mutex_consistent()*. If the application is unable to recover the state, it should unlock the
53627 mutex without a prior call to *pthread_mutex_consistent()*, after which the mutex is marked
53628 permanently unusable.

53629 If *mutex* does not refer to an initialized mutex object, the behavior of *pthread_mutex_lock()*,
53630 *pthread_mutex_trylock()*, and *pthread_mutex_unlock()* is undefined.

53631 RETURN VALUE

53632 If successful, the *pthread_mutex_lock()*, *pthread_mutex_trylock()*, and *pthread_mutex_unlock()*
53633 functions shall return zero; otherwise, an error number shall be returned to indicate the error. -

53634 ERRORS

53635 The *pthread_mutex_lock()* and *pthread_mutex_trylock()* functions shall fail if:

53636 [EAGAIN] The mutex could not be acquired because the maximum number of recursive
53637 locks for *mutex* has been exceeded.

53638 RPP|TPP [EINVAL] The *mutex* was created with the protocol attribute having the value
53639 PTHREAD_PRIO_PROTECT and the calling thread's priority is higher than
53640 the mutex's current priority ceiling.

53641 [ENOTRECOVERABLE]
53642 The state protected by the mutex is not recoverable.

53643 [EOWNERDEAD]
53644 The mutex is a robust mutex and the process containing the previous owning
53645 thread terminated while holding the mutex lock. The mutex lock shall be
53646 acquired by the calling thread and it is up to the new owner to make the state
53647 consistent.

53648 The *pthread_mutex_lock()* function shall fail if:

53649 [EDEADLK] The mutex type is PTHREAD_MUTEX_ERRORCHECK and the current
53650 thread already owns the mutex.

53651 The *pthread_mutex_trylock()* function shall fail if:

53652 [EBUSY] The *mutex* could not be acquired because it was already locked.

53653 The *pthread_mutex_unlock()* function shall fail if:

53654 [EPERM] The mutex type is PTHREAD_MUTEX_ERRORCHECK or |
53655 PTHREAD_MUTEX_RECURSIVE, or the mutex is a robust mutex, and the |
53656 current thread does not own the mutex.

53657 The *pthread_mutex_lock()* and *pthread_mutex_trylock()* functions may fail if:

[EOWNERDEAD]

The mutex is a robust mutex and the previous owning thread terminated while holding the mutex lock. The mutex lock shall be acquired by the calling thread and it is up to the new owner to make the state consistent.

The *pthread_mutex_lock()* function may fail if:

[EDEADLK] A deadlock condition was detected.

These functions shall not return an error code of [EINTR].

EXAMPLES

None.

APPLICATION USAGE

Applications that have assumed that non-zero return values are errors will need updating for use with robust mutexes, since a valid return for a thread acquiring a mutex which is protecting a currently inconsistent state is [EOWNERDEAD]. Applications that do not check the error returns, due to ruling out the possibility of such errors arising, should not use robust mutexes. If an application is supposed to work with normal and robust mutexes it should check all return values for error conditions and if necessary take appropriate action.

RATIONALE

Mutex objects are intended to serve as a low-level primitive from which other thread synchronization functions can be built. As such, the implementation of mutexes should be as efficient as possible, and this has ramifications on the features available at the interface.

The mutex functions and the particular default settings of the mutex attributes have been motivated by the desire to not preclude fast, inlined implementations of mutex locking and unlocking.

Since most attributes only need to be checked when a thread is going to be blocked, the use of attributes does not slow the (common) mutex-locking case.

Likewise, while being able to extract the thread ID of the owner of a mutex might be desirable, it would require storing the current thread ID when each mutex is locked, and this could incur unacceptable levels of overhead. Similar arguments apply to a *mutex_tryunlock* operation.

For further rationale on the extended mutex types, see XRAT [Threads Extensions](#) (on page 3601).

If an implementation detects that the value specified by the *mutex* argument does not refer to an initialized mutex object, it is recommended that the function should fail and report an [EINVAL] error.

FUTURE DIRECTIONS

None.

SEE ALSO

[pthread_mutex_consistent\(\)](#), [pthread_mutex_destroy\(\)](#), [pthread_mutex_timedlock\(\)](#),
[pthread_mutexattr_getrobust\(\)](#)

XBD [Section 4.11](#) (on page 110), [<pthread.h>](#)

CHANGE HISTORY

First released in Issue 5. Included for alignment with the POSIX Threads Extension.

Issue 6

The *pthread_mutex_lock()*, *pthread_mutex_trylock()*, and *pthread_mutex_unlock()* functions are marked as part of the Threads option.

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The behavior when attempting to relock a mutex is defined.

The *pthread_mutex_timedlock()* function is added to the SEE ALSO section for alignment with IEEE Std 1003.1d-1999.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/98 is applied, updating the ERRORS section so that the [EDEADLK] error includes detection of a deadlock condition. The RATIONALE section is also reworded to take into account non-XSI-conformant systems.

Issue 7

SD5-XSH-ERN-43 is applied, marking the “shall fail” case of the [EINVAL] error as dependent on the Thread Priority Protection option.

Changes are made from The Open Group Technical Standard, 2006, Extended API Set Part 3.

The *pthread_mutex_lock()*, *pthread_mutex_trylock()*, and *pthread_mutex_unlock()* functions are moved from the Threads option to the Base.

The following extended mutex types are moved from the XSI option to the Base:

```
PTHREAD_MUTEX_NORMAL
PTHREAD_MUTEX_ERRORCHECK
PTHREAD_MUTEX_RECURSIVE
PTHREAD_MUTEX_DEFAULT
```

The DESCRIPTION is updated to clarify the behavior when *mutex* does not refer to an initialized mutex.

The ERRORS section is updated to account properly for all of the various mutex types.

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0461 [121], XSH/TC1-2008/0462 + [92,428], and XSH/TC1-2008/0463 [121] are applied.

53725 **NAME**

53726 pthread_mutex_setprioceiling — change the priority ceiling of a mutex (**REALTIME**
53727 **THREADS**)

53728 **SYNOPSIS**

```
53729 RPP|TPP #include <pthread.h>  
53730 int pthread_mutex_setprioceiling(pthread_mutex_t *restrict mutex,  
53731 int prioceiling, int *restrict old_ceiling);
```

53732 **DESCRIPTION**

53733 Refer to *pthread_mutex_getprioceiling()*.

NAME

pthread_mutex_timedlock — lock a mutex

SYNOPSIS

```
#include <pthread.h>
#include <time.h>

int pthread_mutex_timedlock(pthread_mutex_t *restrict mutex,
    const struct timespec *restrict abstime);
```

DESCRIPTION

The *pthread_mutex_timedlock()* function shall lock the mutex object referenced by *mutex*. If the mutex is already locked, the calling thread shall block until the mutex becomes available as in the *pthread_mutex_lock()* function. If the mutex cannot be locked without waiting for another thread to unlock the mutex, this wait shall be terminated when the specified timeout expires.

The timeout shall expire when the absolute time specified by *abstime* passes, as measured by the clock on which timeouts are based (that is, when the value of that clock equals or exceeds *abstime*), or if the absolute time specified by *abstime* has already been passed at the time of the call.

The timeout shall be based on the CLOCK_REALTIME clock. The resolution of the timeout shall be the resolution of the clock on which it is based. The **timespec** data type is defined in the **<time.h>** header.

Under no circumstance shall the function fail with a timeout if the mutex can be locked immediately. The validity of the *abstime* parameter need not be checked if the mutex can be locked immediately.

RPI | TPI

As a consequence of the priority inheritance rules (for mutexes initialized with the PRIO_INHERIT protocol), if a timed mutex wait is terminated because its timeout expires, the priority of the owner of the mutex shall be adjusted as necessary to reflect the fact that this thread is no longer among the threads waiting for the mutex.

If *mutex* is a robust mutex and the process containing the owning thread terminated while holding the mutex lock, a call to *pthread_mutex_timedlock()* shall return the error value [EOWNERDEAD]. If *mutex* is a robust mutex and the owning thread terminated while holding the mutex lock, a call to *pthread_mutex_timedlock()* may return the error value [EOWNERDEAD] even if the process in which the owning thread resides has not terminated. In these cases, the mutex is locked by the thread but the state it protects is marked as inconsistent. The application should ensure that the state is made consistent for reuse and when that is complete call *pthread_mutex_consistent()*. If the application is unable to recover the state, it should unlock the mutex without a prior call to *pthread_mutex_consistent()*, after which the mutex is marked permanently unusable.

If *mutex* does not refer to an initialized mutex object, the behavior is undefined.

RETURN VALUE

If successful, the *pthread_mutex_timedlock()* function shall return zero; otherwise, an error number shall be returned to indicate the error.

ERRORS

The *pthread_mutex_timedlock()* function shall fail if:

[EAGAIN] The mutex could not be acquired because the maximum number of recursive locks for *mutex* has been exceeded.

53778	[EDEADLK]	The mutex type is PTHREAD_MUTEX_ERRORCHECK and the current thread already owns the mutex.
53779		
53780	[EINVAL]	The mutex was created with the protocol attribute having the value PTHREAD_PRIO_PROTECT and the calling thread's priority is higher than the mutex' current priority ceiling.
53781		
53782		
53783	[EINVAL]	The process or thread would have blocked, and the <i>abstime</i> parameter specified a nanoseconds field value less than zero or greater than or equal to 1 000 million.
53784		
53785		
53786	[ENOTRECOVERABLE]	
53787		The state protected by the mutex is not recoverable.
53788	[EOWNERDEAD]	
53789		The mutex is a robust mutex and the process containing the previous owning thread terminated while holding the mutex lock. The mutex lock shall be acquired by the calling thread and it is up to the new owner to make the state consistent.
53790		
53791		
53792		
53793	[ETIMEDOUT]	The mutex could not be locked before the specified timeout expired.
53794		The <i>pthread_mutex_timedlock()</i> function may fail if:
53795	[EDEADLK]	A deadlock condition was detected.
53796	[EOWNERDEAD]	
53797		The mutex is a robust mutex and the previous owning thread terminated while holding the mutex lock. The mutex lock shall be acquired by the calling thread and it is up to the new owner to make the state consistent.
53798		
53799		
53800		This function shall not return an error code of [EINTR].

EXAMPLES

None.

APPLICATION USAGE

Applications that have assumed that non-zero return values are errors will need updating for use with robust mutexes, since a valid return for a thread acquiring a mutex which is protecting a currently inconsistent state is [EOWNERDEAD]. Applications that do not check the error returns, due to ruling out the possibility of such errors arising, should not use robust mutexes. If an application is supposed to work with normal and robust mutexes, it should check all return values for error conditions and if necessary take appropriate action.

RATIONALERefer to *pthread_mutex_lock()*.**FUTURE DIRECTIONS**

None.

SEE ALSO*pthread_mutex_destroy()*, *pthread_mutex_lock()*, *time()*XBD Section 4.11 (on page 110), **<pthread.h>**, **<time.h>****CHANGE HISTORY**

First released in Issue 6. Derived from IEEE Std 1003.1d-1999.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/99 is applied, marking the last paragraph in the DESCRIPTION as part of the Thread Priority Inheritance option.

- 53821 IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/100 is applied, updating the ERRORS
53822 section so that the [EDEADLK] error includes detection of a deadlock condition.
- 53823 **Issue 7**
- 53824 Changes are made from The Open Group Technical Standard, 2006, Extended API Set Part 3.
- 53825 The *pthread_mutex_timedlock()* function is moved from the Timeouts option to the Base.
- 53826 Functionality relating to the Timers option is moved to the Base.
- 53827 The DESCRIPTION is updated to clarify the behavior when *mutex* does not refer to an initialized
53828 mutex.
- 53829 The ERRORS section is updated to account properly for all of the various mutex types.

53830 **NAME**

53831 pthread_mutex_trylock, pthread_mutex_unlock — lock and unlock a mutex

53832 **SYNOPSIS**

53833 #include <pthread.h>

53834 int pthread_mutex_trylock(pthread_mutex_t *mutex);

53835 int pthread_mutex_unlock(pthread_mutex_t *mutex);

53836 **DESCRIPTION**53837 Refer to *pthread_mutex_lock()*.

NAME

pthread_mutexattr_destroy, pthread_mutexattr_init — destroy and initialize the mutex attributes object

SYNOPSIS

```
#include <pthread.h>

int pthread_mutexattr_destroy(pthread_mutexattr_t *attr);
int pthread_mutexattr_init(pthread_mutexattr_t *attr);
```

DESCRIPTION

The *pthread_mutexattr_destroy()* function shall destroy a mutex attributes object; the object becomes, in effect, uninitialized. An implementation may cause *pthread_mutexattr_destroy()* to set the object referenced by *attr* to an invalid value.

A destroyed *attr* attributes object can be reinitialized using *pthread_mutexattr_init()*; the results of otherwise referencing the object after it has been destroyed are undefined.

The *pthread_mutexattr_init()* function shall initialize a mutex attributes object *attr* with the default value for all of the attributes defined by the implementation.

Results are undefined if *pthread_mutexattr_init()* is called specifying an already initialized *attr* attributes object.

After a mutex attributes object has been used to initialize one or more mutexes, any function affecting the attributes object (including destruction) shall not affect any previously initialized mutexes.

The behavior is undefined if the value specified by the *attr* argument to *pthread_mutexattr_destroy()* does not refer to an initialized mutex attributes object.

RETURN VALUE

Upon successful completion, *pthread_mutexattr_destroy()* and *pthread_mutexattr_init()* shall return zero; otherwise, an error number shall be returned to indicate the error.

ERRORS

The *pthread_mutexattr_init()* function shall fail if:

[ENOMEM] Insufficient memory exists to initialize the mutex attributes object.

These functions shall not return an error code of [EINTR].

EXAMPLES

None.

APPLICATION USAGE

None.

RATIONALE

If an implementation detects that the value specified by the *attr* argument to *pthread_mutexattr_destroy()* does not refer to an initialized mutex attributes object, it is recommended that the function should fail and report an [EINVAL] error.

See *pthread_attr_destroy()* for a general explanation of attributes. Attributes objects allow implementations to experiment with useful extensions and permit extension of this volume of POSIX.1-2008 without changing the existing functions. Thus, they provide for future extensibility of this volume of POSIX.1-2008 and reduce the temptation to standardize prematurely on semantics that are not yet widely implemented or understood.

Examples of possible additional mutex attributes that have been discussed are *spin_only*, *limited_spin*, *no_spin*, *recursive*, and *metered*. (To explain what the latter attributes might mean:

recursive mutexes would allow for multiple re-locking by the current owner; metered mutexes would transparently keep records of queue length, wait time, and so on.) Since there is not yet wide agreement on the usefulness of these resulting from shared implementation and usage experience, they are not yet specified in this volume of POSIX.1-2008. Mutex attributes objects, however, make it possible to test out these concepts for possible standardization at a later time.

Mutex Attributes and Performance

Care has been taken to ensure that the default values of the mutex attributes have been defined such that mutexes initialized with the defaults have simple enough semantics so that the locking and unlocking can be done with the equivalent of a test-and-set instruction (plus possibly a few other basic instructions).

There is at least one implementation method that can be used to reduce the cost of testing at lock-time if a mutex has non-default attributes. One such method that an implementation can employ (and this can be made fully transparent to fully conforming POSIX applications) is to secretly pre-lock any mutexes that are initialized to non-default attributes. Any later attempt to lock such a mutex causes the implementation to branch to the “slow path” as if the mutex were unavailable; then, on the slow path, the implementation can do the “real work” to lock a non-default mutex. The underlying unlock operation is more complicated since the implementation never really wants to release the pre-lock on this kind of mutex. This illustrates that, depending on the hardware, there may be certain optimizations that can be used so that whatever mutex attributes are considered “most frequently used” can be processed most efficiently.

Process Shared Memory and Synchronization

The existence of memory mapping functions in this volume of POSIX.1-2008 leads to the possibility that an application may allocate the synchronization objects from this section in memory that is accessed by multiple processes (and therefore, by threads of multiple processes).

In order to permit such usage, while at the same time keeping the usual case (that is, usage within a single process) efficient, a *process-shared* option has been defined.

If an implementation supports the `_POSIX_THREAD_PROCESS_SHARED` option, then the *process-shared* attribute can be used to indicate that mutexes or condition variables may be accessed by threads of multiple processes.

The default setting of `PTHREAD_PROCESS_PRIVATE` has been chosen for the *process-shared* attribute so that the most efficient forms of these synchronization objects are created by default.

Synchronization variables that are initialized with the `PTHREAD_PROCESS_PRIVATE` *process-shared* attribute may only be operated on by threads in the process that initialized them. Synchronization variables that are initialized with the `PTHREAD_PROCESS_SHARED` *process-shared* attribute may be operated on by any thread in any process that has access to it. In particular, these processes may exist beyond the lifetime of the initializing process. For example, the following code implements a simple counting semaphore in a mapped file that may be used by many processes.

```
/* sem.h */
struct semaphore {
    pthread_mutex_t lock;
    pthread_cond_t nonzero;
    unsigned count;
};
typedef struct semaphore semaphore_t;
```

```

53927 semaphore_t *semaphore_create(char *semaphore_name);
53928 semaphore_t *semaphore_open(char *semaphore_name);
53929 void semaphore_post(semaphore_t *semap);
53930 void semaphore_wait(semaphore_t *semap);
53931 void semaphore_close(semaphore_t *semap);

53932 /* sem.c */
53933 #include <sys/types.h>
53934 #include <sys/stat.h>
53935 #include <sys/mman.h>
53936 #include <fcntl.h>
53937 #include <pthread.h>
53938 #include "sem.h"

53939 semaphore_t *
53940 semaphore_create(char *semaphore_name)
53941 {
53942     int fd;
53943     semaphore_t *semap;
53944     pthread_mutexattr_t psharedm;
53945     pthread_condattr_t psharedc;

53946     fd = open(semaphore_name, O_RDWR | O_CREAT | O_EXCL, 0666);
53947     if (fd < 0)
53948         return (NULL);
53949     (void) ftruncate(fd, sizeof(semaphore_t));
53950     (void) pthread_mutexattr_init(&psharedm);
53951     (void) pthread_mutexattr_setpshared(&psharedm,
53952         PTHREAD_PROCESS_SHARED);
53953     (void) pthread_condattr_init(&psharedc);
53954     (void) pthread_condattr_setpshared(&psharedc,
53955         PTHREAD_PROCESS_SHARED);
53956     semap = (semaphore_t *) mmap(NULL, sizeof(semaphore_t),
53957         PROT_READ | PROT_WRITE, MAP_SHARED,
53958         fd, 0);
53959     close (fd);
53960     (void) pthread_mutex_init(&semap->lock, &psharedm);
53961     (void) pthread_cond_init(&semap->nonzero, &psharedc);
53962     semap->count = 0;
53963     return (semap);
53964 }

53965 semaphore_t *
53966 semaphore_open(char *semaphore_name)
53967 {
53968     int fd;
53969     semaphore_t *semap;

53970     fd = open(semaphore_name, O_RDWR, 0666);
53971     if (fd < 0)
53972         return (NULL);
53973     semap = (semaphore_t *) mmap(NULL, sizeof(semaphore_t),
53974         PROT_READ | PROT_WRITE, MAP_SHARED,
53975         fd, 0);

```

```

53976         close (fd);
53977         return (semap);
53978     }

53979 void
53980 semaphore_post(semaphore_t *semap)
53981 {
53982     pthread_mutex_lock(&semap->lock);
53983     if (semap->count == 0)
53984         pthread_cond_signal(&semap->nonzero);
53985     semap->count++;
53986     pthread_mutex_unlock(&semap->lock);
53987 }

53988 void
53989 semaphore_wait(semaphore_t *semap)
53990 {
53991     pthread_mutex_lock(&semap->lock);
53992     while (semap->count == 0)
53993         pthread_cond_wait(&semap->nonzero, &semap->lock);
53994     semap->count--;
53995     pthread_mutex_unlock(&semap->lock);
53996 }

53997 void
53998 semaphore_close(semaphore_t *semap)
53999 {
54000     munmap((void *) semap, sizeof(semaphore_t));
54001 }

```

The following code is for three separate processes that create, post, and wait on a semaphore in the file **/tmp/semaphore**. Once the file is created, the post and wait programs increment and decrement the counting semaphore (waiting and waking as required) even though they did not initialize the semaphore.

```

54006 /* create.c */
54007 #include "pthread.h"
54008 #include "sem.h"

54009 int
54010 main()
54011 {
54012     semaphore_t *semap;

54013     semap = semaphore_create("/tmp/semaphore");
54014     if (semap == NULL)
54015         exit(1);
54016     semaphore_close(semap);
54017     return (0);
54018 }

54019 /* post */
54020 #include "pthread.h"
54021 #include "sem.h"

54022 int

```

```

54023     main()
54024     {
54025         semaphore_t *semaph;
54026
54027         semaph = semaphore_open("/tmp/semaphore");
54028         if (semaph == NULL)
54029             exit(1);
54030         semaphore_post(semaph);
54031         semaphore_close(semaph);
54032         return (0);
54033     }
54034
54035     /* wait */
54036     #include "pthread.h"
54037     #include "sem.h"
54038
54039     int
54040     main()
54041     {
54042         semaphore_t *semaph;
54043
54044         semaph = semaphore_open("/tmp/semaphore");
54045         if (semaph == NULL)
54046             exit(1);
54047         semaphore_wait(semaph);
54048         semaphore_close(semaph);
54049         return (0);
54050     }

```

FUTURE DIRECTIONS

None.

SEE ALSO

[*pthread_cond_destroy\(\)*](#), [*pthread_create\(\)*](#), [*pthread_mutex_destroy\(\)*](#)

XBD [**<pthread.h>**](#)

CHANGE HISTORY

First released in Issue 5. Included for alignment with the POSIX Threads Extension.

Issue 6

The *pthread_mutexattr_destroy()* and *pthread_mutexattr_init()* functions are marked as part of the Threads option.

IEEE PASC Interpretation 1003.1c #27 is applied, updating the ERRORS section.

Issue 7

The *pthread_mutexattr_destroy()* and *pthread_mutexattr_init()* functions are moved from the Threads option to the Base.

The [EINVAL] error for an uninitialized mutex attributes object is removed; this condition results in undefined behavior.

NAME

pthread_mutexattr_getprioceiling, pthread_mutexattr_setprioceiling — get and set the prioceiling attribute of the mutex attributes object (**REALTIME THREADS**)

SYNOPSIS

```
RPP|TPP #include <pthread.h>
```

```
int pthread_mutexattr_getprioceiling(const pthread_mutexattr_t
    *restrict attr, int *restrict prioceiling);
int pthread_mutexattr_setprioceiling(pthread_mutexattr_t *attr,
    int prioceiling);
```

DESCRIPTION

The *pthread_mutexattr_getprioceiling()* and *pthread_mutexattr_setprioceiling()* functions, respectively, shall get and set the priority ceiling attribute of a mutex attributes object pointed to by *attr* which was previously created by the function *pthread_mutexattr_init()*.

The *prioceiling* attribute contains the priority ceiling of initialized mutexes. The values of *prioceiling* are within the maximum range of priorities defined by SCHED_FIFO.

The *prioceiling* attribute defines the priority ceiling of initialized mutexes, which is the minimum priority level at which the critical section guarded by the mutex is executed. In order to avoid priority inversion, the priority ceiling of the mutex shall be set to a priority higher than or equal to the highest priority of all the threads that may lock that mutex. The values of *prioceiling* are within the maximum range of priorities defined under the SCHED_FIFO scheduling policy.

The behavior is undefined if the value specified by the *attr* argument to *pthread_mutexattr_getprioceiling()* or *pthread_mutexattr_setprioceiling()* does not refer to an initialized mutex attributes object.

RETURN VALUE

Upon successful completion, the *pthread_mutexattr_getprioceiling()* and *pthread_mutexattr_setprioceiling()* functions shall return zero; otherwise, an error number shall be returned to indicate the error.

ERRORS

These functions may fail if:

[EINVAL] The value specified by *prioceiling* is invalid.

[EPERM] The caller does not have the privilege to perform the operation.

These functions shall not return an error code of [EINTR].

EXAMPLES

None.

APPLICATION USAGE

None.

RATIONALE

If an implementation detects that the value specified by the *attr* argument to *pthread_mutexattr_getprioceiling()* or *pthread_mutexattr_setprioceiling()* does not refer to an initialized mutex attributes object, it is recommended that the function should fail and report an [EINVAL] error.

54104 **FUTURE DIRECTIONS**

54105 None.

54106 **SEE ALSO**54107 *pthread_cond_destroy()*, *pthread_create()*, *pthread_mutex_destroy()*

54108 XBD <pthread.h>

54109 **CHANGE HISTORY**

54110 First released in Issue 5. Included for alignment with the POSIX Threads Extension.

54111 Marked as part of the Realtime Threads Feature Group.

54112 **Issue 6**54113 The *pthread_mutexattr_getprioceiling()* and *pthread_mutexattr_setprioceiling()* functions are marked
54114 as part of the Threads and Thread Priority Protection options.54115 The [ENOSYS] error condition has been removed as stubs need not be provided if an
54116 implementation does not support the Thread Priority Protection option.54117 The [ENOTSUP] error condition has been removed since these functions do not have a *protocol*
54118 argument.54119 The **restrict** keyword is added to the *pthread_mutexattr_getprioceiling()* prototype for alignment
54120 with the ISO/IEC 9899:1999 standard.54121 **Issue 7**54122 The *pthread_mutexattr_getprioceiling()* and *pthread_mutexattr_setprioceiling()* functions are moved
54123 from the Threads option to require support of either the Robust Mutex Priority Protection option
54124 or the Non-Robust Mutex Priority Protection option.54125 The [EINVAL] error for an uninitialized mutex attributes object is removed; this condition
54126 results in undefined behavior.

54127 **NAME**

54128 pthread_mutexattr_getprotocol, pthread_mutexattr_setprotocol — get and set the protocol
 54129 attribute of the mutex attributes object (**REALTIME THREADS**)

54130 **SYNOPSIS**

```
54131 MC1      #include <pthread.h>

54132      int pthread_mutexattr_getprotocol(const pthread_mutexattr_t
54133          *restrict attr, int *restrict protocol);
54134      int pthread_mutexattr_setprotocol(pthread_mutexattr_t *attr,
54135          int protocol);
```

54136 **DESCRIPTION**

54137 The *pthread_mutexattr_getprotocol()* and *pthread_mutexattr_setprotocol()* functions, respectively,
 54138 shall get and set the protocol attribute of a mutex attributes object pointed to by *attr* which was
 54139 previously created by the function *pthread_mutexattr_init()*.

54140 The *protocol* attribute defines the protocol to be followed in utilizing mutexes. The value of
 54141 *protocol* may be one of:

```
54142 RPI | TPI  PTHREAD_PRIO_INHERIT
54143 MC1        PTHREAD_PRIO_NONE
54144 RPP | TPP  PTHREAD_PRIO_PROTECT
```

54145 which are defined in the **<pthread.h>** header. The default value of the attribute shall be
 54146 PTHREAD_PRIO_NONE.

54147 When a thread owns a mutex with the PTHREAD_PRIO_NONE *protocol* attribute, its priority
 54148 and scheduling shall not be affected by its mutex ownership.

54149 RPI When a thread is blocking higher priority threads because of owning one or more robust
 54150 mutexes with the PTHREAD_PRIO_INHERIT *protocol* attribute, it shall execute at the higher of
 54151 its priority or the priority of the highest priority thread waiting on any of the robust mutexes
 54152 owned by this thread and initialized with this protocol.

54153 TPI When a thread is blocking higher priority threads because of owning one or more non-robust
 54154 mutexes with the PTHREAD_PRIO_INHERIT *protocol* attribute, it shall execute at the higher of
 54155 its priority or the priority of the highest priority thread waiting on any of the non-robust
 54156 mutexes owned by this thread and initialized with this protocol.

54157 RPP When a thread owns one or more robust mutexes initialized with the
 54158 PTHREAD_PRIO_PROTECT protocol, it shall execute at the higher of its priority or the highest
 54159 of the priority ceilings of all the robust mutexes owned by this thread and initialized with this
 54160 attribute, regardless of whether other threads are blocked on any of these robust mutexes or not.

54161 TPP When a thread owns one or more non-robust mutexes initialized with the
 54162 PTHREAD_PRIO_PROTECT protocol, it shall execute at the higher of its priority or the highest
 54163 of the priority ceilings of all the non-robust mutexes owned by this thread and initialized with
 54164 this attribute, regardless of whether other threads are blocked on any of these non-robust
 54165 mutexes or not.

54166 While a thread is holding a mutex which has been initialized with the
 54167 PTHREAD_PRIO_INHERIT or PTHREAD_PRIO_PROTECT protocol attributes, it shall not be
 54168 subject to being moved to the tail of the scheduling queue at its priority in the event that its
 54169 original priority is changed, such as by a call to *sched_setparam()*. Likewise, when a thread

54170 unlocks a mutex that has been initialized with the PTHREAD_PRIO_INHERIT or
54171 PTHREAD_PRIO_PROTECT protocol attributes, it shall not be subject to being moved to the tail
54172 of the scheduling queue at its priority in the event that its original priority is changed.

54173 If a thread simultaneously owns several mutexes initialized with different protocols, it shall
54174 execute at the highest of the priorities that it would have obtained by each of these protocols.

54175 RPI | TPI When a thread makes a call to *pthread_mutex_lock()*, the mutex was initialized with the protocol
54176 attribute having the value PTHREAD_PRIO_INHERIT, when the calling thread is blocked
54177 because the mutex is owned by another thread, that owner thread shall inherit the priority level
54178 of the calling thread as long as it continues to own the mutex. The implementation shall update
54179 its execution priority to the maximum of its assigned priority and all its inherited priorities.
54180 Furthermore, if this owner thread itself becomes blocked on another mutex with the *protocol*
54181 attribute having the value PTHREAD_PRIO_INHERIT, the same priority inheritance effect shall
54182 be propagated to this other owner thread, in a recursive manner.

54183 The behavior is undefined if the value specified by the *attr* argument to
54184 *pthread_mutexattr_getprotocol()* or *pthread_mutexattr_setprotocol()* does not refer to an initialized
54185 mutex attributes object.

54186 RETURN VALUE

54187 Upon successful completion, the *pthread_mutexattr_getprotocol()* and
54188 *pthread_mutexattr_setprotocol()* functions shall return zero; otherwise, an error number shall be
54189 returned to indicate the error.

54190 ERRORS

54191 The *pthread_mutexattr_setprotocol()* function shall fail if:

54192 [ENOTSUP] The value specified by *protocol* is an unsupported value.

54193 The *pthread_mutexattr_getprotocol()* and *pthread_mutexattr_setprotocol()* functions may fail if:

54194 [EINVAL] The value specified by *protocol* is invalid.

54195 [EPERM] The caller does not have the privilege to perform the operation.

54196 These functions shall not return an error code of [EINTR].

54197 EXAMPLES

54198 None.

54199 APPLICATION USAGE

54200 None.

54201 RATIONALE

54202 If an implementation detects that the value specified by the *attr* argument to
54203 *pthread_mutexattr_getprotocol()* or *pthread_mutexattr_setprotocol()* does not refer to an initialized
54204 mutex attributes object, it is recommended that the function should fail and report an [EINVAL]
54205 error.

54206 FUTURE DIRECTIONS

54207 None.

54208 SEE ALSO

54209 *pthread_cond_destroy()*, *pthread_create()*, *pthread_mutex_destroy()*

54210 XBD <pthread.h>

54211 **CHANGE HISTORY**

54212 First released in Issue 5. Included for alignment with the POSIX Threads Extension.

54213 Marked as part of the Realtime Threads Feature Group.

54214 **Issue 6**

54215 The *pthread_mutexattr_getprotocol()* and *pthread_mutexattr_setprotocol()* functions are marked as
54216 part of the Threads option and either the Thread Priority Protection or Thread Priority
54217 Inheritance options.

54218 The [ENOSYS] error condition has been removed as stubs need not be provided if an
54219 implementation does not support the Thread Priority Protection or Thread Priority Inheritance
54220 options.

54221 The **restrict** keyword is added to the *pthread_mutexattr_getprotocol()* prototype for alignment
54222 with the ISO/IEC 9899:1999 standard.

54223 **Issue 7**

54224 SD5-XSH-ERN-135 is applied, updating the DESCRIPTION to define a default value for the
54225 *protocol* attribute.

54226 SD5-XSH-ERN-188 is applied, updating the DESCRIPTION.

54227 The *pthread_mutexattr_getprotocol()* and *pthread_mutexattr_setprotocol()* functions are moved from
54228 the Threads option to require support of either the Non-Robust Mutex Priority Protection option
54229 or the Non-Robust Mutex Priority Inheritance option or the Robust Mutex Priority Protection
54230 option or the Robust Mutex Priority Inheritance option.

54231 The [EINVAL] error for an uninitialized mutex attributes object is removed; this condition
54232 results in undefined behavior.

NAME

pthread_mutexattr_getpshared, pthread_mutexattr_setpshared — get and set the process-shared attribute

SYNOPSIS

```
TSH #include <pthread.h>

int pthread_mutexattr_getpshared(const pthread_mutexattr_t
    *restrict attr, int *restrict pshared);
int pthread_mutexattr_setpshared(pthread_mutexattr_t *attr,
    int pshared);
```

DESCRIPTION

The *pthread_mutexattr_getpshared()* function shall obtain the value of the *process-shared* attribute from the attributes object referenced by *attr*.

The *pthread_mutexattr_setpshared()* function shall set the *process-shared* attribute in an initialized attributes object referenced by *attr*.

The *process-shared* attribute is set to `PTHREAD_PROCESS_SHARED` to permit a mutex to be operated upon by any thread that has access to the memory where the mutex is allocated, even if the mutex is allocated in memory that is shared by multiple processes. If the *process-shared* attribute is `PTHREAD_PROCESS_PRIVATE`, the mutex shall only be operated upon by threads created within the same process as the thread that initialized the mutex; if threads of differing processes attempt to operate on such a mutex, the behavior is undefined. The default value of the attribute shall be `PTHREAD_PROCESS_PRIVATE`.

The behavior is undefined if the value specified by the *attr* argument to *pthread_mutexattr_getpshared()* or *pthread_mutexattr_setpshared()* does not refer to an initialized mutex attributes object.

RETURN VALUE

Upon successful completion, *pthread_mutexattr_setpshared()* shall return zero; otherwise, an error number shall be returned to indicate the error.

Upon successful completion, *pthread_mutexattr_getpshared()* shall return zero and store the value of the *process-shared* attribute of *attr* into the object referenced by the *pshared* parameter. Otherwise, an error number shall be returned to indicate the error.

ERRORS

The *pthread_mutexattr_setpshared()* function may fail if:

[EINVAL] The new value specified for the attribute is outside the range of legal values for that attribute.

These functions shall not return an error code of [EINTR].

EXAMPLES

None.

APPLICATION USAGE

None.

RATIONALE

If an implementation detects that the value specified by the *attr* argument to *pthread_mutexattr_getpshared()* or *pthread_mutexattr_setpshared()* does not refer to an initialized mutex attributes object, it is recommended that the function should fail and report an [EINVAL] error.

54277 **FUTURE DIRECTIONS**

54278 None.

54279 **SEE ALSO**54280 *pthread_cond_destroy()*, *pthread_create()*, *pthread_mutex_destroy()*, *pthread_mutexattr_destroy()*

54281 XBD <pthread.h>

54282 **CHANGE HISTORY**

54283 First released in Issue 5. Included for alignment with the POSIX Threads Extension.

54284 **Issue 6**54285 The *pthread_mutexattr_getpshared()* and *pthread_mutexattr_setpshared()* functions are marked as
54286 part of the Threads and Thread Process-Shared Synchronization options.54287 The **restrict** keyword is added to the *pthread_mutexattr_getpshared()* prototype for alignment
54288 with the ISO/IEC 9899:1999 standard.54289 **Issue 7**54290 The *pthread_mutexattr_getpshared()* and *pthread_mutexattr_setpshared()* functions are moved from
54291 the Threads option.54292 The [EINVAL] error for an uninitialized mutex attributes object is removed; this condition
54293 results in undefined behavior.

NAME

pthread_mutexattr_getrobust, pthread_mutexattr_setrobust — get and set the mutex robust attribute

SYNOPSIS

```
#include <pthread.h>

int pthread_mutexattr_getrobust(const pthread_mutexattr_t *restrict
    attr, int *restrict robust);
int pthread_mutexattr_setrobust(pthread_mutexattr_t *attr,
    int robust);
```

DESCRIPTION

The *pthread_mutexattr_getrobust()* and *pthread_mutexattr_setrobust()* functions, respectively, shall get and set the mutex *robust* attribute. This attribute is set in the *robust* parameter. Valid values for *robust* include:

PTHREAD_MUTEX_STALLED

No special actions are taken if the owner of the mutex is terminated while holding the mutex lock. This can lead to deadlocks if no other thread can unlock the mutex.

This is the default value.

PTHREAD_MUTEX_ROBUST

If the process containing the owning thread of a robust mutex terminates while holding the mutex lock, the next thread that acquires the mutex shall be notified about the termination by the return value [EOWNERDEAD] from the locking function. If the owning thread of a robust mutex terminates while holding the mutex lock, the next thread that acquires the mutex may be notified about the termination by the return value [EOWNERDEAD]. The notified thread can then attempt to mark the state protected by the mutex as consistent again by a call to *pthread_mutex_consistent()*. After a subsequent successful call to *pthread_mutex_unlock()*, the mutex lock shall be released and can be used normally by other threads. If the mutex is unlocked without a call to *pthread_mutex_consistent()*, it shall be in a permanently unusable state and all attempts to lock the mutex shall fail with the error [ENOTRECOVERABLE]. The only permissible operation on such a mutex is *pthread_mutex_destroy()*.

The behavior is undefined if the value specified by the *attr* argument to *pthread_mutexattr_getrobust()* or *pthread_mutexattr_setrobust()* does not refer to an initialized mutex attributes object.

RETURN VALUE

Upon successful completion, the *pthread_mutexattr_getrobust()* function shall return zero and store the value of the *robust* attribute of *attr* into the object referenced by the *robust* parameter. Otherwise, an error value shall be returned to indicate the error. If successful, the *pthread_mutexattr_setrobust()* function shall return zero; otherwise, an error number shall be returned to indicate the error.

ERRORS

The *pthread_mutexattr_setrobust()* function shall fail if:

[EINVAL] The value of *robust* is invalid.

These functions shall not return an error code of [EINTR].

EXAMPLES

None.

APPLICATION USAGE

The actions required to make the state protected by the mutex consistent again are solely dependent on the application. If it is not possible to make the state of a mutex consistent, robust mutexes can be used to notify this situation by calling *pthread_mutex_unlock()* without a prior call to *pthread_mutex_consistent()*.

If the state is declared inconsistent by calling *pthread_mutex_unlock()* without a prior call to *pthread_mutex_consistent()*, a possible approach could be to destroy the mutex and then reinitialize it. However, it should be noted that this is possible only in certain situations where the state protected by the mutex has to be reinitialized and coordination achieved with other threads blocked on the mutex, because otherwise a call to a locking function with a reference to a mutex object invalidated by a call to *pthread_mutex_destroy()* results in undefined behavior.

RATIONALE

If an implementation detects that the value specified by the *attr* argument to *pthread_mutexattr_getrobust()* or *pthread_mutexattr_setrobust()* does not refer to an initialized mutex attributes object, it is recommended that the function should fail and report an [EINVAL] error.

FUTURE DIRECTIONS

None.

SEE ALSO

pthread_mutex_consistent(), *pthread_mutex_destroy()*, *pthread_mutex_lock()*

XBD <pthread.h>

CHANGE HISTORY

First released in Issue 7.

NAME

pthread_mutexattr_gettype, pthread_mutexattr_settype — get and set the mutex type attribute

SYNOPSIS

```
#include <pthread.h>

int pthread_mutexattr_gettype(const pthread_mutexattr_t *restrict attr,
    int *restrict type);
int pthread_mutexattr_settype(pthread_mutexattr_t *attr, int type);
```

DESCRIPTION

The *pthread_mutexattr_gettype()* and *pthread_mutexattr_settype()* functions, respectively, shall get and set the mutex *type* attribute. This attribute is set in the *type* parameter to these functions. The default value of the *type* attribute is PTHREAD_MUTEX_DEFAULT.

The type of mutex is contained in the *type* attribute of the mutex attributes. Valid mutex types include:

PTHREAD_MUTEX_NORMAL	
PTHREAD_MUTEX_ERRORCHECK	
PTHREAD_MUTEX_RECURSIVE	
PTHREAD_MUTEX_DEFAULT	

The mutex type affects the behavior of calls which lock and unlock the mutex. See *pthread_mutex_lock()* for details. An implementation may map PTHREAD_MUTEX_DEFAULT to one of the other mutex types.

The behavior is undefined if the value specified by the *attr* argument to *pthread_mutexattr_gettype()* or *pthread_mutexattr_settype()* does not refer to an initialized mutex attributes object.

RETURN VALUE

Upon successful completion, the *pthread_mutexattr_gettype()* function shall return zero and store the value of the *type* attribute of *attr* into the object referenced by the *type* parameter. Otherwise, an error shall be returned to indicate the error.

If successful, the *pthread_mutexattr_settype()* function shall return zero; otherwise, an error number shall be returned to indicate the error.

ERRORS

The *pthread_mutexattr_settype()* function shall fail if:

[EINVAL] The value *type* is invalid.

These functions shall not return an error code of [EINTR].

EXAMPLES

None.

APPLICATION USAGE

It is advised that an application should not use a PTHREAD_MUTEX_RECURSIVE mutex with condition variables because the implicit unlock performed for a *pthread_cond_timedwait()* or *pthread_cond_wait()* may not actually release the mutex (if it had been locked multiple times). If this happens, no other thread can satisfy the condition of the predicate.

RATIONALE

If an implementation detects that the value specified by the *attr* argument to *pthread_mutexattr_gettype()* or *pthread_mutexattr_settype()* does not refer to an initialized mutex attributes object, it is recommended that the function should fail and report an [EINVAL] error.

54406 **FUTURE DIRECTIONS**

54407 None.

54408 **SEE ALSO**54409 *pthread_cond_timedwait()*, *pthread_mutex_lock()*

54410 XBD <pthread.h>

54411 **CHANGE HISTORY**

54412 First released in Issue 5.

54413 **Issue 6**

54414 The Open Group Corrigendum U033/3 is applied. The SYNOPSIS for
54415 *pthread_mutexattr_gettype()* is updated so that the first argument is of type **const**
54416 **pthread_mutexattr_t** *.

54417 The **restrict** keyword is added to the *pthread_mutexattr_gettype()* prototype for alignment with
54418 the ISO/IEC 9899:1999 standard.

54419 **Issue 7**

54420 The *pthread_mutexattr_gettype()* and *pthread_mutexattr_settype()* functions are moved from the
54421 XSI option to the Base.

54422 The [EINVAL] error for an uninitialized mutex attributes object is removed; this condition
54423 results in undefined behavior.

54424 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0464 [121] is applied.

+

54425 NAME

54426 pthread_mutexattr_init — initialize the mutex attributes object

54427 SYNOPSIS

54428 #include <pthread.h>

54429 int pthread_mutexattr_init(pthread_mutexattr_t *attr);

54430 DESCRIPTION

54431 Refer to *pthread_mutexattr_destroy()*.

54432 **NAME**

54433 pthread_mutexattr_setprioceiling — set the prioceiling attribute of the mutex attributes object
54434 (**REALTIME THREADS**)

54435 **SYNOPSIS**

```
54436 RPP|TPP #include <pthread.h>  
54437 int pthread_mutexattr_setprioceiling(pthread_mutexattr_t *attr,  
54438 int prioceiling);
```

54439 **DESCRIPTION**

54440 Refer to *pthread_mutexattr_getprioceiling()*.

54441 NAME

54442 pthread_mutexattr_setprotocol — set the protocol attribute of the mutex attributes object
54443 (REALTIME THREADS)

54444 SYNOPSIS

```
54445 MC1 #include <pthread.h>  
54446 int pthread_mutexattr_setprotocol(pthread_mutexattr_t *attr,  
54447 int protocol);
```

54448 DESCRIPTION

54449 Refer to *pthread_mutexattr_getprotocol()*.

54450 **NAME**

54451 pthread_mutexattr_setpshared — set the process-shared attribute

54452 **SYNOPSIS**

```
54453 TSH      #include <pthread.h>
54454          int pthread_mutexattr_setpshared(pthread_mutexattr_t *attr,
54455          int pshared);
```

54456 **DESCRIPTION**54457 Refer to *pthread_mutexattr_getpshared()*.

54458 NAME

54459 pthread_mutexattr_setrobust — get and set the mutex robust attribute

54460 SYNOPSIS

```
54461     #include <pthread.h>
54462     int pthread_mutexattr_setrobust(pthread_mutexattr_t *attr,
54463                                     int robust);
```

54464 DESCRIPTION

54465 Refer to *pthread_mutexattr_getrobust()*.

54466 **NAME**

54467 pthread_mutexattr_settype — set the mutex type attribute

54468 **SYNOPSIS**

54469 #include <pthread.h>

54470 int pthread_mutexattr_settype(pthread_mutexattr_t *attr, int type);

54471 **DESCRIPTION**54472 Refer to *pthread_mutexattr_gettype()*.

NAME

pthread_once — dynamic package initialization

SYNOPSIS

```
#include <pthread.h>

int pthread_once(pthread_once_t *once_control,
                 void (*init_routine)(void));

pthread_once_t once_control = PTHREAD_ONCE_INIT;
```

DESCRIPTION

The first call to *pthread_once()* by any thread in a process, with a given *once_control*, shall call the *init_routine* with no arguments. Subsequent calls of *pthread_once()* with the same *once_control* shall not call the *init_routine*. On return from *pthread_once()*, *init_routine* shall have completed. The *once_control* parameter shall determine whether the associated initialization routine has been called.

The *pthread_once()* function is not a cancellation point. However, if *init_routine* is a cancellation point and is canceled, the effect on *once_control* shall be as if *pthread_once()* was never called.

The constant PTHREAD_ONCE_INIT is defined in the **<pthread.h>** header.

The behavior of *pthread_once()* is undefined if *once_control* has automatic storage duration or is not initialized by PTHREAD_ONCE_INIT.

RETURN VALUE

Upon successful completion, *pthread_once()* shall return zero; otherwise, an error number shall be returned to indicate the error.

ERRORS

The *pthread_once()* function shall not return an error code of [EINTR].

EXAMPLES

None.

APPLICATION USAGE

None.

RATIONALE

Some C libraries are designed for dynamic initialization. That is, the global initialization for the library is performed when the first procedure in the library is called. In a single-threaded program, this is normally implemented using a static variable whose value is checked on entry to a routine, as follows:

```
static int random_is_initialized = 0;
extern int initialize_random();

int random_function()
{
    if (random_is_initialized == 0) {
        initialize_random();
        random_is_initialized = 1;
    }
    ... /* Operations performed after initialization. */
}
```

To keep the same structure in a multi-threaded program, a new primitive is needed. Otherwise, library initialization has to be accomplished by an explicit call to a library-exported initialization function prior to any use of the library.

For dynamic library initialization in a multi-threaded process, a simple initialization flag is not sufficient; the flag needs to be protected against modification by multiple threads simultaneously calling into the library. Protecting the flag requires the use of a mutex; however, mutexes have to be initialized before they are used. Ensuring that the mutex is only initialized once requires a recursive solution to this problem.

The use of *pthread_once()* not only supplies an implementation-guaranteed means of dynamic initialization, it provides an aid to the reliable construction of multi-threaded and realtime systems. The preceding example then becomes:

```
#include <pthread.h>
static pthread_once_t random_is_initialized = PTHREAD_ONCE_INIT;
extern int initialize_random();

int random_function()
{
    (void) pthread_once(&random_is_initialized, initialize_random);
    ... /* Operations performed after initialization. */
}
```

Note that a **pthread_once_t** cannot be an array because some compilers do not accept the construct **&<array_name>**.

If an implementation detects that the value specified by the *once_control* argument to *pthread_once()* does not refer to a **pthread_once_t** object initialized by **PTHREAD_ONCE_INIT**, it is recommended that the function should fail and report an [EINVAL] error.

FUTURE DIRECTIONS

None.

SEE ALSO

XBD [<pthread.h>](#)

CHANGE HISTORY

First released in Issue 5. Included for alignment with the POSIX Threads Extension.

Issue 6

The *pthread_once()* function is marked as part of the Threads option.

The [EINVAL] error is added as a “may fail” case for if either argument is invalid.

Issue 7

The *pthread_once()* function is moved from the Threads option to the Base.

The [EINVAL] error for an uninitialized **pthread_once_t** object is removed; this condition results in undefined behavior.

NAME

pthread_rwlock_destroy, pthread_rwlock_init — destroy and initialize a read-write lock object

SYNOPSIS

```
#include <pthread.h>

int pthread_rwlock_destroy(pthread_rwlock_t *rwlock);
int pthread_rwlock_init(pthread_rwlock_t *restrict rwlock,
    const pthread_rwlockattr_t *restrict attr);
pthread_rwlock_t rwlock = PTHREAD_RWLOCK_INITIALIZER;
```

DESCRIPTION

The *pthread_rwlock_destroy()* function shall destroy the read-write lock object referenced by *rwlock* and release any resources used by the lock. The effect of subsequent use of the lock is undefined until the lock is reinitialized by another call to *pthread_rwlock_init()*. An implementation may cause *pthread_rwlock_destroy()* to set the object referenced by *rwlock* to an invalid value. Results are undefined if *pthread_rwlock_destroy()* is called when any thread holds *rwlock*. Attempting to destroy an uninitialized read-write lock results in undefined behavior.

The *pthread_rwlock_init()* function shall allocate any resources required to use the read-write lock referenced by *rwlock* and initializes the lock to an unlocked state with attributes referenced by *attr*. If *attr* is NULL, the default read-write lock attributes shall be used; the effect is the same as passing the address of a default read-write lock attributes object. Once initialized, the lock can be used any number of times without being reinitialized. Results are undefined if *pthread_rwlock_init()* is called specifying an already initialized read-write lock. Results are undefined if a read-write lock is used without first being initialized.

If the *pthread_rwlock_init()* function fails, *rwlock* shall not be initialized and the contents of *rwlock* are undefined.

Only the object referenced by *rwlock* may be used for performing synchronization. The result of referring to copies of that object in calls to *pthread_rwlock_destroy()*, *pthread_rwlock_rdlock()*, *pthread_rwlock_timedrdlock()*, *pthread_rwlock_timedwrlock()*, *pthread_rwlock_tryrdlock()*, *pthread_rwlock_trywrlock()*, *pthread_rwlock_unlock()*, or *pthread_rwlock_wrlock()* is undefined.

In cases where default read-write lock attributes are appropriate, the macro PTHREAD_RWLOCK_INITIALIZER can be used to initialize read-write locks. The effect shall be equivalent to dynamic initialization by a call to *pthread_rwlock_init()* with the *attr* parameter specified as NULL, except that no error checks are performed.

The behavior is undefined if the value specified by the *attr* argument to *pthread_rwlock_init()* does not refer to an initialized read-write lock attributes object.

RETURN VALUE

If successful, the *pthread_rwlock_destroy()* and *pthread_rwlock_init()* functions shall return zero; otherwise, an error number shall be returned to indicate the error.

ERRORS

The *pthread_rwlock_init()* function shall fail if:

- [EAGAIN] The system lacked the necessary resources (other than memory) to initialize another read-write lock.
- [ENOMEM] Insufficient memory exists to initialize the read-write lock.
- [EPERM] The caller does not have the privilege to perform the operation.

These functions shall not return an error code of [EINTR].

EXAMPLES

None.

APPLICATION USAGE

Applications using these and related read-write lock functions may be subject to priority inversion, as discussed in XBD [Section 3.287](#) (on page 79).

RATIONALE

If an implementation detects that the value specified by the *rwlock* argument to *pthread_rwlock_destroy()* does not refer to an initialized read-write lock object, it is recommended that the function should fail and report an [EINVAL] error.

If an implementation detects that the value specified by the *attr* argument to *pthread_rwlock_init()* does not refer to an initialized read-write lock attributes object, it is recommended that the function should fail and report an [EINVAL] error.

If an implementation detects that the value specified by the *rwlock* argument to *pthread_rwlock_destroy()* or *pthread_rwlock_init()* refers to a locked read-write lock object, or detects that the value specified by the *rwlock* argument to *pthread_rwlock_init()* refers to an already initialized read-write lock object, it is recommended that the function should fail and report an [EBUSY] error.

FUTURE DIRECTIONS

None.

SEE ALSO

[*pthread_rwlock_rdlock\(\)*](#), [*pthread_rwlock_timedrdlock\(\)*](#), [*pthread_rwlock_timedwrlock\(\)*](#),
[*pthread_rwlock_trywrlock\(\)*](#), [*pthread_rwlock_unlock\(\)*](#)

XBD [Section 3.287](#) (on page 79), [**<pthread.h>**](#)

CHANGE HISTORY

First released in Issue 5.

Issue 6

The following changes are made for alignment with IEEE Std 1003.1j-2000:

- The margin code in the SYNOPSIS is changed to THR to indicate that the functionality is now part of the Threads option (previously it was part of the Read-Write Locks option in IEEE Std 1003.1j-2000 and also part of the XSI extension). The initializer macro is also deleted from the SYNOPSIS.
- The DESCRIPTION is updated as follows:
 - It explicitly notes allocation of resources upon initialization of a read-write lock object.
 - A paragraph is added specifying that copies of read-write lock objects may not be used.
- An [EINVAL] error is added to the ERRORS section for *pthread_rwlock_init()*, indicating that the *rwlock* value is invalid.
- The SEE ALSO section is updated.

The **restrict** keyword is added to the *pthread_rwlock_init()* prototype for alignment with the ISO/IEC 9899:1999 standard.

IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/45 is applied, adding APPLICATION USAGE relating to priority inversion.

Issue 7

Austin Group Interpretation 1003.1-2001 #048 is applied, adding the PTHREAD_RWLOCK_INITIALIZER macro.

The *pthread_rwlock_destroy()* and *pthread_rwlock_init()* functions are moved from the Threads option to the Base.

The [EINVAL] error for an uninitialized read-write lock object or read-write lock attributes object is removed; this condition results in undefined behavior.

The [EBUSY] error for a locked read-write lock object or an already initialized read-write lock object is removed; this condition results in undefined behavior.

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0465 [70] is applied.

+

NAME

pthread_rwlock_rdlock, pthread_rwlock_tryrdlock — lock a read-write lock object for reading

SYNOPSIS

```
#include <pthread.h>

int pthread_rwlock_rdlock(pthread_rwlock_t *rwlock);
int pthread_rwlock_tryrdlock(pthread_rwlock_t *rwlock);
```

DESCRIPTION

The *pthread_rwlock_rdlock()* function shall apply a read lock to the read-write lock referenced by *rwlock*. The calling thread acquires the read lock if a writer does not hold the lock and there are no writers blocked on the lock.

TPS

If the Thread Execution Scheduling option is supported, and the threads involved in the lock are executing with the scheduling policies SCHED_FIFO or SCHED_RR, the calling thread shall not acquire the lock if a writer holds the lock or if writers of higher or equal priority are blocked on the lock; otherwise, the calling thread shall acquire the lock.

TPS TSP

If the Thread Execution Scheduling option is supported, and the threads involved in the lock are executing with the SCHED_SPORADIC scheduling policy, the calling thread shall not acquire the lock if a writer holds the lock or if writers of higher or equal priority are blocked on the lock; otherwise, the calling thread shall acquire the lock.

If the Thread Execution Scheduling option is not supported, it is implementation-defined whether the calling thread acquires the lock when a writer does not hold the lock and there are writers blocked on the lock. If a writer holds the lock, the calling thread shall not acquire the read lock. If the read lock is not acquired, the calling thread shall block until it can acquire the lock. The calling thread may deadlock if at the time the call is made it holds a write lock.

A thread may hold multiple concurrent read locks on *rwlock* (that is, successfully call the *pthread_rwlock_rdlock()* function *n* times). If so, the application shall ensure that the thread performs matching unlocks (that is, it calls the *pthread_rwlock_unlock()* function *n* times).

The maximum number of simultaneous read locks that an implementation guarantees can be applied to a read-write lock shall be implementation-defined. The *pthread_rwlock_rdlock()* function may fail if this maximum would be exceeded.

The *pthread_rwlock_tryrdlock()* function shall apply a read lock as in the *pthread_rwlock_rdlock()* function, with the exception that the function shall fail if the equivalent *pthread_rwlock_rdlock()* call would have blocked the calling thread. In no case shall the *pthread_rwlock_tryrdlock()* function ever block; it always either acquires the lock or fails and returns immediately.

Results are undefined if any of these functions are called with an uninitialized read-write lock.

If a signal is delivered to a thread waiting for a read-write lock for reading, upon return from the signal handler the thread resumes waiting for the read-write lock for reading as if it was not interrupted.

RETURN VALUE

If successful, the *pthread_rwlock_rdlock()* function shall return zero; otherwise, an error number shall be returned to indicate the error.

The *pthread_rwlock_tryrdlock()* function shall return zero if the lock for reading on the read-write lock object referenced by *rwlock* is acquired. Otherwise, an error number shall be returned to indicate the error.

ERRORS

The *pthread_rwlock_tryrdlock()* function shall fail if:

[EBUSY] The read-write lock could not be acquired for reading because a writer holds the lock or a writer with the appropriate priority was blocked on it.

The *pthread_rwlock_rdlock()* and *pthread_rwlock_tryrdlock()* functions may fail if:

[EAGAIN] The read lock could not be acquired because the maximum number of read locks for *rwlock* has been exceeded.

The *pthread_rwlock_rdlock()* function may fail if:

[EDEADLK] A deadlock condition was detected or the current thread already owns the read-write lock for writing.

These functions shall not return an error code of [EINTR].

EXAMPLES

None.

APPLICATION USAGE

Applications using these functions may be subject to priority inversion, as discussed in XBD [Section 3.287](#) (on page 79).

RATIONALE

If an implementation detects that the value specified by the *rwlock* argument to *pthread_rwlock_rdlock()* or *pthread_rwlock_tryrdlock()* does not refer to an initialized read-write lock object, it is recommended that the function should fail and report an [EINVAL] error.

FUTURE DIRECTIONS

None.

SEE ALSO

[*pthread_rwlock_destroy\(\)*](#), [*pthread_rwlock_timedrdlock\(\)*](#), [*pthread_rwlock_timedwrlock\(\)*](#), [*pthread_rwlock_trywrlock\(\)*](#), [*pthread_rwlock_unlock\(\)*](#)

XBD [Section 3.287](#) (on page 79), [Section 4.11](#) (on page 110), [**<pthread.h>**](#)

CHANGE HISTORY

First released in Issue 5.

Issue 6

The following changes are made for alignment with IEEE Std 1003.1j-2000:

- The margin code in the SYNOPSIS is changed to THR to indicate that the functionality is now part of the Threads option (previously it was part of the Read-Write Locks option in IEEE Std 1003.1j-2000 and also part of the XSI extension).
- The DESCRIPTION is updated as follows:
 - Conditions under which writers have precedence over readers are specified.
 - Failure of *pthread_rwlock_tryrdlock()* is clarified.
 - A paragraph on the maximum number of read locks is added.
- In the ERRORS sections, [EBUSY] is modified to take into account write priority, and [EDEADLK] is deleted as a *pthread_rwlock_tryrdlock()* error.

- 54731 • The SEE ALSO section is updated.
- 54732 IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/101 is applied, updating the ERRORS
- 54733 section so that the [EDEADLK] error includes detection of a deadlock condition.
- 54734 **Issue 7**
- 54735 The *pthread_rwlock_rdlock()* and *pthread_rwlock_tryrdlock()* functions are moved from the Threads
- 54736 option to the Base.
- 54737 The [EINVAL] error for an uninitialized read-write lock object is removed; this condition results
- 54738 in undefined behavior.

NAME

pthread_rwlock_timedrdlock — lock a read-write lock for reading

SYNOPSIS

```
#include <pthread.h>
#include <time.h>
```

```
int pthread_rwlock_timedrdlock(pthread_rwlock_t *restrict rwlock,
    const struct timespec *restrict abstime);
```

DESCRIPTION

The *pthread_rwlock_timedrdlock()* function shall apply a read lock to the read-write lock referenced by *rwlock* as in the *pthread_rwlock_rdlock()* function. However, if the lock cannot be acquired without waiting for other threads to unlock the lock, this wait shall be terminated when the specified timeout expires. The timeout shall expire when the absolute time specified by *abstime* passes, as measured by the clock on which timeouts are based (that is, when the value of that clock equals or exceeds *abstime*), or if the absolute time specified by *abstime* has already been passed at the time of the call.

The timeout shall be based on the CLOCK_REALTIME clock. The resolution of the timeout shall be the resolution of the CLOCK_REALTIME clock. The **timespec** data type is defined in the **<time.h>** header. Under no circumstances shall the function fail with a timeout if the lock can be acquired immediately. The validity of the *abstime* parameter need not be checked if the lock can be immediately acquired.

If a signal that causes a signal handler to be executed is delivered to a thread blocked on a read-write lock via a call to *pthread_rwlock_timedrdlock()*, upon return from the signal handler the thread shall resume waiting for the lock as if it was not interrupted.

The calling thread may deadlock if at the time the call is made it holds a write lock on *rwlock*. The results are undefined if this function is called with an uninitialized read-write lock.

RETURN VALUE

The *pthread_rwlock_timedrdlock()* function shall return zero if the lock for reading on the read-write lock object referenced by *rwlock* is acquired. Otherwise, an error number shall be returned to indicate the error.

ERRORS

The *pthread_rwlock_timedrdlock()* function shall fail if:

[ETIMEDOUT] The lock could not be acquired before the specified timeout expired.

The *pthread_rwlock_timedrdlock()* function may fail if:

[EAGAIN] The read lock could not be acquired because the maximum number of read locks for lock would be exceeded.

[EDEADLK] A deadlock condition was detected or the calling thread already holds a write lock on *rwlock*.

[EINVAL] The *abstime* nanosecond value is less than zero or greater than or equal to 1 000 million.

This function shall not return an error code of [EINTR].

EXAMPLES

None.

APPLICATION USAGE

Applications using this function may be subject to priority inversion, as discussed in XBD Section 3.287 (on page 79).

RATIONALE

If an implementation detects that the value specified by the *rwlock* argument to *pthread_rwlock_timedrdlock()* does not refer to an initialized read-write lock object, it is recommended that the function should fail and report an [EINVAL] error.

FUTURE DIRECTIONS

None.

SEE ALSO

pthread_rwlock_destroy(), *pthread_rwlock_rdlock()*, *pthread_rwlock_timedwrlock()*, *pthread_rwlock_trywrlock()*, *pthread_rwlock_unlock()*

XBD Section 3.287 (on page 79), Section 4.11 (on page 110), <pthread.h>, <time.h>

CHANGE HISTORY

First released in Issue 6. Derived from IEEE Std 1003.1j-2000.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/102 is applied, updating the ERRORS section so that the [EDEADLK] error includes detection of a deadlock condition.

Issue 7

The *pthread_rwlock_timedrdlock()* function is moved from the Timeouts option to the Base.

The [EINVAL] error for an uninitialized read-write lock object is removed; this condition results in undefined behavior.

NAME

pthread_rwlock_timedwrlock — lock a read-write lock for writing

SYNOPSIS

```
#include <pthread.h>
#include <time.h>
```

```
int pthread_rwlock_timedwrlock(pthread_rwlock_t *restrict rwlock,
    const struct timespec *restrict abstime);
```

DESCRIPTION

The *pthread_rwlock_timedwrlock()* function shall apply a write lock to the read-write lock referenced by *rwlock* as in the *pthread_rwlock_wrlock()* function. However, if the lock cannot be acquired without waiting for other threads to unlock the lock, this wait shall be terminated when the specified timeout expires. The timeout shall expire when the absolute time specified by *abstime* passes, as measured by the clock on which timeouts are based (that is, when the value of that clock equals or exceeds *abstime*), or if the absolute time specified by *abstime* has already been passed at the time of the call.

The timeout shall be based on the CLOCK_REALTIME clock. The resolution of the timeout shall be the resolution of the CLOCK_REALTIME clock. The **timespec** data type is defined in the **<time.h>** header. Under no circumstances shall the function fail with a timeout if the lock can be acquired immediately. The validity of the *abstime* parameter need not be checked if the lock can be immediately acquired.

If a signal that causes a signal handler to be executed is delivered to a thread blocked on a read-write lock via a call to *pthread_rwlock_timedwrlock()*, upon return from the signal handler the thread shall resume waiting for the lock as if it was not interrupted.

The calling thread may deadlock if at the time the call is made it holds the read-write lock. The results are undefined if this function is called with an uninitialized read-write lock.

RETURN VALUE

The *pthread_rwlock_timedwrlock()* function shall return zero if the lock for writing on the read-write lock object referenced by *rwlock* is acquired. Otherwise, an error number shall be returned to indicate the error.

ERRORS

The *pthread_rwlock_timedwrlock()* function shall fail if:

[ETIMEDOUT] The lock could not be acquired before the specified timeout expired.

The *pthread_rwlock_timedwrlock()* function may fail if:

[EDEADLK] A deadlock condition was detected or the calling thread already holds the *rwlock*.

[EINVAL] The *abstime* nanosecond value is less than zero or greater than or equal to 1 000 million.

This function shall not return an error code of [EINTR].

EXAMPLES

None.

APPLICATION USAGE

Applications using this function may be subject to priority inversion, as discussed in XBD [Section 3.287](#) (on page 79).

RATIONALE

If an implementation detects that the value specified by the *rwlock* argument to *pthread_rwlock_timedwrlock()* does not refer to an initialized read-write lock object, it is recommended that the function should fail and report an [EINVAL] error.

FUTURE DIRECTIONS

None.

SEE ALSO

[*pthread_rwlock_destroy\(\)*](#), [*pthread_rwlock_rdlock\(\)*](#), [*pthread_rwlock_timedrdlock\(\)*](#), [*pthread_rwlock_trywrlock\(\)*](#), [*pthread_rwlock_unlock\(\)*](#)

XBD [Section 3.287](#) (on page 79), [Section 4.11](#) (on page 110), [**<pthread.h>**](#), [**<time.h>**](#)

CHANGE HISTORY

First released in Issue 6. Derived from IEEE Std 1003.1j-2000.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/103 is applied, updating the ERRORS section so that the [EDEADLK] error includes detection of a deadlock condition.

Issue 7

The *pthread_rwlock_timedwrlock()* function is moved from the Timeouts option to the Base.

The [EINVAL] error for an uninitialized read-write lock object is removed; this condition results in undefined behavior.

54863 NAME

54864 pthread_rwlock_tryrdlock — lock a read-write lock object for reading

54865 SYNOPSIS

54866 #include <pthread.h>

54867 int pthread_rwlock_tryrdlock(pthread_rwlock_t *rwlock);

54868 DESCRIPTION

54869 Refer to *pthread_rwlock_rdlock()*.

NAME

pthread_rwlock_trywrlock, pthread_rwlock_wrlock — lock a read-write lock object for writing

SYNOPSIS

```
#include <pthread.h>

int pthread_rwlock_trywrlock(pthread_rwlock_t *rwlock);
int pthread_rwlock_wrlock(pthread_rwlock_t *rwlock);
```

DESCRIPTION

The *pthread_rwlock_trywrlock()* function shall apply a write lock like the *pthread_rwlock_wrlock()* function, with the exception that the function shall fail if any thread currently holds *rwlock* (for reading or writing).

The *pthread_rwlock_wrlock()* function shall apply a write lock to the read-write lock referenced by *rwlock*. The calling thread acquires the write lock if no other thread (reader or writer) holds the read-write lock *rwlock*. Otherwise, the thread shall block until it can acquire the lock. The calling thread may deadlock if at the time the call is made it holds the read-write lock (whether a read or write lock).

Implementations may favor writers over readers to avoid writer starvation.

Results are undefined if any of these functions are called with an uninitialized read-write lock.

If a signal is delivered to a thread waiting for a read-write lock for writing, upon return from the signal handler the thread resumes waiting for the read-write lock for writing as if it was not interrupted.

RETURN VALUE

The *pthread_rwlock_trywrlock()* function shall return zero if the lock for writing on the read-write lock object referenced by *rwlock* is acquired. Otherwise, an error number shall be returned to indicate the error.

If successful, the *pthread_rwlock_wrlock()* function shall return zero; otherwise, an error number shall be returned to indicate the error.

ERRORS

The *pthread_rwlock_trywrlock()* function shall fail if:

[EBUSY] The read-write lock could not be acquired for writing because it was already locked for reading or writing.

The *pthread_rwlock_wrlock()* function may fail if:

[EDEADLK] A deadlock condition was detected or the current thread already owns the read-write lock for writing or reading.

These functions shall not return an error code of [EINTR].

EXAMPLES

None.

APPLICATION USAGE

Applications using these functions may be subject to priority inversion, as discussed in XBD [Section 3.287](#) (on page 79).

RATIONALE

If an implementation detects that the value specified by the *rwlock* argument to *pthread_rwlock_trywrlock()* or *pthread_rwlock_wrlock()* does not refer to an initialized read-write lock object, it is recommended that the function should fail and report an [EINVAL] error.

FUTURE DIRECTIONS

None.

SEE ALSO

pthread_rwlock_destroy(), *pthread_rwlock_rdlock()*, *pthread_rwlock_timedrdlock()*,
pthread_rwlock_timedwrlock(), *pthread_rwlock_unlock()*

XBD [Section 3.287](#) (on page 79), [Section 4.11](#) (on page 110), [<pthread.h>](#)

CHANGE HISTORY

First released in Issue 5.

Issue 6

The following changes are made for alignment with IEEE Std 1003.1j-2000:

- The margin code in the SYNOPSIS is changed to THR to indicate that the functionality is now part of the Threads option (previously it was part of the Read-Write Locks option in IEEE Std 1003.1j-2000 and also part of the XSI extension).
- The [EDEADLK] error is deleted as a *pthread_rwlock_trywrlock()* error.
- The SEE ALSO section is updated.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/104 is applied, updating the ERRORS section so that the [EDEADLK] error includes detection of a deadlock condition.

Issue 7

The *pthread_rwlock_trywrlock()* and *pthread_rwlock_wrlock()* functions are moved from the Threads option to the Base.

The [EINVAL] error for an uninitialized read-write lock object is removed; this condition results in undefined behavior.

NAME

pthread_rwlock_unlock — unlock a read-write lock object

SYNOPSIS

```
#include <pthread.h>
```

```
int pthread_rwlock_unlock(pthread_rwlock_t *rwlock);
```

DESCRIPTION

The *pthread_rwlock_unlock()* function shall release a lock held on the read-write lock object referenced by *rwlock*. Results are undefined if the read-write lock *rwlock* is not held by the calling thread.

If this function is called to release a read lock from the read-write lock object and there are other read locks currently held on this read-write lock object, the read-write lock object remains in the read locked state. If this function releases the last read lock for this read-write lock object, the read-write lock object shall be put in the unlocked state with no owners.

If this function is called to release a write lock for this read-write lock object, the read-write lock object shall be put in the unlocked state.

TPS If there are threads blocked on the lock when it becomes available, the scheduling policy shall determine which thread(s) shall acquire the lock. If the Thread Execution Scheduling option is supported, when threads executing with the scheduling policies SCHED_FIFO, SCHED_RR, or SCHED_SPORADIC are waiting on the lock, they shall acquire the lock in priority order when the lock becomes available. For equal priority threads, write locks shall take precedence over read locks. If the Thread Execution Scheduling option is not supported, it is implementation-defined whether write locks take precedence over read locks.

Results are undefined if this function is called with an uninitialized read-write lock.

RETURN VALUE

If successful, the *pthread_rwlock_unlock()* function shall return zero; otherwise, an error number shall be returned to indicate the error.

ERRORS

The *pthread_rwlock_unlock()* function shall not return an error code of [EINTR].

EXAMPLES

None.

APPLICATION USAGE

None.

RATIONALE

If an implementation detects that the value specified by the *rwlock* argument to *pthread_rwlock_unlock()* does not refer to an initialized read-write lock object, it is recommended that the function should fail and report an [EINVAL] error.

If an implementation detects that the value specified by the *rwlock* argument to *pthread_rwlock_unlock()* refers to a read-write lock object for which the current thread does not hold a lock, it is recommended that the function should fail and report an [EPERM] error.

FUTURE DIRECTIONS

None.

SEE ALSO

pthread_rwlock_destroy(), *pthread_rwlock_rdlock()*, *pthread_rwlock_timedrdlock()*,
pthread_rwlock_timedwrlock(), *pthread_rwlock_trywrlock()*

XBD Section 4.11 (on page 110), <pthread.h>

CHANGE HISTORY

First released in Issue 5.

Issue 6

The following changes are made for alignment with IEEE Std 1003.1j-2000:

- The margin code in the SYNOPSIS is changed to THR to indicate that the functionality is now part of the Threads option (previously it was part of the Read-Write Locks option in IEEE Std 1003.1j-2000 and also part of the XSI extension).
- The DESCRIPTION is updated as follows:
 - The conditions under which writers have precedence over readers are specified.
 - The concept of read-write lock owner is deleted.
- The SEE ALSO section is updated.

Issue 7

SD5-XSH-ERN-183 is applied.

The *pthread_rwlock_unlock()* function is moved from the Threads option to the Base.

The [EINVAL] error for an uninitialized read-write lock object is removed; this condition results in undefined behavior.

The [EPERM] error for a read-write lock object for which the current thread does not hold a lock is removed; this condition results in undefined behavior.

54998 **NAME**

54999 pthread_rwlock_wrlock — lock a read-write lock object for writing

55000 **SYNOPSIS**

55001 #include <pthread.h>

55002 int pthread_rwlock_wrlock(pthread_rwlock_t *rwlock);

55003 **DESCRIPTION**55004 Refer to *pthread_rwlock_trywrlock()*.

55005 **NAME**

55006 pthread_rwlockattr_destroy, pthread_rwlockattr_init — destroy and initialize the read-write
55007 lock attributes object

55008 **SYNOPSIS**

55009 #include <pthread.h>

55010 int pthread_rwlockattr_destroy(pthread_rwlockattr_t *attr);
55011 int pthread_rwlockattr_init(pthread_rwlockattr_t *attr);

55012 **DESCRIPTION**

55013 The *pthread_rwlockattr_destroy()* function shall destroy a read-write lock attributes object. A
55014 destroyed *attr* attributes object can be reinitialized using *pthread_rwlockattr_init()*; the results of
55015 otherwise referencing the object after it has been destroyed are undefined. An implementation
55016 may cause *pthread_rwlockattr_destroy()* to set the object referenced by *attr* to an invalid value.

55017 The *pthread_rwlockattr_init()* function shall initialize a read-write lock attributes object *attr* with
55018 the default value for all of the attributes defined by the implementation.

55019 Results are undefined if *pthread_rwlockattr_init()* is called specifying an already initialized *attr*
55020 attributes object.

55021 After a read-write lock attributes object has been used to initialize one or more read-write locks,
55022 any function affecting the attributes object (including destruction) shall not affect any previously
55023 initialized read-write locks.

55024 The behavior is undefined if the value specified by the *attr* argument to
55025 *pthread_rwlockattr_destroy()* does not refer to an initialized read-write lock attributes object.

55026 **RETURN VALUE**

55027 If successful, the *pthread_rwlockattr_destroy()* and *pthread_rwlockattr_init()* functions shall return
55028 zero; otherwise, an error number shall be returned to indicate the error.

55029 **ERRORS**

55030 The *pthread_rwlockattr_init()* function shall fail if:

55031 [ENOMEM] Insufficient memory exists to initialize the read-write lock attributes object.

55032 These functions shall not return an error code of [EINTR].

55033 **EXAMPLES**

55034 None.

55035 **APPLICATION USAGE**

55036 None.

55037 **RATIONALE**

55038 If an implementation detects that the value specified by the *attr* argument to
55039 *pthread_rwlockattr_destroy()* does not refer to an initialized read-write lock attributes object, it is
55040 recommended that the function should fail and report an [EINVAL] error.

55041 **FUTURE DIRECTIONS**

55042 None.

55043 **SEE ALSO**

55044 *pthread_rwlock_destroy()*, *pthread_rwlockattr_getpshared()*

55045 XBD <pthread.h>

55046 **CHANGE HISTORY**

55047 First released in Issue 5.

55048 **Issue 6**

55049 The following changes are made for alignment with IEEE Std 1003.1j-2000:

- 55050 • The margin code in the SYNOPSIS is changed to THR to indicate that the functionality is
- 55051 now part of the Threads option (previously it was part of the Read-Write Locks option in
- 55052 IEEE Std 1003.1j-2000 and also part of the XSI extension).
- 55053 • The SEE ALSO section is updated.

55054 **Issue 7**55055 The *pthread_rwlockattr_destroy()* and *pthread_rwlockattr_init()* functions are moved from the

55056 Threads option to the Base.

55057 The [EINVAL] error for an uninitialized read-write lock attributes object is removed; this

55058 condition results in undefined behavior.

NAME

pthread_rwlockattr_getpshared, pthread_rwlockattr_setpshared — get and set the process-shared attribute of the read-write lock attributes object

SYNOPSIS

```
TSH #include <pthread.h>

int pthread_rwlockattr_getpshared(const pthread_rwlockattr_t
    *restrict attr, int *restrict pshared);
int pthread_rwlockattr_setpshared(pthread_rwlockattr_t *attr,
    int pshared);
```

DESCRIPTION

The *pthread_rwlockattr_getpshared()* function shall obtain the value of the *process-shared* attribute from the initialized attributes object referenced by *attr*. The *pthread_rwlockattr_setpshared()* function shall set the *process-shared* attribute in an initialized attributes object referenced by *attr*.

The *process-shared* attribute shall be set to `PTHREAD_PROCESS_SHARED` to permit a read-write lock to be operated upon by any thread that has access to the memory where the read-write lock is allocated, even if the read-write lock is allocated in memory that is shared by multiple processes. If the *process-shared* attribute is `PTHREAD_PROCESS_PRIVATE`, the read-write lock shall only be operated upon by threads created within the same process as the thread that initialized the read-write lock; if threads of differing processes attempt to operate on such a read-write lock, the behavior is undefined. The default value of the *process-shared* attribute shall be `PTHREAD_PROCESS_PRIVATE`.

Additional attributes, their default values, and the names of the associated functions to get and set those attribute values are implementation-defined.

The behavior is undefined if the value specified by the *attr* argument to *pthread_rwlockattr_getpshared()* or *pthread_rwlockattr_setpshared()* does not refer to an initialized read-write lock attributes object.

RETURN VALUE

Upon successful completion, the *pthread_rwlockattr_getpshared()* function shall return zero and store the value of the *process-shared* attribute of *attr* into the object referenced by the *pshared* parameter. Otherwise, an error number shall be returned to indicate the error.

If successful, the *pthread_rwlockattr_setpshared()* function shall return zero; otherwise, an error number shall be returned to indicate the error.

ERRORS

The *pthread_rwlockattr_setpshared()* function may fail if:

[EINVAL] The new value specified for the attribute is outside the range of legal values for that attribute.

These functions shall not return an error code of [EINTR].

55096 **EXAMPLES**

55097 None.

55098 **APPLICATION USAGE**

55099 None.

55100 **RATIONALE**

55101 None.

55102 **FUTURE DIRECTIONS**

55103 None.

55104 **SEE ALSO**55105 *pthread_rwlock_destroy()*, *pthread_rwlockattr_destroy()*

55106 XBD <pthread.h>

55107 **CHANGE HISTORY**

55108 First released in Issue 5.

55109 **Issue 6**

55110 The following changes are made for alignment with IEEE Std 1003.1j-2000:

- 55111 • The margin code in the SYNOPSIS is changed to THR TSH to indicate that the
- 55112 functionality is now part of the Threads option (previously it was part of the Read-Write
- 55113 Locks option in IEEE Std 1003.1j-2000 and also part of the XSI extension).
- 55114 • The DESCRIPTION notes that additional attributes are implementation-defined.
- 55115 • The SEE ALSO section is updated.

55116 The **restrict** keyword is added to the *pthread_rwlockattr_getpshared()* prototype for alignment

55117 with the ISO/IEC 9899:1999 standard.

55118 **Issue 7**

55119 The *pthread_rwlockattr_getpshared()* and *pthread_rwlockattr_setpshared()* functions are moved from

55120 the Threads option.

55121 The [EINVAL] error for an uninitialized read-write lock attributes object is removed; this

55122 condition results in undefined behavior.

55123 NAME

55124 pthread_rwlockattr_init — initialize the read-write lock attributes object

55125 SYNOPSIS

55126 #include <pthread.h>

55127 int pthread_rwlockattr_init(pthread_rwlockattr_t *attr);

55128 DESCRIPTION

55129 Refer to *pthread_rwlockattr_destroy()*.

55130 **NAME**

55131 pthread_rwlockattr_setpshared — set the process-shared attribute of the read-write lock
55132 attributes object

55133 **SYNOPSIS**

```
55134 TSH      #include <pthread.h>
55135          int pthread_rwlockattr_setpshared(pthread_rwlockattr_t *attr,
55136          int pshared);
```

55137 **DESCRIPTION**

55138 Refer to *pthread_rwlockattr_getpshared()*.

55139 NAME

55140 pthread_self — get the calling thread ID

55141 SYNOPSIS

55142 #include <pthread.h>

55143 pthread_t pthread_self(void);

55144 DESCRIPTION

55145 The *pthread_self()* function shall return the thread ID of the calling thread.

55146 RETURN VALUE

55147 The *pthread_self()* function shall always be successful and no return value is reserved to indicate
55148 an error.

55149 ERRORS

55150 No errors are defined.

55151 EXAMPLES

55152 None.

55153 APPLICATION USAGE

55154 None.

55155 RATIONALE

55156 The *pthread_self()* function provides a capability similar to the *getpid()* function for processes
55157 and the rationale is the same: the creation call does not provide the thread ID to the created
55158 thread.

55159 FUTURE DIRECTIONS

55160 None.

55161 SEE ALSO

55162 *pthread_create()*, *pthread_equal()*

55163 XBD <pthread.h>

55164 CHANGE HISTORY

55165 First released in Issue 5. Included for alignment with the POSIX Threads Extension.

55166 Issue 6

55167 The *pthread_self()* function is marked as part of the Threads option.

55168 Issue 7

55169 Austin Group Interpretation 1003.1-2001 #063 is applied, updating the RETURN VALUE section.

55170 The *pthread_self()* function is moved from the Threads option to the Base.

55171 **NAME**

55172 pthread_setcancelstate, pthread_setcanceltype, pthread_testcancel — set cancelability state

55173 **SYNOPSIS**

```
55174 #include <pthread.h>
55175 int pthread_setcancelstate(int state, int *oldstate);
55176 int pthread_setcanceltype(int type, int *oldtype);
55177 void pthread_testcancel(void);
```

55178 **DESCRIPTION**

55179 The *pthread_setcancelstate()* function shall atomically both set the calling thread's cancelability
 55180 state to the indicated *state* and return the previous cancelability state at the location referenced
 55181 by *oldstate*. Legal values for *state* are PTHREAD_CANCEL_ENABLE and
 55182 PTHREAD_CANCEL_DISABLE.

55183 The *pthread_setcanceltype()* function shall atomically both set the calling thread's cancelability
 55184 type to the indicated *type* and return the previous cancelability type at the location referenced by
 55185 *oldtype*. Legal values for *type* are PTHREAD_CANCEL_DEFERRED and
 55186 PTHREAD_CANCEL_ASYNCHRONOUS.

55187 The cancelability state and type of any newly created threads, including the thread in which
 55188 *main()* was first invoked, shall be PTHREAD_CANCEL_ENABLE and
 55189 PTHREAD_CANCEL_DEFERRED respectively.

55190 The *pthread_testcancel()* function shall create a cancellation point in the calling thread. The
 55191 *pthread_testcancel()* function shall have no effect if cancelability is disabled.

55192 **RETURN VALUE**

55193 If successful, the *pthread_setcancelstate()* and *pthread_setcanceltype()* functions shall return zero;
 55194 otherwise, an error number shall be returned to indicate the error.

55195 **ERRORS**

55196 The *pthread_setcancelstate()* function may fail if:

55197 [EINVAL] The specified state is not PTHREAD_CANCEL_ENABLE or
 55198 PTHREAD_CANCEL_DISABLE.

55199 The *pthread_setcanceltype()* function may fail if:

55200 [EINVAL] The specified type is not PTHREAD_CANCEL_DEFERRED or
 55201 PTHREAD_CANCEL_ASYNCHRONOUS.

55202 These functions shall not return an error code of [EINTR].

55203 **EXAMPLES**

55204 None.

55205 **APPLICATION USAGE**

55206 None.

55207 **RATIONALE**

55208 The *pthread_setcancelstate()* and *pthread_setcanceltype()* functions control the points at which a
 55209 thread may be asynchronously canceled. For cancellation control to be usable in modular
 55210 fashion, some rules need to be followed.

55211 An object can be considered to be a generalization of a procedure. It is a set of procedures and
 55212 global variables written as a unit and called by clients not known by the object. Objects may
 55213 depend on other objects.

55214 First, cancelability should only be disabled on entry to an object, never explicitly enabled. On

exit from an object, the cancelability state should always be restored to its value on entry to the object.

This follows from a modularity argument: if the client of an object (or the client of an object that uses that object) has disabled cancelability, it is because the client does not want to be concerned about cleaning up if the thread is canceled while executing some sequence of actions. If an object is called in such a state and it enables cancelability and a cancellation request is pending for that thread, then the thread is canceled, contrary to the wish of the client that disabled.

Second, the cancelability type may be explicitly set to either *deferred* or *asynchronous* upon entry to an object. But as with the cancelability state, on exit from an object the cancelability type should always be restored to its value on entry to the object.

Finally, only functions that are cancel-safe may be called from a thread that is asynchronously cancelable.

FUTURE DIRECTIONS

None.

SEE ALSO

[*pthread_cancel\(\)*](#)

XBD [*<pthread.h>*](#)

CHANGE HISTORY

First released in Issue 5. Included for alignment with the POSIX Threads Extension.

Issue 6

The *pthread_setcancelstate()*, *pthread_setcanceltype()*, and *pthread_testcancel()* functions are marked as part of the Threads option.

Issue 7

The *pthread_setcancelstate()*, *pthread_setcanceltype()*, and *pthread_testcancel()* functions are moved from the Threads option to the Base.

55240 **NAME**

55241 pthread_setconcurrency — set the level of concurrency

55242 **SYNOPSIS**55243 OB XSI `#include <pthread.h>`55244 `int pthread_setconcurrency(int new_level);`55245 **DESCRIPTION**55246 Refer to *pthread_getconcurrency()*.

55247 NAME

55248 pthread_setschedparam — dynamic thread scheduling parameters access (REALTIME
55249 THREADS)

55250 SYNOPSIS

```
55251 TPS    #include <pthread.h>
55252
55252    int pthread_setschedparam(pthread_t thread, int policy,
55253                               const struct sched_param *param);
```

55254 DESCRIPTION

55255 Refer to *pthread_getschedparam()*.

NAME

pthread_setschedprio — dynamic thread scheduling parameters access (REALTIME THREADS)

SYNOPSIS

```
#include <pthread.h>

int pthread_setschedprio(pthread_t thread, int prio);
```

DESCRIPTION

The *pthread_setschedprio()* function shall set the scheduling priority for the thread whose thread ID is given by *thread* to the value given by *prio*. See [Scheduling Policies](#) (on page 506) for a description on how this function call affects the ordering of the thread in the thread list for its new priority.

If the *pthread_setschedprio()* function fails, the scheduling priority of the target thread shall not be changed.

RETURN VALUE

If successful, the *pthread_setschedprio()* function shall return zero; otherwise, an error number shall be returned to indicate the error.

ERRORS

The *pthread_setschedprio()* function may fail if:

- [EINVAL] The value of *prio* is invalid for the scheduling policy of the specified thread. -
- [EPERM] The caller does not have appropriate privileges to set the scheduling priority of the specified thread. |

The *pthread_setschedprio()* function shall not return an error code of [EINTR].

EXAMPLES

None.

APPLICATION USAGE

None.

RATIONALE

The *pthread_setschedprio()* function provides a way for an application to temporarily raise its priority and then lower it again, without having the undesired side-effect of yielding to other threads of the same priority. This is necessary if the application is to implement its own strategies for bounding priority inversion, such as priority inheritance or priority ceilings. This capability is especially important if the implementation does not support the Thread Priority Protection or Thread Priority Inheritance options, but even if those options are supported it is needed if the application is to bound priority inheritance for other resources, such as semaphores.

The standard developers considered that while it might be preferable conceptually to solve this problem by modifying the specification of *pthread_setschedparam()*, it was too late to make such a change, as there may be implementations that would need to be changed. Therefore, this new function was introduced.

If an implementation detects use of a thread ID after the end of its lifetime, it is recommended that the function should fail and report an [ESRCH] error.

55297 **FUTURE DIRECTIONS**

55298 None.

55299 **SEE ALSO**55300 [Scheduling Policies](#) (on page 506), [pthread_getschedparam\(\)](#)55301 XBD [<pthread.h>](#)55302 **CHANGE HISTORY**

55303 First released in Issue 6. Included as a response to IEEE PASC Interpretation 1003.1 #96.

55304 **Issue 7**55305 The `pthread_setschedprio()` function is moved from the Threads option.

55306 Austin Group Interpretation 1003.1-2001 #069 is applied, updating the [EPERM] error.

55307 Austin Group Interpretation 1003.1-2001 #142 is applied, removing the [ESRCH] error condition.

55308 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0466 [314] is applied. +

55309 **NAME**

55310 pthread_setspecific — thread-specific data management

55311 **SYNOPSIS**

55312 #include <pthread.h>

55313 int pthread_setspecific(pthread_key_t *key*, const void **value*);55314 **DESCRIPTION**55315 Refer to *pthread_getspecific()*.

NAME

pthread_sigmask, sigprocmask — examine and change blocked signals

SYNOPSIS

```
CX      #include <signal.h>

55320      int pthread_sigmask(int how, const sigset_t *restrict set,
55321                          sigset_t *restrict oset);
55322      int sigprocmask(int how, const sigset_t *restrict set,
55323                      sigset_t *restrict oset);
```

DESCRIPTION

The *pthread_sigmask()* function shall examine or change (or both) the calling thread's signal mask, regardless of the number of threads in the process. The function shall be equivalent to *sigprocmask()*, without the restriction that the call be made in a single-threaded process.

In a single-threaded process, the *sigprocmask()* function shall examine or change (or both) the signal mask of the calling thread.

If the argument *set* is not a null pointer, it points to a set of signals to be used to change the currently blocked set.

The argument *how* indicates the way in which the set is changed, and the application shall ensure it consists of one of the following values:

SIG_BLOCK The resulting set shall be the union of the current set and the signal set pointed to by *set*.

SIG_SETMASK The resulting set shall be the signal set pointed to by *set*.

SIG_UNBLOCK The resulting set shall be the intersection of the current set and the complement of the signal set pointed to by *set*.

If the argument *oset* is not a null pointer, the previous mask shall be stored in the location pointed to by *oset*. If *set* is a null pointer, the value of the argument *how* is not significant and the thread's signal mask shall be unchanged; thus the call can be used to enquire about currently blocked signals.

If there are any pending unblocked signals after the call to *sigprocmask()*, at least one of those signals shall be delivered before the call to *sigprocmask()* returns.

It is not possible to block those signals which cannot be ignored. This shall be enforced by the system without causing an error to be indicated.

If any of the SIGFPE, SIGILL, SIGSEGV, or SIGBUS signals are generated while they are blocked, the result is undefined, unless the signal was generated by the action of another process, or by one of the functions *kill()*, *pthread_kill()*, *raise()*, or *sigqueue()*.

If *sigprocmask()* fails, the thread's signal mask shall not be changed.

The use of the *sigprocmask()* function is unspecified in a multi-threaded process.

RETURN VALUE

Upon successful completion *pthread_sigmask()* shall return 0; otherwise, it shall return the corresponding error number.

Upon successful completion, *sigprocmask()* shall return 0; otherwise, -1 shall be returned, *errno* shall be set to indicate the error, and the signal mask of the process shall be unchanged.

ERRORS

The *pthread_sigmask()* and *sigprocmask()* functions shall fail if:

[EINVAL] The value of the *how* argument is not equal to one of the defined values.

The *pthread_sigmask()* function shall not return an error code of [EINTR].

EXAMPLES**Signaling in a Multi-Threaded Process**

This example shows the use of *pthread_sigmask()* in order to deal with signals in a multi-threaded process. It provides a fairly general framework that could be easily adapted/extended.

```
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#include <signal.h>
#include <string.h>
#include <errno.h>
...

static sigset_t  signal_mask; /* signals to block          */

int main (int argc, char *argv[])
{
    pthread_t  sig_thr_id;      /* signal handler thread ID */
    int        rc;              /* return code              */

    sigemptyset (&signal_mask);
    sigaddset (&signal_mask, SIGINT);
    sigaddset (&signal_mask, SIGTERM);
    rc = pthread_sigmask (SIG_BLOCK, &signal_mask, NULL);
    if (rc != 0) {
        /* handle error */
        ...
    }
    /* any newly created threads inherit the signal mask */

    rc = pthread_create (&sig_thr_id, NULL, signal_thread, NULL);
    if (rc != 0) {
        /* handle error */
        ...
    }

    /* APPLICATION CODE */
    ...
}

void *signal_thread (void *arg)
{
    int        sig_caught;      /* signal caught            */
    int        rc;              /* returned code            */

    rc = sigwait (&signal_mask, &sig_caught);
    if (rc != 0) {
        /* handle error */
    }
}
```

```

55402         switch (sig_caught)
55403         {
55404             case SIGINT:      /* process SIGINT */
55405                 ...
55406                 break;
55407             case SIGTERM:     /* process SIGTERM */
55408                 ...
55409                 break;
55410             default:          /* should normally not happen */
55411                 fprintf (stderr, "\nUnexpected signal %d\n", sig_caught);
55412                 break;
55413         }
55414     }

```

APPLICATION USAGE

None.

RATIONALE

When a thread's signal mask is changed in a signal-catching function that is installed by *sigaction()*, the restoration of the signal mask on return from the signal-catching function overrides that change (see *sigaction()*). If the signal-catching function was installed with *signal()*, it is unspecified whether this occurs.

See *kill()* for a discussion of the requirement on delivery of signals.

FUTURE DIRECTIONS

None.

SEE ALSO

exec, *kill()*, *sigaction()*, *sigaddset()*, *sigdelset()*, *sigemptyset()*, *sigfillset()*, *sigismember()*, *sigpending()*, *sigqueue()*, *sigsuspend()*

XBD <signal.h>

CHANGE HISTORY

First released in Issue 3. Included for alignment with the POSIX.1-1988 standard.

Issue 5

The DESCRIPTION is updated for alignment with the POSIX Threads Extension.

The *pthread_sigmask()* function is added for alignment with the POSIX Threads Extension.

Issue 6

The *pthread_sigmask()* function is marked as part of the Threads option.

The SYNOPSIS for *sigprocmask()* is marked as a CX extension to note that the presence of this function in the <signal.h> header is an extension to the ISO C standard.

The following changes are made for alignment with the ISO POSIX-1:1996 standard:

- The DESCRIPTION is updated to explicitly state the functions which may generate the signal.

The normative text is updated to avoid use of the term “must” for application requirements.

The **restrict** keyword is added to the *pthread_sigmask()* and *sigprocmask()* prototypes for alignment with the ISO/IEC 9899:1999 standard.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/105 is applied, updating “process’ signal mask” to “thread’s signal mask” in the DESCRIPTION and RATIONALE sections.

55446	IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/106 is applied, adding the example to the EXAMPLES section.	
55447		
55448	Issue 7	
55449	The <i>pthread_sigmask()</i> function is moved from the Threads option to the Base.	
55450	POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0467 [319] is applied.	+

NAME

pthread_spin_destroy, pthread_spin_init — destroy or initialize a spin lock object

SYNOPSIS

```
#include <pthread.h>
```

```
int pthread_spin_destroy(pthread_spinlock_t *lock);
```

```
int pthread_spin_init(pthread_spinlock_t *lock, int pshared);
```

DESCRIPTION

The *pthread_spin_destroy()* function shall destroy the spin lock referenced by *lock* and release any resources used by the lock. The effect of subsequent use of the lock is undefined until the lock is reinitialized by another call to *pthread_spin_init()*. The results are undefined if *pthread_spin_destroy()* is called when a thread holds the lock, or if this function is called with an uninitialized thread spin lock.

The *pthread_spin_init()* function shall allocate any resources required to use the spin lock referenced by *lock* and initialize the lock to an unlocked state.

TSH

If the Thread Process-Shared Synchronization option is supported and the value of *pshared* is `PTHREAD_PROCESS_SHARED`, the implementation shall permit the spin lock to be operated upon by any thread that has access to the memory where the spin lock is allocated, even if it is allocated in memory that is shared by multiple processes.

If the Thread Process-Shared Synchronization option is supported and the value of *pshared* is `PTHREAD_PROCESS_PRIVATE`, or if the option is not supported, the spin lock shall only be operated upon by threads created within the same process as the thread that initialized the spin lock. If threads of differing processes attempt to operate on such a spin lock, the behavior is undefined.

The results are undefined if *pthread_spin_init()* is called specifying an already initialized spin lock. The results are undefined if a spin lock is used without first being initialized.

If the *pthread_spin_init()* function fails, the lock is not initialized and the contents of *lock* are undefined.

Only the object referenced by *lock* may be used for performing synchronization.

The result of referring to copies of that object in calls to *pthread_spin_destroy()*, *pthread_spin_lock()*, *pthread_spin_trylock()*, or *pthread_spin_unlock()* is undefined.

RETURN VALUE

Upon successful completion, these functions shall return zero; otherwise, an error number shall be returned to indicate the error.

ERRORS

The *pthread_spin_init()* function shall fail if:

[EAGAIN] The system lacks the necessary resources to initialize another spin lock.

[ENOMEM] Insufficient memory exists to initialize the lock.

These functions shall not return an error code of [EINTR].

EXAMPLES

None.

APPLICATION USAGE

None.

RATIONALE

If an implementation detects that the value specified by the *lock* argument to *pthread_spin_destroy()* does not refer to an initialized spin lock object, it is recommended that the function should fail and report an [EINVAL] error.

If an implementation detects that the value specified by the *lock* argument to *pthread_spin_destroy()* or *pthread_spin_init()* refers to a locked spin lock object, or detects that the value specified by the *lock* argument to *pthread_spin_init()* refers to an already initialized spin lock object, it is recommended that the function should fail and report an [EBUSY] error.

FUTURE DIRECTIONS

None.

SEE ALSO

pthread_spin_lock(), *pthread_spin_unlock()*

XBD <pthread.h>

CHANGE HISTORY

First released in Issue 6. Derived from IEEE Std 1003.1j-2000.

In the SYNOPSIS, the inclusion of <sys/types.h> is no longer required.

Issue 7

The *pthread_spin_destroy()* and *pthread_spin_init()* functions are moved from the Spin Locks option to the Base.

The [EINVAL] error for an uninitialized spin lock object is removed; this condition results in undefined behavior.

The [EBUSY] error for a locked spin lock object or an already initialized spin lock object is removed; this condition results in undefined behavior.

55516 NAME

55517 pthread_spin_lock, pthread_spin_trylock — lock a spin lock object

55518 SYNOPSIS

55519 #include <pthread.h>
55520 int pthread_spin_lock(pthread_spinlock_t *lock);
55521 int pthread_spin_trylock(pthread_spinlock_t *lock);

55522 DESCRIPTION

55523 The *pthread_spin_lock()* function shall lock the spin lock referenced by *lock*. The calling thread
55524 shall acquire the lock if it is not held by another thread. Otherwise, the thread shall spin (that is,
55525 shall not return from the *pthread_spin_lock()* call) until the lock becomes available. The results are
55526 undefined if the calling thread holds the lock at the time the call is made. The
55527 *pthread_spin_trylock()* function shall lock the spin lock referenced by *lock* if it is not held by any
55528 thread. Otherwise, the function shall fail.

55529 The results are undefined if any of these functions is called with an uninitialized spin lock.

55530 RETURN VALUE

55531 Upon successful completion, these functions shall return zero; otherwise, an error number shall
55532 be returned to indicate the error.

55533 ERRORS

55534 The *pthread_spin_lock()* function may fail if:

55535 [EDEADLK] A deadlock condition was detected.

55536 The *pthread_spin_trylock()* function shall fail if:

55537 [EBUSY] A thread currently holds the lock.

55538 These functions shall not return an error code of [EINTR].

55539 EXAMPLES

55540 None.

55541 APPLICATION USAGE

55542 Applications using this function may be subject to priority inversion, as discussed in XBD
55543 [Section 3.287](#) (on page 79).

55544 RATIONALE

55545 If an implementation detects that the value specified by the *lock* argument to *pthread_spin_lock()*
55546 or *pthread_spin_trylock()* does not refer to an initialized spin lock object, it is recommended that
55547 the function should fail and report an [EINVAL] error.

55548 If an implementation detects that the value specified by the *lock* argument to *pthread_spin_lock()*
55549 refers to a spin lock object for which the calling thread already holds the lock, it is recommended
55550 that the function should fail and report an [EDEADLK] error.

55551 FUTURE DIRECTIONS

55552 None.

55553 SEE ALSO

55554 [pthread_spin_destroy\(\)](#), [pthread_spin_unlock\(\)](#)

55555 XBD [Section 3.287](#) (on page 79), [Section 4.11](#) (on page 110), [<pthread.h>](#)

CHANGE HISTORY

55556 First released in Issue 6. Derived from IEEE Std 1003.1j-2000.

55558 In the SYNOPSIS, the inclusion of **<sys/types.h>** is no longer required.

55559 IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/107 is applied, updating the ERRORS
55560 section so that the [EDEADLK] error includes detection of a deadlock condition.

Issue 7

55562 The *pthread_spin_lock()* and *pthread_spin_trylock()* functions are moved from the Spin Locks
55563 option to the Base.

55564 The [EINVAL] error for an uninitialized spin lock object is removed; this condition results in
55565 undefined behavior.

55566 The [EDEADLK] error for a spin lock object for which the calling thread already holds the lock is
55567 removed; this condition results in undefined behavior.

55568 **NAME**

55569 pthread_spin_unlock — unlock a spin lock object

55570 **SYNOPSIS**

55571 #include <pthread.h>

55572 int pthread_spin_unlock(pthread_spinlock_t *lock);

55573 **DESCRIPTION**55574 The *pthread_spin_unlock()* function shall release the spin lock referenced by *lock* which was
55575 locked via the *pthread_spin_lock()* or *pthread_spin_trylock()* functions.

55576 The results are undefined if the lock is not held by the calling thread.

55577 If there are threads spinning on the lock when *pthread_spin_unlock()* is called, the lock becomes
55578 available and an unspecified spinning thread shall acquire the lock.

55579 The results are undefined if this function is called with an uninitialized thread spin lock.

55580 **RETURN VALUE**55581 Upon successful completion, the *pthread_spin_unlock()* function shall return zero; otherwise, an
55582 error number shall be returned to indicate the error.55583 **ERRORS**

55584 This function shall not return an error code of [EINTR].

55585 **EXAMPLES**

55586 None.

55587 **APPLICATION USAGE**

55588 None.

55589 **RATIONALE**55590 If an implementation detects that the value specified by the *lock* argument to
55591 *pthread_spin_unlock()* does not refer to an initialized spin lock object, it is recommended that the
55592 function should fail and report an [EINVAL] error.55593 If an implementation detects that the value specified by the *lock* argument to
55594 *pthread_spin_unlock()* refers to a spin lock object for which the current thread does not hold the
55595 lock, it is recommended that the function should fail and report an [EPERM] error.55596 **FUTURE DIRECTIONS**

55597 None.

55598 **SEE ALSO**55599 *pthread_spin_destroy()*, *pthread_spin_lock()*

55600 XBD Section 4.11 (on page 110), <pthread.h>

55601 **CHANGE HISTORY**

55602 First released in Issue 6. Derived from IEEE Std 1003.1j-2000.

55603 In the SYNOPSIS, the inclusion of <sys/types.h> is no longer required.

55604 **Issue 7**55605 The *pthread_spin_unlock()* function is moved from the Spin Locks option to the Base.55606 The [EINVAL] error for an uninitialized spin lock object is removed; this condition results in
55607 undefined behavior.

55608
55609

The [EPERM] error for a spin lock object for which the current thread does not hold the lock is removed; this condition results in undefined behavior.

55610 NAME

55611 pthread_testcancel — set cancelability state

55612 SYNOPSIS

55613 #include <pthread.h>

55614 void pthread_testcancel(void);

55615 DESCRIPTION

55616 Refer to *pthread_setcancelstate()*.

55617 **NAME**

55618 ptsname — get name of the slave pseudo-terminal device

55619 **SYNOPSIS**

```
55620 XSI      #include <stdlib.h>
55621          char *ptsname(int fildes);
```

55622 **DESCRIPTION**

55623 The *ptsname()* function shall return the name of the slave pseudo-terminal device associated
 55624 with a master pseudo-terminal device. The *fildes* argument is a file descriptor that refers to the
 55625 master device. The *ptsname()* function shall return a pointer to a string containing the pathname
 55626 of the corresponding slave device.

55627 The *ptsname()* function need not be thread-safe.

55628 **RETURN VALUE**

55629 Upon successful completion, *ptsname()* shall return a pointer to a string which is the name of the
 55630 pseudo-terminal slave device. Upon failure, *ptsname()* shall return a null pointer. This could
 55631 occur if *fildes* is an invalid file descriptor or if the slave device name does not exist in the file
 55632 system.

55633 The application shall not modify the string returned. The returned pointer might be invalidated +
 55634 or the string content might be overwritten by a subsequent call to *ptsname()*.

55635 **ERRORS**

55636 No errors are defined.

55637 **EXAMPLES**

55638 None.

55639 **APPLICATION USAGE**

55640 None. -

55641 **RATIONALE**

55642 See the RATIONALE section for *posix_openpt()*. +

55643 **FUTURE DIRECTIONS**

55644 None.

55645 **SEE ALSO**

55646 *grantpt()*, *open()*, *posix_openpt()*, *ttyname()*, *unlockpt()*

55647 XBD [<stdlib.h>](#)

55648 **CHANGE HISTORY**

55649 First released in Issue 4, Version 2.

55650 **Issue 5**

55651 Moved from X/OPEN UNIX extension to BASE.

55652 A note indicating that this function need not be reentrant is added to the DESCRIPTION.

55653 **Issue 7**

55654 Austin Group Interpretation 1003.1-2001 #156 is applied.

55655 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0468 [75] and XSH/TC1-2008/0469 +
 55656 [96] are applied.

NAME

putc — put a byte on a stream

SYNOPSIS

```
#include <stdio.h>
```

```
int putc(int c, FILE *stream);
```

DESCRIPTION

CX The functionality described on this reference page is aligned with the ISO C standard. Any conflict between the requirements described here and the ISO C standard is unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

The *putc()* function shall be equivalent to *fputc()*, except that if it is implemented as a macro it may evaluate *stream* more than once, so the argument should never be an expression with side-effects.

RETURN VALUE

Refer to *fputc()*.

ERRORS

Refer to *fputc()*.

EXAMPLES

None.

APPLICATION USAGE

Since it may be implemented as a macro, *putc()* may treat a *stream* argument with side-effects incorrectly. In particular, *putc(c,*f++)* does not necessarily work correctly. Therefore, use of this function is not recommended in such situations; *fputc()* should be used instead.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

Section 2.5 (on page 495), *fputc()*

XBD <stdio.h>

CHANGE HISTORY

First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 7

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0470 [14] is applied.

+

55690 **NAME**

55691 putc_unlocked — stdio with explicit client locking

55692 **SYNOPSIS**

```
55693 CX      #include <stdio.h>
55694          int putc_unlocked(int c, FILE *stream);
```

55695 **DESCRIPTION**55696 Refer to *getc_unlocked()*.

55697 NAME

55698 putchar — put a byte on a stdout stream

55699 SYNOPSIS

55700 #include <stdio.h>

55701 int putchar(int c);

55702 DESCRIPTION

55703 CX The functionality described on this reference page is aligned with the ISO C standard. Any
55704 conflict between the requirements described here and the ISO C standard is unintentional. This
55705 volume of POSIX.1-2008 defers to the ISO C standard.

55706 The function call *putchar(c)* shall be equivalent to *putc(c,stdout)*.

55707 RETURN VALUE

55708 Refer to *fputc()*.

55709 ERRORS

55710 Refer to *fputc()*.

55711 EXAMPLES

55712 None.

55713 APPLICATION USAGE

55714 None.

55715 RATIONALE

55716 None.

55717 FUTURE DIRECTIONS

55718 None.

55719 SEE ALSO

55720 [Section 2.5](#) (on page 495), *putc()*

55721 XBD [<stdio.h>](#)

55722 CHANGE HISTORY

55723 First released in Issue 1. Derived from Issue 1 of the SVID.

55724 Issue 7

55725 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0471 [14] is applied.

+

55726 **NAME**

55727 putchar_unlocked — stdio with explicit client locking

55728 **SYNOPSIS**55729 CX `#include <stdio.h>`55730 `int putchar_unlocked(int c);`55731 **DESCRIPTION**55732 Refer to *getc_unlocked()*.

NAME

putenv — change or add a value to an environment

SYNOPSIS

```
XSI      #include <stdlib.h>
int putenv(char *string);
```

DESCRIPTION

The *putenv()* function shall use the *string* argument to set environment variable values. The *string* argument should point to a string of the form "*name=value*". The *putenv()* function shall make the value of the environment variable *name* equal to *value* by altering an existing variable or creating a new one. In either case, the string pointed to by *string* shall become part of the environment, so altering the string shall change the environment.

The *putenv()* function need not be thread-safe.

RETURN VALUE

Upon successful completion, *putenv()* shall return 0; otherwise, it shall return a non-zero value and set *errno* to indicate the error.

ERRORS

The *putenv()* function may fail if:

[ENOMEM] Insufficient memory was available.

EXAMPLES**Changing the Value of an Environment Variable**

The following example changes the value of the *HOME* environment variable to the value */usr/home*.

```
#include <stdlib.h>
...
static char *var = "HOME=/usr/home";
int ret;

ret = putenv(var);
```

APPLICATION USAGE

The *putenv()* function manipulates the environment pointed to by *environ*, and can be used in conjunction with *getenv()*.

See *exec()* for restrictions on changing the environment in multi-threaded applications.

This routine may use *malloc()* to enlarge the environment.

A potential error is to call *putenv()* with an automatic variable as the argument, then return from the calling function while *string* is still part of the environment.

Although the space used by *string* is no longer used once a new string which defines *name* is passed to *putenv()*, if any thread in the application has used *getenv()* to retrieve a pointer to this variable, it should not be freed by calling *free()*. If the changed environment variable is one known by the system (such as the locale environment variables) the application should never free the buffer used by earlier calls to *putenv()* for the same variable.

The *setenv()* function is preferred over this function. One reason is that *putenv()* is optional and therefore less portable. Another is that using *putenv()* can slow down environment searches, as explained in the RATIONALE section for *getenv()*.

55775 **RATIONALE**55776 Refer to the RATIONALE section in *setenv()*.55777 **FUTURE DIRECTIONS**

55778 None.

55779 **SEE ALSO**55780 *exec*, *free()*, *getenv()*, *malloc()*, *setenv()*55781 XBD **<stdlib.h>**55782 **CHANGE HISTORY**

55783 First released in Issue 1. Derived from Issue 1 of the SVID.

55784 **Issue 5**55785 The type of the argument to this function is changed from **const char *** to **char ***. This was
55786 indicated as a FUTURE DIRECTION in previous issues.

55787 A note indicating that this function need not be reentrant is added to the DESCRIPTION.

55788 **Issue 6**55789 IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/48 is applied, clarifying wording in the
55790 DESCRIPTION and adding a new paragraph into APPLICATION USAGE referring readers to
55791 *exec*.55792 **Issue 7**

55793 Austin Group Interpretation 1003.1-2001 #156 is applied.

55794 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0472 [167], XSH/TC1-2008/0473 [167], +
55795 XSH/TC1-2008/0474 [273,438], and XSH/TC1-2008/0475 [273] are applied.

NAME

putmsg, putpmsg — send a message on a STREAM (STREAMS)

SYNOPSIS

```
#include <stropts.h>

int putmsg(int fildes, const struct strbuf *ctlptr,
            const struct strbuf *dataptr, int flags);
int putpmsg(int fildes, const struct strbuf *ctlptr,
             const struct strbuf *dataptr, int band, int flags);
```

DESCRIPTION

The *putmsg()* function shall create a message from a process buffer(s) and send the message to a STREAMS file. The message may contain either a data part, a control part, or both. The data and control parts are distinguished by placement in separate buffers, as described below. The semantics of each part are defined by the STREAMS module that receives the message.

The *putpmsg()* function is equivalent to *putmsg()*, except that the process can send messages in different priority bands. Except where noted, all requirements on *putmsg()* also pertain to *putpmsg()*.

The *fildes* argument specifies a file descriptor referencing an open STREAM. The *ctlptr* and *dataptr* arguments each point to a **strbuf** structure.

The *ctlptr* argument points to the structure describing the control part, if any, to be included in the message. The *buf* member in the **strbuf** structure points to the buffer where the control information resides, and the *len* member indicates the number of bytes to be sent. The *maxlen* member is not used by *putmsg()*. In a similar manner, the argument *dataptr* specifies the data, if any, to be included in the message. The *flags* argument indicates what type of message should be sent and is described further below.

To send the data part of a message, the application shall ensure that *dataptr* is not a null pointer and the *len* member of *dataptr* is 0 or greater. To send the control part of a message, the application shall ensure that the corresponding values are set for *ctlptr*. No data (control) part shall be sent if either *dataptr(ctlptr)* is a null pointer or the *len* member of *dataptr(ctlptr)* is set to -1.

For *putmsg()*, if a control part is specified and *flags* is set to RS_HIPRI, a high priority message shall be sent. If no control part is specified, and *flags* is set to RS_HIPRI, *putmsg()* shall fail and set *errno* to [EINVAL]. If *flags* is set to 0, a normal message (priority band equal to 0) shall be sent. If a control part and data part are not specified and *flags* is set to 0, no message shall be sent and 0 shall be returned.

For *putpmsg()*, the flags are different. The *flags* argument is a bitmask with the following mutually-exclusive flags defined: MSG_HIPRI and MSG_BAND. If *flags* is set to 0, *putpmsg()* shall fail and set *errno* to [EINVAL]. If a control part is specified and *flags* is set to MSG_HIPRI and *band* is set to 0, a high-priority message shall be sent. If *flags* is set to MSG_HIPRI and either no control part is specified or *band* is set to a non-zero value, *putpmsg()* shall fail and set *errno* to [EINVAL]. If *flags* is set to MSG_BAND, then a message shall be sent in the priority band specified by *band*. If a control part and data part are not specified and *flags* is set to MSG_BAND, no message shall be sent and 0 shall be returned.

55838 The *putmsg()* function shall block if the STREAM write queue is full due to internal flow control
 55839 conditions, with the following exceptions:

- 55840 • For high-priority messages, *putmsg()* shall not block on this condition and continues
 55841 processing the message.
- 55842 • For other messages, *putmsg()* shall not block but shall fail when the write queue is full and
 55843 O_NONBLOCK is set.

55844 The *putmsg()* function shall also block, unless prevented by lack of internal resources, while
 55845 waiting for the availability of message blocks in the STREAM, regardless of priority or whether
 55846 O_NONBLOCK has been specified. No partial message shall be sent.

55847 RETURN VALUE

55848 Upon successful completion, *putmsg()* and *putpmsg()* shall return 0; otherwise, they shall return
 55849 -1 and set *errno* to indicate the error.

55850 ERRORS

55851 The *putmsg()* and *putpmsg()* functions shall fail if:

55852 55853 55854	[EAGAIN]	A non-priority message was specified, the O_NONBLOCK flag is set, and the STREAM write queue is full due to internal flow control conditions; or buffers could not be allocated for the message that was to be created.
55855	[EBADF]	<i>fildev</i> is not a valid file descriptor open for writing.
55856	[EINTR]	A signal was caught during <i>putmsg()</i> .
55857 55858 55859 55860 55861	[EINVAL]	An undefined value is specified in <i>flags</i> , or <i>flags</i> is set to RS_HIPRI or MSG_HIPRI and no control part is supplied, or the STREAM or multiplexer referenced by <i>fildev</i> is linked (directly or indirectly) downstream from a multiplexer, or <i>flags</i> is set to MSG_HIPRI and <i>band</i> is non-zero (for <i>putpmsg()</i> only).
55862 55863	[ENOSR]	Buffers could not be allocated for the message that was to be created due to insufficient STREAMS memory resources.
55864	[ENOSTR]	A STREAM is not associated with <i>fildev</i> .
55865	[ENXIO]	A hangup condition was generated downstream for the specified STREAM.
55866 55867	[EPIPE] or [EIO]	The <i>fildev</i> argument refers to a STREAMS-based pipe and the other end of the pipe is closed. A SIGPIPE signal is generated for the calling thread.
55868 55869 55870 55871 55872 55873	[ERANGE]	The size of the data part of the message does not fall within the range specified by the maximum and minimum packet sizes of the topmost STREAM module. This value is also returned if the control part of the message is larger than the maximum configured size of the control part of a message, or if the data part of a message is larger than the maximum configured size of the data part of a message.

55874 In addition, *putmsg()* and *putpmsg()* shall fail if the STREAM head had processed an
 55875 asynchronous error before the call. In this case, the value of *errno* does not reflect the result of
 55876 *putmsg()* or *putpmsg()*, but reflects the prior error.

55877 **EXAMPLES**55878 **Sending a High-Priority Message**

55879 The value of *fd* is assumed to refer to an open STREAMS file. This call to *putmsg()* does the
 55880 following:

- 55881 1. Creates a high-priority message with a control part and a data part, using the buffers
 55882 pointed to by *ctrlbuf* and *databuf*, respectively.
- 55883 2. Sends the message to the STREAMS file identified by *fd*.

```
55884 #include <stropts.h>
55885 #include <string.h>
55886 ...
55887 int fd;
55888 char *ctrlbuf = "This is the control part";
55889 char *databuf = "This is the data part";
55890 struct strbuf ctrl;
55891 struct strbuf data;
55892 int ret;

55893 ctrl.buf = ctrlbuf;
55894 ctrl.len = strlen(ctrlbuf);

55895 data.buf = databuf;
55896 data.len = strlen(databuf);

55897 ret = putmsg(fd, &ctrl, &data, MSG_HIPRI);
```

55898 **Using putpmsg()**

55899 This example has the same effect as the previous example. In this example, however, the
 55900 *putpmsg()* function creates and sends the message to the STREAMS file.

```
55901 #include <stropts.h>
55902 #include <string.h>
55903 ...
55904 int fd;
55905 char *ctrlbuf = "This is the control part";
55906 char *databuf = "This is the data part";
55907 struct strbuf ctrl;
55908 struct strbuf data;
55909 int ret;

55910 ctrl.buf = ctrlbuf;
55911 ctrl.len = strlen(ctrlbuf);

55912 data.buf = databuf;
55913 data.len = strlen(databuf);

55914 ret = putpmsg(fd, &ctrl, &data, 0, MSG_HIPRI);
```

55915 **APPLICATION USAGE**

55916 None.

55917 **RATIONALE**

55918 None.

55919 **FUTURE DIRECTIONS**55920 The *putmsg()* and *putpmsg()* functions may be removed in a future version.55921 **SEE ALSO**55922 [Section 2.6](#) (on page 499), *getmsg()*, *poll()*, *read()*, *write()*55923 XBD [<stropts.h>](#)55924 **CHANGE HISTORY**

55925 First released in Issue 4, Version 2.

55926 **Issue 5**

55927 Moved from X/OPEN UNIX extension to BASE.

55928 The following text is removed from the DESCRIPTION: “The STREAM head guarantees that the
55929 control part of a message generated by *putmsg()* is at least 64 bytes in length”.

55930 **Issue 6**

55931 This function is marked as part of the XSI STREAMS Option Group.

55932 The normative text is updated to avoid use of the term “must” for application requirements.

55933 **Issue 7**55934 The *putmsg()* and *putpmsg()* functions are marked obsolescent.

55935 **NAME**

55936 puts — put a string on standard output

55937 **SYNOPSIS**

55938 #include <stdio.h>

55939 int puts(const char *s);

55940 **DESCRIPTION**

55941 CX The functionality described on this reference page is aligned with the ISO C standard. Any
 55942 conflict between the requirements described here and the ISO C standard is unintentional. This
 55943 volume of POSIX.1-2008 defers to the ISO C standard.

55944 The *puts()* function shall write the string pointed to by *s*, followed by a <newline>, to the
 55945 standard output stream *stdout*. The terminating null byte shall not be written.

55946 CX The last data modification and last file status change timestamps of the file shall be marked for
 55947 update between the successful execution of *puts()* and the next successful completion of a call to
 55948 *fflush()* or *fclose()* on the same stream or a call to *exit()* or *abort()*.

55949 **RETURN VALUE**

55950 Upon successful completion, *puts()* shall return a non-negative number. Otherwise, it shall
 55951 CX return EOF, shall set an error indicator for the stream, and *errno* shall be set to indicate the error.

55952 **ERRORS**55953 Refer to *fputc()*.55954 **EXAMPLES**55955 **Printing to Standard Output**

55956 The following example gets the current time, converts it to a string using *localtime()* and
 55957 *asctime()*, and prints it to standard output using *puts()*. It then prints the number of minutes to
 55958 an event for which it is waiting.

```
55959 #include <time.h>
55960 #include <stdio.h>
55961 ...
55962 time_t now;
55963 int minutes_to_event;
55964 ...
55965 time(&now);
55966 printf("The time is ");
55967 puts(asctime(localtime(&now)));
55968 printf("There are %d minutes to the event.\n",
55969     minutes_to_event);
55970 ...
```

55971 **APPLICATION USAGE**55972 The *puts()* function appends a <newline>, while *fputs()* does not.

55973 This volume of POSIX.1-2008 requires that successful completion simply return a non-negative +
 55974 integer. There are at least three known different implementation conventions for this +
 55975 requirement: +

- 55976 • Return a constant value. +

55977	• Return the last character written.	+
55978	• Return the number of bytes written. Note that this implementation convention cannot be	+
55979	adhered to for strings longer than {INT_MAX} bytes as the value would not be	+
55980	representable in the return type of the function. For backwards compatibility,	+
55981	implementations can return the number of bytes for strings of up to {INT_MAX} bytes, and	+
55982	return {INT_MAX} for all longer strings.	+
55983	RATIONALE	
55984	None.	
55985	FUTURE DIRECTIONS	
55986	None.	
55987	SEE ALSO	
55988	Section 2.5 (on page 495), fopen() , fputs() , putc()	
55989	XBD <stdio.h>	
55990	CHANGE HISTORY	
55991	First released in Issue 1. Derived from Issue 1 of the SVID.	
55992	Issue 6	
55993	Extensions beyond the ISO C standard are marked.	
55994	Issue 7	
55995	Changes are made related to support for finegrained timestamps.	
55996	POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0476 [174,412] and	+
55997	XSH/TC1-2008/0477 [14] are applied.	

55998 NAME

55999 pututxline — put an entry into the user accounting database

56000 SYNOPSIS

```
56001 XSI      #include <utmpx.h>  
56002      struct utmpx *pututxline(const struct utmpx *utmpx);
```

56003 DESCRIPTION

56004 Refer to *endutxent()*.

56005 **NAME**

56006 putwc — put a wide character on a stream

56007 **SYNOPSIS**

56008 #include <stdio.h>

56009 #include <wchar.h>

56010 wint_t putwc(wchar_t *wc*, FILE **stream*);56011 **DESCRIPTION**

56012 CX The functionality described on this reference page is aligned with the ISO C standard. Any
 56013 conflict between the requirements described here and the ISO C standard is unintentional. This
 56014 volume of POSIX.1-2008 defers to the ISO C standard.

56015 The *putwc()* function shall be equivalent to *fputwc()*, except that if it is implemented as a macro
 56016 it may evaluate *stream* more than once, so the argument should never be an expression with
 56017 side-effects.

56018 **RETURN VALUE**56019 Refer to *fputwc()*.56020 **ERRORS**56021 Refer to *fputwc()*.56022 **EXAMPLES**

56023 None.

56024 **APPLICATION USAGE**

56025 Since it may be implemented as a macro, *putwc()* may treat a *stream* argument with side-effects
 56026 incorrectly. In particular, *putwc(wc,*f++)* need not work correctly. Therefore, use of this function
 56027 is not recommended; *fputwc()* should be used instead.

56028 **RATIONALE**

56029 None.

56030 **FUTURE DIRECTIONS**

56031 None.

56032 **SEE ALSO**56033 [Section 2.5](#) (on page 495), *fputwc()*56034 XBD [<stdio.h>](#), [<wchar.h>](#)56035 **CHANGE HISTORY**

56036 First released as a World-wide Portability Interface in Issue 4.

56037 **Issue 5**

56038 Aligned with ISO/IEC 9899:1990/Amendment 1:1995 (E). Specifically, the type of argument *wc*
 56039 is changed from **wint_t** to **wchar_t**.

56040 The Optional Header (OH) marking is removed from **<stdio.h>**.56041 **Issue 7**

56042 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0478 [14] is applied.

+

56043 **NAME**

56044 putwchar — put a wide character on a stdout stream

56045 **SYNOPSIS**

56046 #include <wchar.h>

56047 wint_t putwchar(wchar_t wc);

56048 **DESCRIPTION**

56049 CX The functionality described on this reference page is aligned with the ISO C standard. Any
56050 conflict between the requirements described here and the ISO C standard is unintentional. This
56051 volume of POSIX.1-2008 defers to the ISO C standard.

56052 The function call *putwchar(wc)* shall be equivalent to *putwc(wc,stdout)*.56053 **RETURN VALUE**56054 Refer to *fputwc()*.56055 **ERRORS**56056 Refer to *fputwc()*.56057 **EXAMPLES**

56058 None.

56059 **APPLICATION USAGE**

56060 None.

56061 **RATIONALE**

56062 None.

56063 **FUTURE DIRECTIONS**

56064 None.

56065 **SEE ALSO**56066 Section 2.5 (on page 495), *fputwc()*, *putwc()*

56067 XBD <wchar.h>

56068 **CHANGE HISTORY**

56069 First released in Issue 4.

56070 **Issue 5**

56071 Aligned with ISO/IEC 9899:1990/Amendment 1:1995 (E). Specifically, the type of argument *wc*
56072 is changed from **wint_t** to **wchar_t**.

56073 **Issue 7**

56074 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0479 [14] is applied.

+

56075 **NAME**

56076 pwrite — write on a file

56077 **SYNOPSIS**

56078 #include <unistd.h>

```
56079       ssize_t pwrite(int fildes, const void *buf, size_t nbyte,  
56080                   off_t offset);
```

56081 **DESCRIPTION**56082 Refer to *write()*.

NAME

qsort — sort a table of data

SYNOPSIS

```
#include <stdlib.h>
```

```
void qsort(void *base, size_t nel, size_t width,
           int (*compar)(const void *, const void *));
```

DESCRIPTION

CX The functionality described on this reference page is aligned with the ISO C standard. Any conflict between the requirements described here and the ISO C standard is unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

The *qsort()* function shall sort an array of *nel* objects, the initial element of which is pointed to by *base*. The size of each object, in bytes, is specified by the *width* argument. If the *nel* argument has the value zero, the comparison function pointed to by *compar* shall not be called and no rearrangement shall take place.

The application shall ensure that the comparison function pointed to by *compar* does not alter the contents of the array. The implementation may reorder elements of the array between calls to the comparison function, but shall not alter the contents of any individual element.

When the same objects (consisting of *width* bytes, irrespective of their current positions in the array) are passed more than once to the comparison function, the results shall be consistent with one another. That is, they shall define a total ordering on the array.

The contents of the array shall be sorted in ascending order according to a comparison function. The *compar* argument is a pointer to the comparison function, which is called with two arguments that point to the elements being compared. The application shall ensure that the function returns an integer less than, equal to, or greater than 0, if the first argument is considered respectively less than, equal to, or greater than the second. If two members compare as equal, their order in the sorted array is unspecified.

RETURN VALUE

The *qsort()* function shall not return a value.

ERRORS

No errors are defined.

EXAMPLES

None.

APPLICATION USAGE

The comparison function need not compare every byte, so arbitrary data may be contained in the elements in addition to the values being compared.

RATIONALE

The requirement that each argument (hereafter referred to as *p*) to the comparison function is a pointer to elements of the array implies that for every call, for each argument separately, all of the following expressions are non-zero:

```
((char *)p - (char *)base) % width == 0
(char *)p >= (char *)base
(char *)p < (char *)base + nel * width
```

56125 **FUTURE DIRECTIONS**

56126 None.

56127 **SEE ALSO**56128 *alphasort()*

56129 XBD <stdlib.h>

56130 **CHANGE HISTORY**

56131 First released in Issue 1. Derived from Issue 1 of the SVID.

56132 **Issue 6**

56133 The normative text is updated to avoid use of the term “must” for application requirements.

56134 IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/49 is applied, adding the last sentence to
56135 the first non-shaded paragraph in the DESCRIPTION, and the following two paragraphs. The
56136 RATIONALE is also updated. These changes are for alignment with the ISO C standard.

56137 **NAME**

56138 raise — send a signal to the executing process

56139 **SYNOPSIS**

56140 #include <signal.h>

56141 int raise(int sig);

56142 **DESCRIPTION**

56143 CX The functionality described on this reference page is aligned with the ISO C standard. Any
 56144 conflict between the requirements described here and the ISO C standard is unintentional. This
 56145 volume of POSIX.1-2008 defers to the ISO C standard.

56146 CX The *raise()* function shall send the signal *sig* to the executing thread or process. If a signal
 56147 handler is called, the *raise()* function shall not return until after the signal handler does.

56148 CX The effect of the *raise()* function shall be equivalent to calling:

56149 pthread_kill(pthread_self(), sig);

56150 **RETURN VALUE**

56151 CX Upon successful completion, 0 shall be returned. Otherwise, a non-zero value shall be returned
 56152 and *errno* shall be set to indicate the error.

56153 **ERRORS**

56154 The *raise()* function shall fail if:

56155 CX [EINVAL] The value of the *sig* argument is an invalid signal number.

56156 **EXAMPLES**

56157 None.

56158 **APPLICATION USAGE**

56159 None.

56160 **RATIONALE**

56161 The term “thread” is an extension to the ISO C standard.

56162 **FUTURE DIRECTIONS**

56163 None.

56164 **SEE ALSO**

56165 *kill()*, *sigaction()*

56166 XBD <signal.h>, <sys/types.h>

56167 **CHANGE HISTORY**

56168 First released in Issue 4. Derived from the ANSI C standard.

56169 **Issue 5**

56170 The DESCRIPTION is updated for alignment with the POSIX Threads Extension.

56171 **Issue 6**

56172 Extensions beyond the ISO C standard are marked.

56173 The following new requirements on POSIX implementations derive from alignment with the
 56174 Single UNIX Specification:

- 56175 • In the RETURN VALUE section, the requirement to set *errno* on error is added.

- 56176
- The [EINVAL] error condition is added.
- 56177
- Issue 7**
- 56178
- Functionality relating to the Threads option is moved to the Base.

56179 **NAME**

56180 rand, rand_r, srand — pseudo-random number generator

56181 **SYNOPSIS**

56182 #include <stdlib.h>

56183 int rand(void);

56184 OB CX int rand_r(unsigned *seed);

56185 void srand(unsigned seed);

56186 **DESCRIPTION**

56187 CX For *rand()* and *srand()*: The functionality described on this reference page is aligned with the
 56188 ISO C standard. Any conflict between the requirements described here and the ISO C standard is
 56189 unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

56190 The *rand()* function shall compute a sequence of pseudo-random integers in the range
 56191 XSI [0,{RAND_MAX}] with a period of at least 2^{32} .

56192 CX The *rand()* function need not be thread-safe.

56193 OB CX The *rand_r()* function shall compute a sequence of pseudo-random integers in the range
 56194 [0,{RAND_MAX}]. (The value of the {RAND_MAX} macro shall be at least 32767.)

56195 If *rand_r()* is called with the same initial value for the object pointed to by *seed* and that object is
 56196 not modified between successive returns and calls to *rand_r()*, the same sequence shall be
 56197 generated.

56198 The *srand()* function uses the argument as a seed for a new sequence of pseudo-random
 56199 numbers to be returned by subsequent calls to *rand()*. If *srand()* is then called with the same
 56200 seed value, the sequence of pseudo-random numbers shall be repeated. If *rand()* is called before
 56201 any calls to *srand()* are made, the same sequence shall be generated as when *srand()* is first
 56202 called with a seed value of 1.

56203 The implementation shall behave as if no function defined in this volume of POSIX.1-2008 calls
 56204 *rand()* or *srand()*.

56205 **RETURN VALUE**56206 The *rand()* function shall return the next pseudo-random number in the sequence.56207 OB CX The *rand_r()* function shall return a pseudo-random integer.56208 The *srand()* function shall not return a value.56209 **ERRORS**

56210 No errors are defined.

56211 **EXAMPLES**56212 **Generating a Pseudo-Random Number Sequence**

56213 The following example demonstrates how to generate a sequence of pseudo-random numbers.

56214 #include <stdio.h>

56215 #include <stdlib.h>

56216 ...

56217 long count, i;

56218 char *keystr;

56219 int elementlen, len;

56220 char c;

56221 ...

```

56222      /* Initial random number generator. */
56223      srand(1);

56224      /* Create keys using only lowercase characters */
56225      len = 0;
56226      for (i=0; i<count; i++) {
56227          while (len < elementlen) {
56228              c = (char) (rand() % 128);
56229              if (islower(c))
56230                  keystr[len++] = c;
56231          }

56232          keystr[len] = '\\0';
56233          printf("%s Element%0*ld\\n", keystr, elementlen, i);
56234          len = 0;
56235      }

```

Generating the Same Sequence on Different Machines

The following code defines a pair of functions that could be incorporated into applications wishing to ensure that the same sequence of numbers is generated across different machines.

```

56239      static unsigned long next = 1;
56240      int myrand(void) /* RAND_MAX assumed to be 32767. */
56241      {
56242          next = next * 1103515245 + 12345;
56243          return((unsigned) (next/65536) % 32768);
56244      }

56245      void mysrand(unsigned seed)
56246      {
56247          next = seed;
56248      }

```

APPLICATION USAGE

The *drand48()* function provides a much more elaborate random number generator.

The limitations on the amount of state that can be carried between one function call and another mean the *rand_r()* function can never be implemented in a way which satisfies all of the requirements on a pseudo-random number generator. Therefore this function should be avoided whenever non-trivial requirements (including safety) have to be fulfilled.

RATIONALE

The ISO C standard *rand()* and *srand()* functions allow per-process pseudo-random streams shared by all threads. Those two functions need not change, but there has to be mutual-exclusion that prevents interference between two threads concurrently accessing the random number generator.

With regard to *rand()*, there are two different behaviors that may be wanted in a multi-threaded program:

1. A single per-process sequence of pseudo-random numbers that is shared by all threads that call *rand()*
2. A different sequence of pseudo-random numbers for each thread that calls *rand()*

This is provided by the modified thread-safe function based on whether the seed value is global

- 56266 to the entire process or local to each thread.
- 56267 This does not address the known deficiencies of the *rand()* function implementations, which
- 56268 have been approached by maintaining more state. In effect, this specifies new thread-safe forms
- 56269 of a deficient function.
- 56270 **FUTURE DIRECTIONS**
- 56271 The *rand_r()* function may be removed in a future version.
- 56272 **SEE ALSO**
- 56273 *drand48()*
- 56274 XBD <stdlib.h>
- 56275 **CHANGE HISTORY**
- 56276 First released in Issue 1. Derived from Issue 1 of the SVID.
- 56277 **Issue 5**
- 56278 The *rand_r()* function is included for alignment with the POSIX Threads Extension.
- 56279 A note indicating that the *rand()* function need not be reentrant is added to the DESCRIPTION.
- 56280 **Issue 6**
- 56281 Extensions beyond the ISO C standard are marked.
- 56282 The *rand_r()* function is marked as part of the Thread-Safe Functions option.
- 56283 **Issue 7**
- 56284 Austin Group Interpretation 1003.1-2001 #156 is applied.
- 56285 The *rand_r()* function is marked obsolescent.

56286 **NAME**

56287 random — generate pseudo-random number

56288 **SYNOPSIS**56289 XSI `#include <stdlib.h>`56290 `long random(void);`56291 **DESCRIPTION**56292 Refer to *initstate()*.

NAME

pread, read — read from a file

SYNOPSIS

```
#include <unistd.h>
```

```
ssize_t pread(int fildes, void *buf, size_t nbyte, off_t offset);
ssize_t read(int fildes, void *buf, size_t nbyte);
```

DESCRIPTION

The *read()* function shall attempt to read *nbyte* bytes from the file associated with the open file descriptor, *fildes*, into the buffer pointed to by *buf*. The behavior of multiple concurrent reads on the same pipe, FIFO, or terminal device is unspecified.

Before any action described below is taken, and if *nbyte* is zero, the *read()* function may detect and return errors as described below. In the absence of errors, or if error detection is not performed, the *read()* function shall return zero and have no other results.

On files that support seeking (for example, a regular file), the *read()* shall start at a position in the file given by the file offset associated with *fildes*. The file offset shall be incremented by the number of bytes actually read.

Files that do not support seeking—for example, terminals—always read from the current position. The value of a file offset associated with such a file is undefined.

No data transfer shall occur past the current end-of-file. If the starting position is at or after the end-of-file, 0 shall be returned. If the file refers to a device special file, the result of subsequent *read()* requests is implementation-defined.

If the value of *nbyte* is greater than {SSIZE_MAX}, the result is implementation-defined.

When attempting to read from an empty pipe or FIFO:

- If no process has the pipe open for writing, *read()* shall return 0 to indicate end-of-file.
- If some process has the pipe open for writing and O_NONBLOCK is set, *read()* shall return -1 and set *errno* to [EAGAIN].
- If some process has the pipe open for writing and O_NONBLOCK is clear, *read()* shall block the calling thread until some data is written or the pipe is closed by all processes that had the pipe open for writing.

When attempting to read a file (other than a pipe or FIFO) that supports non-blocking reads and has no data currently available:

- If O_NONBLOCK is set, *read()* shall return -1 and set *errno* to [EAGAIN].
- If O_NONBLOCK is clear, *read()* shall block the calling thread until some data becomes available.
- The use of the O_NONBLOCK flag has no effect if there is some data available.

The *read()* function reads data previously written to a file. If any portion of a regular file prior to the end-of-file has not been written, *read()* shall return bytes with value 0. For example, *lseek()* allows the file offset to be set beyond the end of existing data in the file. If data is later written at this point, subsequent reads in the gap between the previous end of data and the newly written data shall return bytes with value 0 until data is written into the gap.

Upon successful completion, where *nbyte* is greater than 0, *read()* shall mark for update the last data access timestamp of the file, and shall return the number of bytes read. This number shall never be greater than *nbyte*. The value returned may be less than *nbyte* if the number of bytes

56336		left in the file is less than <i>nbyte</i> , if the <i>read()</i> request was interrupted by a signal, or if the file is a
56337		pipe or FIFO or special file and has fewer than <i>nbyte</i> bytes immediately available for reading.
56338		For example, a <i>read()</i> from a file associated with a terminal may return one typed line of data.
56339		If a <i>read()</i> is interrupted by a signal before it reads any data, it shall return -1 with <i>errno</i> set to
56340		[EINTR].
56341		If a <i>read()</i> is interrupted by a signal after it has successfully read some data, it shall return the
56342		number of bytes read.
56343		For regular files, no data transfer shall occur past the offset maximum established in the open
56344		file description associated with <i>filides</i> .
56345		If <i>filides</i> refers to a socket, <i>read()</i> shall be equivalent to <i>recv()</i> with no flags set.
56346	SIO	If the O_DSYNC and O_RSYNC bits have been set, read I/O operations on the file descriptor
56347		shall complete as defined by synchronized I/O data integrity completion. If the O_SYNC and
56348		O_RSYNC bits have been set, read I/O operations on the file descriptor shall complete as
56349		defined by synchronized I/O file integrity completion.
56350	SHM	If <i>filides</i> refers to a shared memory object, the result of the <i>read()</i> function is unspecified.
56351	TYM	If <i>filides</i> refers to a typed memory object, the result of the <i>read()</i> function is unspecified.
56352	OB XSR	A <i>read()</i> from a STREAMS file can read data in three different modes: <i>byte-stream</i> mode, <i>message-</i>
56353		<i>nondiscard</i> mode, and <i>message-discard</i> mode. The default shall be byte-stream mode. This can be
56354		changed using the I_SRDOPT <i>ioctl()</i> request, and can be tested with I_GRDOPT <i>ioctl()</i> . In byte-
56355		stream mode, <i>read()</i> shall retrieve data from the STREAM until as many bytes as were requested
56356		are transferred, or until there is no more data to be retrieved. Byte-stream mode ignores
56357		message boundaries.
56358		In STREAMS message-nondiscard mode, <i>read()</i> shall retrieve data until as many bytes as were
56359		requested are transferred, or until a message boundary is reached. If <i>read()</i> does not retrieve all
56360		the data in a message, the remaining data shall be left on the STREAM, and can be retrieved by
56361		the next <i>read()</i> call. Message-discard mode also retrieves data until as many bytes as were
56362		requested are transferred, or a message boundary is reached. However, unread data remaining
56363		in a message after the <i>read()</i> returns shall be discarded, and shall not be available for a
56364		subsequent <i>read()</i> , <i>getmsg()</i> , or <i>getpmsg()</i> call.
56365		How <i>read()</i> handles zero-byte STREAMS messages is determined by the current read mode
56366		setting. In byte-stream mode, <i>read()</i> shall accept data until it has read <i>nbyte</i> bytes, or until there
56367		is no more data to read, or until a zero-byte message block is encountered. The <i>read()</i> function
56368		shall then return the number of bytes read, and place the zero-byte message back on the
56369		STREAM to be retrieved by the next <i>read()</i> , <i>getmsg()</i> , or <i>getpmsg()</i> . In message-nondiscard
56370		mode or message-discard mode, a zero-byte message shall return 0 and the message shall be
56371		removed from the STREAM. When a zero-byte message is read as the first message on a
56372		STREAM, the message shall be removed from the STREAM and 0 shall be returned, regardless
56373		of the read mode.
56374		A <i>read()</i> from a STREAMS file shall return the data in the message at the front of the STREAM
56375		head read queue, regardless of the priority band of the message.
56376		By default, STREAMs are in control-normal mode, in which a <i>read()</i> from a STREAMS file can
56377		only process messages that contain a data part but do not contain a control part. The <i>read()</i> shall
56378		fail if a message containing a control part is encountered at the STREAM head. This default
56379		action can be changed by placing the STREAM in either control-data mode or control-discard
56380		mode with the I_SRDOPT <i>ioctl()</i> command. In control-data mode, <i>read()</i> shall convert any
56381		control part to data and pass it to the application before passing any data part originally present

56382 in the same message. In control-discard mode, *read()* shall discard message control parts but
 56383 return to the process any data part in the message.

56384 In addition, *read()* shall fail if the STREAM head had processed an asynchronous error before
 56385 the call. In this case, the value of *errno* shall not reflect the result of *read()*, but reflect the prior
 56386 error. If a hangup occurs on the STREAM being read, *read()* shall continue to operate normally
 56387 until the STREAM head read queue is empty. Thereafter, it shall return 0.

56388 The *pread()* function shall be equivalent to *read()*, except that it shall read from a given position
 56389 in the file without changing the file pointer. The first three arguments to *pread()* are the same as
 56390 *read()* with the addition of a fourth argument *offset* for the desired position inside the file. An
 56391 attempt to perform a *pread()* on a file that is incapable of seeking shall result in an error.

56392 RETURN VALUE

56393 Upon successful completion, these functions shall return a non-negative integer indicating the
 56394 number of bytes actually read. Otherwise, the functions shall return -1 and set *errno* to indicate
 56395 the error.

56396 ERRORS

56397 These functions shall fail if:

56398 [EAGAIN] The file is neither a pipe, nor a FIFO, nor a socket, the O_NONBLOCK flag is
 56399 set for the file descriptor, and the thread would be delayed in the read
 56400 operation.

56401 [EBADF] The *fildev* argument is not a valid file descriptor open for reading.

56402 OB XSR [EBADMSG] The file is a STREAM file that is set to control-normal mode and the message
 56403 waiting to be read includes a control part.

56404 [EINTR] The read operation was terminated due to the receipt of a signal, and no data
 56405 was transferred.

56406 OB XSR [EINVAL] The STREAM or multiplexer referenced by *fildev* is linked (directly or
 56407 indirectly) downstream from a multiplexer.

56408 [EIO] The process is a member of a background process group attempting to read
 56409 from its controlling terminal, and either the calling thread is blocking
 56410 SIGTTIN or the process is ignoring SIGTTIN or the process group of the
 56411 process is orphaned. This error may also be generated for implementation-
 56412 defined reasons.

56413 XSI [EISDIR] The *fildev* argument refers to a directory and the implementation does not
 56414 allow the directory to be read using *read()* or *pread()*. The *readdir()* function
 56415 should be used instead.

56416 [EOVERFLOW] The file is a regular file, *nbyte* is greater than 0, the starting position is before
 56417 the end-of-file, and the starting position is greater than or equal to the offset
 56418 maximum established in the open file description associated with *fildev*.

56419 The *pread()* function shall fail if: +

56420 [EINVAL] The file is a regular file or block special file, and the *offset* argument is +
 56421 negative. The file pointer shall remain unchanged. +

56422 [ESPIPE] The file is a pipe, FIFO, or socket. +

56423 The *read()* function shall fail if:

56424 [EAGAIN] The file is a pipe or FIFO, the O_NONBLOCK flag is set for the file descriptor, +
 56425 and the thread would be delayed in the read operation.

56426 [EAGAIN] or [EWOULDBLOCK]
 56427 The file is a socket, the O_NONBLOCK flag is set for the file descriptor, and |
 56428 the thread would be delayed in the read operation.

56429 [ECONNRESET] A read was attempted on a socket and the connection was forcibly closed by
 56430 its peer.

56431 [ENOTCONN] A read was attempted on a socket that is not connected.

56432 [ETIMEDOUT] A read was attempted on a socket and a transmission timeout occurred.

56433 These functions may fail if:

56434 [EIO] A physical I/O error has occurred.

56435 [ENOBUFFS] Insufficient resources were available in the system to perform the operation.

56436 [ENOMEM] Insufficient memory was available to fulfill the request.

56437 [ENXIO] A request was made of a nonexistent device, or the request was outside the
 56438 capabilities of the device.

56439 EXAMPLES

56440 Reading Data into a Buffer

56441 The following example reads data from the file associated with the file descriptor *fd* into the
 56442 buffer pointed to by *buf*.

```
56443 #include <sys/types.h>
56444 #include <unistd.h>
56445 ...
56446 char buf[20];
56447 size_t nbytes;
56448 ssize_t bytes_read;
56449 int fd;
56450 ...
56451 nbytes = sizeof(buf);
56452 bytes_read = read(fd, buf, nbytes);
56453 ...
```

56454 APPLICATION USAGE

56455 None.

56456 RATIONALE

56457 This volume of POSIX.1-2008 does not specify the value of the file offset after an error is
 56458 returned; there are too many cases. For programming errors, such as [EBADF], the concept is
 56459 meaningless since no file is involved. For errors that are detected immediately, such as
 56460 [EAGAIN], clearly the pointer should not change. After an interrupt or hardware error, however,
 56461 an updated value would be very useful and is the behavior of many implementations.

56462 Note that a *read()* of zero bytes does not modify the last data access timestamp. A *read()* that
 56463 requests more than zero bytes, but returns zero, is required to modify the last data access
 56464 timestamp.

Implementations are allowed, but not required, to perform error checking for *read()* requests of zero bytes.

Input and Output

The use of I/O with large byte counts has always presented problems. Ideas such as *lread()* and *lwrite()* (using and returning **longs**) were considered at one time. The current solution is to use abstract types on the ISO C standard function to *read()* and *write()*. The abstract types can be declared so that existing functions work, but can also be declared so that larger types can be represented in future implementations. It is presumed that whatever constraints limit the maximum range of **size_t** also limit portable I/O requests to the same range. This volume of POSIX.1-2008 also limits the range further by requiring that the byte count be limited so that a signed return value remains meaningful. Since the return type is also a (signed) abstract type, the byte count can be defined by the implementation to be larger than an **int** can hold.

The standard developers considered adding atomicity requirements to a pipe or FIFO, but recognized that due to the nature of pipes and FIFOs there could be no guarantee of atomicity of reads of {PIPE_BUF} or any other size that would be an aid to applications portability.

This volume of POSIX.1-2008 requires that no action be taken for *read()* or *write()* when *nbyte* is zero. This is not intended to take precedence over detection of errors (such as invalid buffer pointers or file descriptors). This is consistent with the rest of this volume of POSIX.1-2008, but the phrasing here could be misread to require detection of the zero case before any other errors. A value of zero is to be considered a correct value, for which the semantics are a no-op.

I/O is intended to be atomic to ordinary files and pipes and FIFOs. Atomic means that all the bytes from a single operation that started out together end up together, without interleaving from other I/O operations. It is a known attribute of terminals that this is not honored, and terminals are explicitly (and implicitly permanently) excepted, making the behavior unspecified. The behavior for other device types is also left unspecified, but the wording is intended to imply that future standards might choose to specify atomicity (or not).

There were recommendations to add format parameters to *read()* and *write()* in order to handle networked transfers among heterogeneous file system and base hardware types. Such a facility may be required for support by the OSI presentation of layer services. However, it was determined that this should correspond with similar C-language facilities, and that is beyond the scope of this volume of POSIX.1-2008. The concept was suggested to the developers of the ISO C standard for their consideration as a possible area for future work.

In 4.3 BSD, a *read()* or *write()* that is interrupted by a signal before transferring any data does not by default return an [EINTR] error, but is restarted. In 4.2 BSD, 4.3 BSD, and the Eighth Edition, there is an additional function, *select()*, whose purpose is to pause until specified activity (data to read, space to write, and so on) is detected on specified file descriptors. It is common in applications written for those systems for *select()* to be used before *read()* in situations (such as keyboard input) where interruption of I/O due to a signal is desired.

The issue of which files or file types are interruptible is considered an implementation design issue. This is often affected primarily by hardware and reliability issues.

There are no references to actions taken following an “unrecoverable error”. It is considered beyond the scope of this volume of POSIX.1-2008 to describe what happens in the case of hardware errors.

Earlier versions of this standard allowed two very different behaviors with regard to the handling of interrupts. In order to minimize the resulting confusion, it was decided that POSIX.1-2008 should support only one of these behaviors. Historical practice on AT&T-derived

systems was to have *read()* and *write()* return `-1` and set *errno* to `[EINTR]` when interrupted after some, but not all, of the data requested had been transferred. However, the US Department of Commerce FIPS 151-1 and FIPS 151-2 require the historical BSD behavior, in which *read()* and *write()* return the number of bytes actually transferred before the interrupt. If `-1` is returned when any data is transferred, it is difficult to recover from the error on a seekable device and impossible on a non-seekable device. Most new implementations support this behavior. The behavior required by POSIX.1-2008 is to return the number of bytes transferred.

POSIX.1-2008 does not specify when an implementation that buffers *read()*s actually moves the data into the user-supplied buffer, so an implementation may choose to do this at the latest possible moment. Therefore, an interrupt arriving earlier may not cause *read()* to return a partial byte count, but rather to return `-1` and set *errno* to `[EINTR]`.

Consideration was also given to combining the two previous options, and setting *errno* to `[EINTR]` while returning a short count. However, not only is there no existing practice that implements this, it is also contradictory to the idea that when *errno* is set, the function responsible shall return `-1`.

This volume of POSIX.1-2008 intentionally does not specify any *pread()* errors related to pipes, + FIFOs, and sockets other than `[ESPIPE]`.

FUTURE DIRECTIONS

None.

SEE ALSO

fcntl(), *ioctl()*, *lseek()*, *open()*, *pipe()*, *readv()*

XBD Chapter 11 (on page 199), `<stropts.h>`, `<sys/uio.h>`, `<unistd.h>`

CHANGE HISTORY

First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 5

The DESCRIPTION is updated for alignment with the POSIX Realtime Extension and the POSIX Threads Extension.

Large File Summit extensions are added.

The *pread()* function is added.

Issue 6

The DESCRIPTION and ERRORS sections are updated so that references to STREAMS are marked as part of the XSI STREAMS Option Group.

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The DESCRIPTION now states that if *read()* is interrupted by a signal after it has successfully read some data, it returns the number of bytes read. In Issue 3, it was optional whether *read()* returned the number of bytes read, or whether it returned `-1` with *errno* set to `[EINTR]`. This is a FIPS requirement.
- In the DESCRIPTION, text is added to indicate that for regular files, no data transfer occurs past the offset maximum established in the open file description associated with *files*. This change is to support large files.
- The `[EOVERFLOW]` mandatory error condition is added.

- The [ENXIO] optional error condition is added.

Text referring to sockets is added to the DESCRIPTION.

The following changes were made to align with the IEEE P1003.1a draft standard:

- The effect of reading zero bytes is clarified.

The DESCRIPTION is updated for alignment with IEEE Std 1003.1j-2000 by specifying that *read()* results are unspecified for typed memory objects.

New RATIONALE is added to explain the atomicity requirements for input and output operations.

The following error conditions are added for operations on sockets: [EAGAIN], [ECONNRESET], [ENOTCONN], and [ETIMEDOUT].

The [EIO] error is made optional.

The following error conditions are added for operations on sockets: [ENOBUFS] and [ENOMEM].

The *readv()* function is split out into a separate reference page.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/108 is applied, updating the [EAGAIN] error in the ERRORS section from “the process would be delayed” to “the thread would be delayed”.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/109 is applied, making an editorial correction in the RATIONALE section.

Issue 7

The *pread()* function is moved from the XSI option to the Base.

Functionality relating to the XSI STREAMS option is marked obsolescent.

Changes are made related to support for finegrained timestamps.

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0480 [218], XSH/TC1-2008/0481 [79], + XSH/TC1-2008/0482 [218], XSH/TC1-2008/0483 [218], XSH/TC1-2008/0484 [218], and + XSH/TC1-2008/0485 [218,428] are applied.

NAME

readdir, readdir_r — read a directory

SYNOPSIS

```
#include <dirent.h>

struct dirent *readdir(DIR *dirp);
int readdir_r(DIR *restrict dirp, struct dirent *restrict entry,
              struct dirent **restrict result);
```

DESCRIPTION

The type **DIR**, which is defined in the **<dirent.h>** header, represents a *directory stream*, which is an ordered sequence of all the directory entries in a particular directory. Directory entries represent files; files may be removed from a directory or added to a directory asynchronously to the operation of *readdir()*.

The *readdir()* function shall return a pointer to a structure representing the directory entry at the current position in the directory stream specified by the argument *dirp*, and position the directory stream at the next entry. It shall return a null pointer upon reaching the end of the directory stream. The structure **dirent** defined in the **<dirent.h>** header describes a directory entry. The value of the structure's *d_ino* member shall be set to the file serial number of the file named by the *d_name* member. If the *d_name* member names a symbolic link, the value of the *d_ino* member shall be set to the file serial number of the symbolic link itself.

The *readdir()* function shall not return directory entries containing empty names. If entries for dot or dot-dot exist, one entry shall be returned for dot and one entry shall be returned for dot-dot; otherwise, they shall not be returned.

The application shall not modify the structure to which the return value of *readdir()* points, nor any storage areas pointed to by pointers within the structure. The returned pointer, and pointers within the structure, might be invalidated or the structure or the storage areas might be overwritten by a subsequent call to *readdir()* on the same directory stream. They shall not be affected by a call to *readdir()* on a different directory stream.

If a file is removed from or added to the directory after the most recent call to *opendir()* or *rewinddir()*, whether a subsequent call to *readdir()* returns an entry for that file is unspecified.

The *readdir()* function may buffer several directory entries per actual read operation; *readdir()* shall mark for update the last data access timestamp of the directory each time the directory is actually read.

After a call to *fork()*, either the parent or child (but not both) may continue processing the directory stream using *readdir()*, *rewinddir()*, or *seekdir()*. If both the parent and child processes use these functions, the result is undefined.

The *readdir()* function need not be thread-safe.

Applications wishing to check for error situations should set *errno* to 0 before calling *readdir()*. If *errno* is set to non-zero on return, an error occurred.

The *readdir_r()* function shall initialize the **dirent** structure referenced by *entry* to represent the directory entry at the current position in the directory stream referred to by *dirp*, store a pointer to this structure at the location referenced by *result*, and position the directory stream at the next entry.

The storage pointed to by *entry* shall be large enough for a **dirent** with an array of **char** *d_name* members containing at least {NAME_MAX}+1 elements.

Upon successful return, the pointer returned at **result* shall have the same value as the argument

entry. Upon reaching the end of the directory stream, this pointer shall have the value NULL.

The *readdir_r()* function shall not return directory entries containing empty names.

If a file is removed from or added to the directory after the most recent call to *opendir()* or *rewinddir()*, whether a subsequent call to *readdir_r()* returns an entry for that file is unspecified.

The *readdir_r()* function may buffer several directory entries per actual read operation; *readdir_r()* shall mark for update the last data access timestamp of the directory each time the directory is actually read.

RETURN VALUE

Upon successful completion, *readdir()* shall return a pointer to an object of type **struct dirent**. When an error is encountered, a null pointer shall be returned and *errno* shall be set to indicate the error. When the end of the directory is encountered, a null pointer shall be returned and *errno* is not changed.

If successful, the *readdir_r()* function shall return zero; otherwise, an error number shall be returned to indicate the error.

ERRORS

These functions shall fail if:

[EOVERFLOW] One of the values in the structure to be returned cannot be represented correctly.

These functions may fail if:

[EBADF] The *dirp* argument does not refer to an open directory stream.

[ENOENT] The current position of the directory stream is invalid.

EXAMPLES

The following sample program searches the current directory for each of the arguments supplied on the command line.

```
#include <dirent.h>
#include <errno.h>
#include <stdio.h>
#include <string.h>

static void lookup(const char *arg)
{
    DIR *dirp;
    struct dirent *dp;

    if ((dirp = opendir(".")) == NULL) {
        perror("couldn't open '.');
        return;
    }

    do {
        errno = 0;
        if ((dp = readdir(dirp)) != NULL) {
            if (strcmp(dp->d_name, arg) != 0)
                continue;

            (void) printf("found %s\n", arg);
            (void) closedir(dirp);
            return;
        }
    } while (1);
}
```

```

56668         }
56669     } while (dp != NULL);
56670
56671     if (errno != 0)
56672         perror("error reading directory");
56673     else
56674         (void) printf("failed to find %s\n", arg);
56675     (void) closedir(dirp);
56676     return;
56677 }
56678
56679 int main(int argc, char *argv[])
56680 {
56681     int i;
56682     for (i = 1; i < argc; i++)
56683         lookup(argv[i]);
56684     return (0);
56685 }

```

APPLICATION USAGE

The *readdir()* function should be used in conjunction with *opendir()*, *closedir()*, and *rewinddir()* to examine the contents of the directory.

The *readdir_r()* function is thread-safe and shall return values in a user-supplied buffer instead of possibly using a static data area that may be overwritten by each call.

RATIONALE

The returned value of *readdir()* merely *represents* a directory entry. No equivalence should be inferred.

Historical implementations of *readdir()* obtain multiple directory entries on a single read operation, which permits subsequent *readdir()* operations to operate from the buffered information. Any wording that required each successful *readdir()* operation to mark the directory last data access timestamp for update would disallow such historical performance-oriented implementations.

When returning a directory entry for the root of a mounted file system, some historical implementations of *readdir()* returned the file serial number of the underlying mount point, rather than of the root of the mounted file system. This behavior is considered to be a bug, since the underlying file serial number has no significance to applications.

Since *readdir()* returns NULL when it detects an error and when the end of the directory is encountered, an application that needs to tell the difference must set *errno* to zero before the call and check it if NULL is returned. Since the function must not change *errno* in the second case and must set it to a non-zero value in the first case, a zero *errno* after a call returning NULL indicates end-of-directory; otherwise, an error.

Routines to deal with this problem more directly were proposed:

```

56707 int derror (dirp)
56708 DIR *dirp;
56709
56710 void clearerr (dirp)
56711 DIR *dirp;

```

The first would indicate whether an error had occurred, and the second would clear the error indication. The simpler method involving *errno* was adopted instead by requiring that *readdir()* not change *errno* when end-of-directory is encountered.

56714 An error or signal indicating that a directory has changed while open was considered but
56715 rejected.

56716 The thread-safe version of the directory reading function returns values in a user-supplied buffer
56717 instead of possibly using a static data area that may be overwritten by each call. Either the
56718 {NAME_MAX} compile-time constant or the corresponding *pathconf()* option can be used to
56719 determine the maximum sizes of returned pathnames.

56720 FUTURE DIRECTIONS

56721 None.

56722 SEE ALSO

56723 *closedir()*, *dirfd()*, *exec*, *fdopendir()*, *fstatat()*, *rewinddir()*, *symlink()*

56724 XBD <dirent.h>, <sys/types.h>

56725 CHANGE HISTORY

56726 First released in Issue 2.

56727 Issue 5

56728 Large File Summit extensions are added.

56729 The *readdir_r()* function is included for alignment with the POSIX Threads Extension.

56730 A note indicating that the *readdir()* function need not be reentrant is added to the
56731 DESCRIPTION.

56732 Issue 6

56733 The *readdir_r()* function is marked as part of the Thread-Safe Functions option.

56734 The Open Group Corrigendum U026/7 is applied, correcting the prototype for *readdir_r()*.

56735 The Open Group Corrigendum U026/8 is applied, clarifying the wording of the successful
56736 return for the *readdir_r()* function.

56737 The following new requirements on POSIX implementations derive from alignment with the
56738 Single UNIX Specification:

- 56739 • The requirement to include <sys/types.h> has been removed. Although <sys/types.h> was
56740 required for conforming implementations of previous POSIX specifications, it was not
56741 required for UNIX applications.
- 56742 • A statement is added to the DESCRIPTION indicating the disposition of certain fields in
56743 **struct dirent** when an entry refers to a symbolic link.
- 56744 • The [EOVERFLOW] mandatory error condition is added. This change is to support large
56745 files.
- 56746 • The [ENOENT] optional error condition is added.

56747 The APPLICATION USAGE section is updated to include a note on the thread-safe function and
56748 its avoidance of possibly using a static data area.

56749 The **restrict** keyword is added to the *readdir_r()* prototype for alignment with the
56750 ISO/IEC 9899:1999 standard.

56751 IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/50 is applied, replacing the EXAMPLES
56752 section with a new example.

56753 **Issue 7**

56754 Austin Group Interpretation 1003.1-2001 #059 is applied, updating the ERRORS section.

56755 Austin Group Interpretation 1003.1-2001 #156 is applied.

56756 The *readdir_r()* function is moved from the Thread-Safe Functions option to the Base.

56757 Changes are made related to support for finegrained timestamps.

56758 The value of the *d_ino* member is no longer unspecified for symbolic links.

56759 SD5-XSH-ERN-193 is applied.

56760 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0486 [75] is applied.

+

NAME

readlink, readlinkat — read the contents of a symbolic link

SYNOPSIS

```
#include <unistd.h>

ssize_t readlink(const char *restrict path, char *restrict buf,
                 size_t bufsize);
ssize_t readlinkat(int fd, const char *restrict path,
                  char *restrict buf, size_t bufsize);
```

DESCRIPTION

The *readlink()* function shall place the contents of the symbolic link referred to by *path* in the buffer *buf* which has size *bufsize*. If the number of bytes in the symbolic link is less than *bufsize*, the contents of the remainder of *buf* are unspecified. If the *buf* argument is not large enough to contain the link content, the first *bufsize* bytes shall be placed in *buf*.

If the value of *bufsize* is greater than {SSIZE_MAX}, the result is implementation-defined.

Upon successful completion, *readlink()* shall mark for update the last data access timestamp of the symbolic link.

The *readlinkat()* function shall be equivalent to the *readlink()* function except in the case where *path* specifies a relative path. In this case the symbolic link whose content is read is relative to the directory associated with the file descriptor *fd* instead of the current working directory. If the file descriptor was opened without O_SEARCH, the function shall check whether directory searches are permitted using the current permissions of the directory underlying the file descriptor. If the file descriptor was opened with O_SEARCH, the function shall not perform the check.

If *readlinkat()* is passed the special value AT_FDCWD in the *fd* parameter, the current working directory shall be used and the behavior shall be identical to a call to *readlink()*.

RETURN VALUE

Upon successful completion, these functions shall return the count of bytes placed in the buffer. Otherwise, these functions shall return a value of -1, leave the buffer unchanged, and set *errno* to indicate the error.

ERRORS

These functions shall fail if:

[EACCES]	Search permission is denied for a component of the path prefix of <i>path</i> .
[EINVAL]	The <i>path</i> argument names a file that is not a symbolic link.
[EIO]	An I/O error occurred while reading from the file system.
[ELOOP]	A loop exists in symbolic links encountered during resolution of the <i>path</i> argument.
[ENAMETOOLONG]	The length of a component of a pathname is longer than {NAME_MAX}.
[ENOENT]	A component of <i>path</i> does not name an existing file or <i>path</i> is an empty string.
[ENOTDIR]	A component of the path prefix names an existing file that is neither a directory nor a symbolic link to a directory, or the <i>path</i> argument contains at least one non- <i><slash></i> character and ends with one or more trailing <i><slash></i> characters and the last pathname component names an existing file that is neither a directory nor a symbolic link to a directory.

56804 The *readlinkat()* function shall fail if:

56805 [EACCES] *fd* was not opened with O_SEARCH and the permissions of the directory
56806 underlying *fd* do not permit directory searches.

56807 [EBADF] The *path* argument does not specify an absolute path and the *fd* argument is
56808 neither AT_FDCWD nor a valid file descriptor open for reading or searching.

56809 [ENOTDIR] The *path* argument is not an absolute path and *fd* is a file descriptor associated +
56810 with a non-directory file.

56811 These functions may fail if:

56812 [ELOOP] More than {SYMLOOP_MAX} symbolic links were encountered during
56813 resolution of the *path* argument.

56814 [ENAMETOOLONG]
56815 The length of a pathname exceeds {PATH_MAX}, or pathname resolution of a
56816 symbolic link produced an intermediate result with a length that exceeds
56817 {PATH_MAX}.

56818 EXAMPLES

56819 Reading the Name of a Symbolic Link

56820 The following example shows how to read the name of a symbolic link named */modules/pass1*.

```
56821 #include <unistd.h>
56822 char buf[1024];
56823 ssize_t len;
56824 ...
56825 if ((len = readlink("/modules/pass1", buf, sizeof(buf)-1)) != -1)
56826     buf[len] = '\0';
```

56827 APPLICATION USAGE

56828 Confirming applications should not assume that the returned contents of the symbolic link are
56829 null-terminated.

56830 RATIONALE

56831 The type associated with *bufsiz* is a **size_t** in order to be consistent with both the ISO C standard -
56832 and the definition of *read()*. The behavior specified for *readlink()* when *bufsiz* is zero represents
56833 historical practice. For this case, the standard developers considered a change whereby *readlink()*
56834 would return the number of non-null bytes contained in the symbolic link with the buffer *buf*
56835 remaining unchanged; however, since the **stat** structure member *st_size* value can be used to
56836 determine the size of buffer necessary to contain the contents of the symbolic link as returned by
56837 *readlink()*, this proposal was rejected, and the historical practice retained.

56838 The purpose of the *readlinkat()* function is to read the content of symbolic links in directories
56839 other than the current working directory without exposure to race conditions. Any part of the
56840 path of a file could be changed in parallel to a call to *readlink()*, resulting in unspecified behavior.
56841 By opening a file descriptor for the target directory and using the *readlinkat()* function it can be
56842 guaranteed that the symbolic link read is located relative to the desired directory.

56843 FUTURE DIRECTIONS

56844 None.

SEE ALSO*fstatat()*, *symlink()*

XBD <unistd.h>

CHANGE HISTORY

First released in Issue 4, Version 2.

Issue 5

Moved from X/OPEN UNIX extension to BASE.

Issue 6The return type is changed to **ssize_t**, to align with the IEEE P1003.1a draft standard.

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- This function is made mandatory.
- In this function it is possible for the return value to exceed the range of the type **ssize_t** (since **size_t** has a larger range of positive values than **ssize_t**). A sentence restricting the size of the **size_t** object is added to the description to resolve this conflict.

The following changes are made for alignment with the ISO POSIX-1: 1996 standard:

- The FUTURE DIRECTIONS section is changed to None.

The following changes were made to align with the IEEE P1003.1a draft standard:

- The [ELOOP] optional error condition is added.

The **restrict** keyword is added to the *readlink()* prototype for alignment with the ISO/IEC 9899: 1999 standard.**Issue 7**

Austin Group Interpretation 1003.1-2001 #143 is applied.

SD5-XSH-ERN-189 is applied, updating the ERRORS section.

The *readlinkat()* function is added from The Open Group Technical Standard, 2006, Extended API Set Part 2.

The [EACCES] error is removed from the “may fail” error conditions.

Changes are made to allow a directory to be opened for searching.

The [ENOTDIR] error condition is clarified to cover the condition where the last component of a pathname exists but is not a directory or a symbolic link to a directory.

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0487 [120], XSH/TC1-2008/0488 [461], +
 XSH/TC1-2008/0489 [143], XSH/TC1-2008/0490 [324], XSH/TC1-2008/0491 [278], +
 XSH/TC1-2008/0492 [278], XSH/TC1-2008/0493 [455], and XSH/TC1-2008/0494 [151,231] are +
 applied.

NAME

readv — read a vector

SYNOPSIS

```
XSI      #include <sys/uio.h>
        ssize_t readv(int fildes, const struct iovec *iov, int iovcnt);
```

DESCRIPTION

The *readv()* function shall be equivalent to *read()*, except as described below. The *readv()* function shall place the input data into the *iovcnt* buffers specified by the members of the *iov* array: *iov*[0], *iov*[1], ..., *iov*[*iovcnt*−1]. The *iovcnt* argument is valid if greater than 0 and less than or equal to {IOV_MAX}.

Each *iovec* entry specifies the base address and length of an area in memory where data should be placed. The *readv()* function shall always fill an area completely before proceeding to the next.

Upon successful completion, *readv()* shall mark for update the last data access timestamp of the file.

RETURN VALUE

Refer to *read()*.

ERRORS

Refer to *read()*.

In addition, the *readv()* function shall fail if:

[EINVAL] The sum of the *iov_len* values in the *iov* array overflowed an *ssize_t*.

The *readv()* function may fail if:

[EINVAL] The *iovcnt* argument was less than or equal to 0, or greater than {IOV_MAX}.

EXAMPLES**Reading Data into an Array**

The following example reads data from the file associated with the file descriptor *fd* into the buffers specified by members of the *iov* array.

```
#include <sys/types.h>
#include <sys/uio.h>
#include <unistd.h>
...
ssize_t bytes_read;
int fd;
char buf0[20];
char buf1[30];
char buf2[40];
int iovcnt;
struct iovec iov[3];

iov[0].iov_base = buf0;
iov[0].iov_len = sizeof(buf0);
iov[1].iov_base = buf1;
iov[1].iov_len = sizeof(buf1);
iov[2].iov_base = buf2;
```

```
56922         iov[2].iov_len = sizeof(buf2);
56923         ...
56924         iovcnt = sizeof(iov) / sizeof(struct iovec);
56925         bytes_read = readv(fd, iov, iovcnt);
56926         ...
```

56927 **APPLICATION USAGE**
56928 None.

56929 **RATIONALE**
56930 Refer to *read()*.

56931 **FUTURE DIRECTIONS**
56932 None.

56933 **SEE ALSO**
56934 *read()*, *writev()*
56935 XBD <sys/uio.h>

56936 **CHANGE HISTORY**
56937 First released in Issue 4, Version 2.

56938 **Issue 6**
56939 Split out from the *read()* reference page.

56940 **Issue 7**
56941 Changes are made related to support for finegrained timestamps.

56942 **NAME**

56943 realloc — memory reallocator

56944 **SYNOPSIS**

56945 #include <stdlib.h>

56946 void *realloc(void *ptr, size_t size);

56947 **DESCRIPTION**

56948 CX The functionality described on this reference page is aligned with the ISO C standard. Any
 56949 conflict between the requirements described here and the ISO C standard is unintentional. This
 56950 volume of POSIX.1-2008 defers to the ISO C standard.

56951 The *realloc()* function shall deallocate the old object pointed to by *ptr* and return a pointer to a
 56952 new object that has the size specified by *size*. The contents of the new object shall be the same as
 56953 that of the old object prior to deallocation, up to the lesser of the new and old sizes. Any bytes in
 56954 the new object beyond the size of the old object have indeterminate values. If the size of the
 56955 space requested is zero, the behavior shall be implementation-defined: either a null pointer is
 56956 returned, or the behavior shall be as if the size were some non-zero value, except that the
 56957 returned pointer shall not be used to access an object. If the space cannot be allocated, the object
 56958 shall remain unchanged.

56959 If *ptr* is a null pointer, *realloc()* shall be equivalent to *malloc()* for the specified size.

56960 If *ptr* does not match a pointer returned earlier by *calloc()*, *malloc()*, or *realloc()* or if the space has
 56961 previously been deallocated by a call to *free()* or *realloc()*, the behavior is undefined.

56962 The order and contiguity of storage allocated by successive calls to *realloc()* is unspecified. The
 56963 pointer returned if the allocation succeeds shall be suitably aligned so that it may be assigned to
 56964 a pointer to any type of object and then used to access such an object in the space allocated (until
 56965 the space is explicitly freed or reallocated). Each such allocation shall yield a pointer to an object
 56966 disjoint from any other object. The pointer returned shall point to the start (lowest byte address)
 56967 of the allocated space. If the space cannot be allocated, a null pointer shall be returned.

56968 **RETURN VALUE**

56969 Upon successful completion, *realloc()* shall return a pointer to the (possibly moved) allocated
 56970 space. If *size* is 0, either:

- 56971 CX • A null pointer shall be returned and *errno* set to an implementation-defined value.
- 56972 • A unique pointer that can be successfully passed to *free()* shall be returned, and the
 56973 memory object pointed to by *ptr* shall be freed. The application shall ensure that the
 56974 pointer is not used to access an object.

56975 CX If there is not enough available memory, *realloc()* shall return a null pointer and set *errno* to
 56976 CX [ENOMEM]. If *realloc()* returns a null pointer and *errno* has been set to [ENOMEM], the
 56977 memory referenced by *ptr* shall not be changed. +

56978 **ERRORS**

56979 The *realloc()* function shall fail if:

56980 CX [ENOMEM] Insufficient memory is available.

EXAMPLES

None.

APPLICATION USAGE

The description of `realloc()` has been modified from previous versions of this standard to align with the ISO/IEC 9899:1999 standard. Previous versions explicitly permitted a call to `realloc(p, 0)` to free the space pointed to by `p` and return a null pointer. While this behavior could be interpreted as permitted by this version of the standard, the C language committee have indicated that this interpretation is incorrect. Applications should assume that if `realloc()` returns a null pointer, the space pointed to by `p` has not been freed. Since this could lead to double-frees, implementations should also set `errno` if a null pointer actually indicates a failure, and applications should only free the space if `errno` was changed.

RATIONALE

None.

FUTURE DIRECTIONS

This standard defers to the ISO C standard. While that standard currently has language that might permit `realloc(p, 0)`, where `p` is not a null pointer, to free `p` while still returning a null pointer, the committee responsible for that standard is considering clarifying the language to explicitly prohibit that alternative.

SEE ALSO

`calloc()`, `free()`, `malloc()`

XBD `<stdlib.h>`

CHANGE HISTORY

First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6

Extensions beyond the ISO C standard are marked.

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- In the RETURN VALUE section, if there is not enough available memory, the setting of `errno` to [ENOMEM] is added.
- The [ENOMEM] error condition is added.

Issue 7

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0495 [400], XSH/TC1-2008/0496 [400], XSH/TC1-2008/0497 [400], and XSH/TC1-2008/0498 [400] are applied.

57013 **NAME**57014 `realpath` — resolve a pathname57015 **SYNOPSIS**

```
57016 XSI      #include <stdlib.h>
57017
57017      char *realpath(const char *restrict file_name,
57018                    char *restrict resolved_name);
```

57019 **DESCRIPTION**

57020 The `realpath()` function shall derive, from the pathname pointed to by *file_name*, an absolute
 57021 pathname that resolves to the same directory entry, whose resolution does not involve '.',
 57022 '..', or symbolic links. If *resolved_name* is a null pointer, the generated pathname shall be
 57023 stored as a null-terminated string in a buffer allocated as if by a call to `malloc()`. Otherwise, if
 57024 {PATH_MAX} is defined as a constant in the `<limits.h>` header, then the generated pathname
 57025 shall be stored as a null-terminated string, up to a maximum of {PATH_MAX} bytes, in the
 57026 buffer pointed to by *resolved_name*.

57027 If *resolved_name* is not a null pointer and {PATH_MAX} is not defined as a constant in the
 57028 `<limits.h>` header, the behavior is undefined.

57029 **RETURN VALUE**

57030 Upon successful completion, `realpath()` shall return a pointer to the buffer containing the
 57031 resolved name. Otherwise, `realpath()` shall return a null pointer and set *errno* to indicate the
 57032 error.

57033 If the *resolved_name* argument is a null pointer, the pointer returned by `realpath()` can be passed
 57034 to `free()`.

57035 If the *resolved_name* argument is not a null pointer and the `realpath()` function fails, the contents
 57036 of the buffer pointed to by *resolved_name* are undefined.

57037 **ERRORS**

57038 The `realpath()` function shall fail if:

57039	[EACCES]	Search permission was denied for a component of the path prefix of <i>file_name</i> .	
57040	[EINVAL]	The <i>file_name</i> argument is a null pointer.	
57041	[EIO]	An error occurred while reading from the file system.	
57042	[ELOOP]	A loop exists in symbolic links encountered during resolution of the <i>file_name</i> argument.	
57043			
57044	[ENAMETOOLONG]		
57045		The length of a component of a pathname is longer than {NAME_MAX}.	
57046	[ENOENT]	A component of <i>file_name</i> does not name an existing file or <i>file_name</i> points to an empty string.	
57047			
57048	[ENOTDIR]	A component of the path prefix names an existing file that is neither a	
57049		directory nor a symbolic link to a directory, or the <i>file_name</i> argument contains	
57050		at least one non- <code><slash></code> character and ends with one or more trailing <code><slash></code>	
57051		characters and the last pathname component names an existing file that is	
57052		neither a directory nor a symbolic link to a directory.	

57053 The *realpath()* function may fail if:

57054 [EACCES] The *file_name* argument does not begin with a <slash> and none of the +
 57055 symbolic links (if any) processed during pathname resolution of *file_name* had +
 57056 contents that began with a <slash>, and either search permission was denied +
 57057 for the current directory or read or search permission was denied for a +
 57058 directory above the current directory in the file hierarchy.

57059 [ELOOP] More than {SYMLOOP_MAX} symbolic links were encountered during
 57060 resolution of the *file_name* argument.

57061 [ENAMETOOLONG]
 57062 The length of a pathname exceeds {PATH_MAX}, or pathname resolution of a
 57063 symbolic link produced an intermediate result with a length that exceeds
 57064 {PATH_MAX}.

57065 [ENOMEM] Insufficient storage space is available.

57066 EXAMPLES

57067 Generating an Absolute Pathname

57068 The following example generates an absolute pathname for the file identified by the *symlinkpath*
 57069 argument. The generated pathname is stored in the buffer pointed to by *actualpath*.

```
57070 #include <stdlib.h>
57071 ...
57072 char *symlinkpath = "/tmp/symlink/file";
57073 char *actualpath;

57074 actualpath = realpath(symlinkpath, NULL);
57075 if (actualpath != NULL)
57076 {
57077     ... use actualpath ...

57078     free(actualpath);
57079 }
57080 else
57081 {
57082     ... handle error ...
57083 }
```

57084 APPLICATION USAGE

57085 For functions that allocate memory as if by *malloc()*, the application should release such memory
 57086 when it is no longer required by a call to *free()*. For *realpath()*, this is the return value.

57087 RATIONALE

57088 Since *realpath()* has no *length* argument, if {PATH_MAX} is not defined as a constant in
 57089 <limits.h>, applications have no way of determining how large a buffer they need to allocate for
 57090 it to be safe to pass to *realpath()*. A {PATH_MAX} value obtained from a prior *pathconf()* call is
 57091 out-of-date by the time *realpath()* is called. Hence the only reliable way to use *realpath()* when
 57092 {PATH_MAX} is not defined in <limits.h> is to pass a null pointer for *resolved_name* so that
 57093 *realpath()* will allocate a buffer of the necessary size.

57094 **FUTURE DIRECTIONS**

57095 None.

57096 **SEE ALSO**57097 *fpathconf()*, *free()*, *getcwd()*, *sysconf()*57098 XBD **<limits.h>**, **<stdlib.h>**57099 **CHANGE HISTORY**

57100 First released in Issue 4, Version 2.

57101 **Issue 5**

57102 Moved from X/OPEN UNIX extension to BASE.

57103 **Issue 6**57104 The **restrict** keyword is added to the *realpath()* prototype for alignment with the
57105 ISO/IEC 9899:1999 standard.57106 The wording of the mandatory [ELOOP] error condition is updated, and a second optional
57107 [ELOOP] error condition is added.57108 IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/51 is applied, adding new text to the
57109 DESCRIPTION for the case when *resolved_name* is a null pointer, changing the [EINVAL] error
57110 text, adding text to the RATIONALE, and adding text to FUTURE DIRECTIONS.57111 IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/110 is applied, updating the ERRORS
57112 section to refer to the *file_name* argument, rather than a nonexistent *path* argument.57113 **Issue 7**

57114 Austin Group Interpretation 1003.1-2001 #143 is applied.

57115 This function is updated for passing a null pointer to *realpath()* for the *resolved_name* argument.57116 The APPLICATION USAGE section is updated to clarify that memory is allocated as if by
57117 *malloc()*.57118 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0499 [353], XSH/TC1-2008/0500 [324], +
57119 and XSH/TC1-2008/0501 [353] are applied.

NAME

recv — receive a message from a connected socket

SYNOPSIS

```
#include <sys/socket.h>
```

```
ssize_t recv(int socket, void *buffer, size_t length, int flags);
```

DESCRIPTION

The *recv()* function shall receive a message from a connection-mode or connectionless-mode socket. It is normally used with connected sockets because it does not permit the application to retrieve the source address of received data.

The *recv()* function takes the following arguments:

socket Specifies the socket file descriptor.

buffer Points to a buffer where the message should be stored.

length Specifies the length in bytes of the buffer pointed to by the *buffer* argument.

flags Specifies the type of message reception. Values of this argument are formed by logically OR'ing zero or more of the following values:

MSG_PEEK Peeks at an incoming message. The data is treated as unread and the next *recv()* or similar function shall still return this data.

MSG_OOB Requests out-of-band data. The significance and semantics of out-of-band data are protocol-specific.

MSG_WAITALL On SOCK_STREAM sockets this requests that the function block until the full amount of data can be returned. The function may return the smaller amount of data if the socket is a message-based socket, if a signal is caught, if the connection is terminated, if MSG_PEEK was specified, or if an error is pending for the socket.

The *recv()* function shall return the length of the message written to the buffer pointed to by the *buffer* argument. For message-based sockets, such as SOCK_DGRAM and SOCK_SEQPACKET, the entire message shall be read in a single operation. If a message is too long to fit in the supplied buffer, and MSG_PEEK is not set in the *flags* argument, the excess bytes shall be discarded. For stream-based sockets, such as SOCK_STREAM, message boundaries shall be ignored. In this case, data shall be returned to the user as soon as it becomes available, and no data shall be discarded.

If the MSG_WAITALL flag is not set, data shall be returned only up to the end of the first message.

If no messages are available at the socket and O_NONBLOCK is not set on the socket's file descriptor, *recv()* shall block until a message arrives. If no messages are available at the socket and O_NONBLOCK is set on the socket's file descriptor, *recv()* shall fail and set *errno* to [EAGAIN] or [EWOULDBLOCK].

RETURN VALUE

Upon successful completion, *recv()* shall return the length of the message in bytes. If no messages are available to be received and the peer has performed an orderly shutdown, *recv()* shall return 0. Otherwise, -1 shall be returned and *errno* set to indicate the error.

ERRORS

The *recv()* function shall fail if:

[EAGAIN] or [EWOULDBLOCK]

The socket's file descriptor is marked O_NONBLOCK and no data is waiting to be received; or MSG_OOB is set and no out-of-band data is available and either the socket's file descriptor is marked O_NONBLOCK or the socket does not support blocking to await out-of-band data.

[EBADF] The *socket* argument is not a valid file descriptor.

[ECONNRESET] A connection was forcibly closed by a peer.

[EINTR] The *recv()* function was interrupted by a signal that was caught, before any data was available.

[EINVAL] The MSG_OOB flag is set and no out-of-band data is available.

[ENOTCONN] A receive is attempted on a connection-mode socket that is not connected.

[ENOTSOCK] The *socket* argument does not refer to a socket.

[EOPNOTSUPP] The specified flags are not supported for this socket type or protocol.

[ETIMEDOUT] The connection timed out during connection establishment, or due to a transmission timeout on active connection.

The *recv()* function may fail if:

[EIO] An I/O error occurred while reading from or writing to the file system.

[ENOBUFS] Insufficient resources were available in the system to perform the operation.

[ENOMEM] Insufficient memory was available to fulfill the request.

EXAMPLES

None.

APPLICATION USAGE

The *recv()* function is equivalent to *recvfrom()* with null pointer *address* and *address_len* arguments, and to *read()* if the *socket* argument refers to a socket and the *flags* argument is 0.

The *select()* and *poll()* functions can be used to determine when data is available to be received.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

poll(), *pselect()*, *read()*, *recvmsg()*, *recvfrom()*, *send()*, *sendmsg()*, *sendto()*, *shutdown()*, *socket()*, *write()*

XBD <sys/socket.h>

CHANGE HISTORY

First released in Issue 6. Derived from the XNS, Issue 5.2 specification.

57199	Issue 7	POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0502 [462] is applied.	+	
57200				

NAME

recvfrom — receive a message from a socket

SYNOPSIS

```
#include <sys/socket.h>

ssize_t recvfrom(int socket, void *restrict buffer, size_t length,
                 int flags, struct sockaddr *restrict address,
                 socklen_t *restrict address_len);
```

DESCRIPTION

The *recvfrom()* function shall receive a message from a connection-mode or connectionless-mode socket. It is normally used with connectionless-mode sockets because it permits the application to retrieve the source address of received data.

The *recvfrom()* function takes the following arguments:

<i>socket</i>	Specifies the socket file descriptor.
<i>buffer</i>	Points to the buffer where the message should be stored.
<i>length</i>	Specifies the length in bytes of the buffer pointed to by the <i>buffer</i> argument.
<i>flags</i>	Specifies the type of message reception. Values of this argument are formed by logically OR'ing zero or more of the following values:

MSG_PEEK	Peeks at an incoming message. The data is treated as unread and the next <i>recvfrom()</i> or similar function shall still return this data.

MSG_OOB	Requests out-of-band data. The significance and semantics of out-of-band data are protocol-specific.

MSG_WAITALL	On SOCK_STREAM sockets this requests that the function block until the full amount of data can be returned. The function may return the smaller amount of data if the socket is a message-based socket, if a signal is caught, if the connection is terminated, if MSG_PEEK was specified, or if an error is pending for the socket.

<i>address</i>	A null pointer, or points to a sockaddr structure in which the sending address is to be stored. The length and format of the address depend on the address family of the socket.

<i>address_len</i>	Either a null pointer, if <i>address</i> is a null pointer, or a pointer to a socklen_t object which on input specifies the length of the supplied sockaddr structure, and on output specifies the length of the stored address.

The *recvfrom()* function shall return the length of the message written to the buffer pointed to by the *buffer* argument. For message-based sockets, such as **SOCK_RAW**, **SOCK_DGRAM**, and **SOCK_SEQPACKET**, the entire message shall be read in a single operation. If a message is too long to fit in the supplied buffer, and MSG_PEEK is not set in the *flags* argument, the excess bytes shall be discarded. For stream-based sockets, such as **SOCK_STREAM**, message boundaries shall be ignored. In this case, data shall be returned to the user as soon as it becomes available, and no data shall be discarded.

If the MSG_WAITALL flag is not set, data shall be returned only up to the end of the first message.

Not all protocols provide the source address for messages. If the *address* argument is not a null

57245 pointer and the protocol provides the source address of messages, the source address of the
 57246 received message shall be stored in the **sockaddr** structure pointed to by the *address* argument,
 57247 and the length of this address shall be stored in the object pointed to by the *address_len*
 57248 argument.

57249 If the actual length of the address is greater than the length of the supplied **sockaddr** structure,
 57250 the stored address shall be truncated.

57251 If the *address* argument is not a null pointer and the protocol does not provide the source address
 57252 of messages, the value stored in the object pointed to by *address* is unspecified.

57253 If no messages are available at the socket and O_NONBLOCK is not set on the socket's file
 57254 descriptor, *recvfrom()* shall block until a message arrives. If no messages are available at the
 57255 socket and O_NONBLOCK is set on the socket's file descriptor, *recvfrom()* shall fail and set *errno*
 57256 to [EAGAIN] or [EWOULDBLOCK].

57257 RETURN VALUE

57258 Upon successful completion, *recvfrom()* shall return the length of the message in bytes. If no
 57259 messages are available to be received and the peer has performed an orderly shutdown,
 57260 *recvfrom()* shall return 0. Otherwise, the function shall return -1 and set *errno* to indicate the
 57261 error.

57262 ERRORS

57263 The *recvfrom()* function shall fail if:

57264 [EAGAIN] or [EWOULDBLOCK]

57265 The socket's file descriptor is marked O_NONBLOCK and no data is waiting
 57266 to be received; or MSG_OOB is set and no out-of-band data is available and
 57267 either the socket's file descriptor is marked O_NONBLOCK or the socket does
 57268 not support blocking to await out-of-band data.

57269 [EBADF] The *socket* argument is not a valid file descriptor.

57270 [ECONNRESET] A connection was forcibly closed by a peer.

57271 [EINTR] A signal interrupted *recvfrom()* before any data was available.

57272 [EINVAL] The MSG_OOB flag is set and no out-of-band data is available.

57273 [ENOTCONN] A receive is attempted on a connection-mode socket that is not connected.

57274 [ENOTSOCK] The *socket* argument does not refer to a socket.

57275 [EOPNOTSUPP] The specified flags are not supported for this socket type.

57276 [ETIMEDOUT] The connection timed out during connection establishment, or due to a
 57277 transmission timeout on active connection.

57278 The *recvfrom()* function may fail if:

57279 [EIO] An I/O error occurred while reading from or writing to the file system.

57280 [ENOBUFS] Insufficient resources were available in the system to perform the operation.

57281 [ENOMEM] Insufficient memory was available to fulfill the request.

57282 **EXAMPLES**

57283 None.

57284 **APPLICATION USAGE**57285 The *select()* and *poll()* functions can be used to determine when data is available to be received.57286 **RATIONALE**

57287 None.

57288 **FUTURE DIRECTIONS**

57289 None.

57290 **SEE ALSO**57291 *poll()*, *pselect()*, *read()*, *recv()*, *recvmsg()*, *send()*, *sendmsg()*, *sendto()*, *shutdown()*, *socket()*, *write()*

57292 XBD <sys/socket.h>

57293 **CHANGE HISTORY**

57294 First released in Issue 6. Derived from the XNS, Issue 5.2 specification.

57295 **Issue 7**

57296 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0503 [464] is applied.

+

57297 NAME

57298 recvmsg — receive a message from a socket

57299 SYNOPSIS

57300 #include <sys/socket.h>

57301 ssize_t recvmsg(int socket, struct msghdr *message, int flags);

57302 DESCRIPTION

57303 The *recvmsg()* function shall receive a message from a connection-mode or connectionless-mode
 57304 socket. It is normally used with connectionless-mode sockets because it permits the application
 57305 to retrieve the source address of received data.

57306 The *recvmsg()* function takes the following arguments:

57307	<i>socket</i>	Specifies the socket file descriptor.
57308	<i>message</i>	Points to a msghdr structure, containing both the buffer to store the source address and the buffers for the incoming message. The length and format of the address depend on the address family of the socket. The <i>msg_flags</i> member is ignored on input, but may contain meaningful values on output.
57312	<i>flags</i>	Specifies the type of message reception. Values of this argument are formed by logically OR'ing zero or more of the following values:
57314	MSG_OOB	Requests out-of-band data. The significance and semantics of out-of-band data are protocol-specific.
57316	MSG_PEEK	Peeks at the incoming message.
57317	MSG_WAITALL	On SOCK_STREAM sockets this requests that the function block until the full amount of data can be returned. The function may return the smaller amount of data if the socket is a message-based socket, if a signal is caught, if the connection is terminated, if MSG_PEEK was specified, or if an error is pending for the socket.

57323 The *recvmsg()* function shall receive messages from unconnected or connected sockets and shall
 57324 return the length of the message.

57325 The *recvmsg()* function shall return the total length of the message. For message-based sockets,
 57326 such as SOCK_DGRAM and SOCK_SEQPACKET, the entire message shall be read in a single
 57327 operation. If a message is too long to fit in the supplied buffers, and MSG_PEEK is not set in the
 57328 *flags* argument, the excess bytes shall be discarded, and MSG_TRUNC shall be set in the
 57329 *msg_flags* member of the **msghdr** structure. For stream-based sockets, such as SOCK_STREAM,
 57330 message boundaries shall be ignored. In this case, data shall be returned to the user as soon as it
 57331 becomes available, and no data shall be discarded.

57332 If the MSG_WAITALL flag is not set, data shall be returned only up to the end of the first
 57333 message.

57334 If no messages are available at the socket and O_NONBLOCK is not set on the socket's file
 57335 descriptor, *recvmsg()* shall block until a message arrives. If no messages are available at the
 57336 socket and O_NONBLOCK is set on the socket's file descriptor, the *recvmsg()* function shall fail
 57337 and set *errno* to [EAGAIN] or [EWOULDBLOCK].

57338 In the **msghdr** structure, the *msg_name* member may be a null pointer if the source address is not
 57339 required. Otherwise, if the socket is unconnected, the *msg_name* member points to a **sockaddr**
 57340 structure in which the source address is to be stored, and the *msg_namelen* member on input
 57341 specifies the length of the supplied **sockaddr** structure and on output specifies the length of the

stored address. If the actual length of the address is greater than the length of the supplied **sockaddr** structure, the stored address shall be truncated. If the socket is connected, the *msg_name* and *msg_namelen* members shall be ignored. The *msg_iov* and *msg_iovlen* fields are used to specify where the received data shall be stored. The *msg_iov* member points to an array of **iovec** structures; the *msg_iovlen* member shall be set to the dimension of this array. In each **iovec** structure, the *iov_base* field specifies a storage area and the *iov_len* field gives its size in bytes. Each storage area indicated by *msg_iov* is filled with received data in turn until all of the received data is stored or all of the areas have been filled.

Upon successful completion, the *msg_flags* member of the message header shall be the bitwise-inclusive OR of all of the following flags that indicate conditions detected for the received message:

MSG_EOR	End-of-record was received (if supported by the protocol).
MSG_OOB	Out-of-band data was received.
MSG_TRUNC	Normal data was truncated.
MSG_CTRUNC	Control data was truncated.

RETURN VALUE

Upon successful completion, *recvmsg()* shall return the length of the message in bytes. If no messages are available to be received and the peer has performed an orderly shutdown, *recvmsg()* shall return 0. Otherwise, -1 shall be returned and *errno* set to indicate the error.

ERRORS

The *recvmsg()* function shall fail if:

[EAGAIN] or [EWOULDBLOCK]

The socket's file descriptor is marked O_NONBLOCK and no data is waiting to be received; or MSG_OOB is set and no out-of-band data is available and either the socket's file descriptor is marked O_NONBLOCK or the socket does not support blocking to await out-of-band data.

[EBADF] The *socket* argument is not a valid open file descriptor.

[ECONNRESET] A connection was forcibly closed by a peer.

[EINTR] This function was interrupted by a signal before any data was available.

[EINVAL] The sum of the *iov_len* values overflows a **ssize_t**, or the MSG_OOB flag is set and no out-of-band data is available.

[EMSGSIZE] The *msg_iovlen* member of the **msghdr** structure pointed to by *message* is less than or equal to 0, or is greater than {IOV_MAX}.

[ENOTCONN] A receive is attempted on a connection-mode socket that is not connected.

[ENOTSOCK] The *socket* argument does not refer to a socket.

[EOPNOTSUPP] The specified flags are not supported for this socket type.

[ETIMEDOUT] The connection timed out during connection establishment, or due to a transmission timeout on active connection.

The *recvmsg()* function may fail if:

[EIO] An I/O error occurred while reading from or writing to the file system.

57382 [ENOBUFS] Insufficient resources were available in the system to perform the operation.

57383 [ENOMEM] Insufficient memory was available to fulfill the request.

57384 **EXAMPLES**

57385 None.

57386 **APPLICATION USAGE**

57387 The *select()* and *poll()* functions can be used to determine when data is available to be received.

57388 **RATIONALE**

57389 None.

57390 **FUTURE DIRECTIONS**

57391 None.

57392 **SEE ALSO**

57393 *poll()*, *pselect()*, *recv()*, *recvfrom()*, *send()*, *sendmsg()*, *sendto()*, *shutdown()*, *socket()*

57394 XBD <sys/socket.h>

57395 **CHANGE HISTORY**

57396 First released in Issue 6. Derived from the XNS, Issue 5.2 specification.

57397 **Issue 7**

57398 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0504 [464] is applied.

+

NAME

regcomp, regerror, regex, regfree — regular expression matching

SYNOPSIS

```
#include <regex.h>

int regcomp(regex_t *restrict preg, const char *restrict pattern,
            int cflags);
size_t regerror(int errcode, const regex_t *restrict preg,
               char *restrict errbuf, size_t errbuf_size);
int regex(const regex_t *restrict preg, const char *restrict string,
          size_t nmatch, regmatch_t pmatch[restrict], int eflags);
void regfree(regex_t *preg);
```

DESCRIPTION

These functions interpret *basic* and *extended* regular expressions as described in XBD [Chapter 9](#) (on page 181).

The **regex_t** structure is defined in **<regex.h>** and contains at least the following member:

Member Type	Member Name	Description
size_t	<i>re_nsub</i>	Number of parenthesized subexpressions.

The **regmatch_t** structure is defined in **<regex.h>** and contains at least the following members:

Member Type	Member Name	Description
regoff_t	<i>rm_so</i>	Byte offset from start of <i>string</i> to start of substring.
regoff_t	<i>rm_eo</i>	Byte offset from start of <i>string</i> of the first character after the end of substring.

The *regcomp()* function shall compile the regular expression contained in the string pointed to by the *pattern* argument and place the results in the structure pointed to by *preg*. The *cflags* argument is the bitwise-inclusive OR of zero or more of the following flags, which are defined in the **<regex.h>** header:

REG_EXTENDED Use Extended Regular Expressions.

REG_ICASE Ignore case in match (see XBD [Chapter 9](#), on page 181).

REG_NOSUB Report only success/fail in *regex()*.

REG_NEWLINE Change the handling of <newline> characters, as described in the text.

The default regular expression type for *pattern* is a Basic Regular Expression. The application can specify Extended Regular Expressions using the **REG_EXTENDED** *cflags* flag.

If the **REG_NOSUB** flag was not set in *cflags*, then *regcomp()* shall set *re_nsub* to the number of parenthesized subexpressions (delimited by "*\(\)*" in basic regular expressions or "*()*" in extended regular expressions) found in *pattern*.

The *regex()* function compares the null-terminated string specified by *string* with the compiled regular expression *preg* initialized by a previous call to *regcomp()*. If it finds a match, *regex()* shall return 0; otherwise, it shall return non-zero indicating either no match or an error. The *eflags* argument is the bitwise-inclusive OR of zero or more of the following flags, which are defined in the **<regex.h>** header:

- 57441 REG_NOTBOL The first character of the string pointed to by *string* is not the beginning of
 57442 the line. Therefore, the <circumflex> character ('^'), when taken as a
 57443 special character, shall not match the beginning of *string*.
- 57444 REG_NOTEOL The last character of the string pointed to by *string* is not the end of the
 57445 line. Therefore, the <dollar-sign> ('\$'), when taken as a special character,
 57446 shall not match the end of *string*.
- 57447 If *nmatch* is 0 or REG_NOSUB was set in the *cflags* argument to *regcomp()*, then *regexexec()* shall
 57448 ignore the *pmatch* argument. Otherwise, the application shall ensure that the *pmatch* argument
 57449 points to an array with at least *nmatch* elements, and *regexexec()* shall fill in the elements of that
 57450 array with offsets of the substrings of *string* that correspond to the parenthesized subexpressions
 57451 of *pattern*: *pmatch[i].rm_so* shall be the byte offset of the beginning and *pmatch[i].rm_eo* shall be
 57452 one greater than the byte offset of the end of substring *i*. (Subexpression *i* begins at the *i*th
 57453 matched open parenthesis, counting from 1.) Offsets in *pmatch[0]* identify the substring that
 57454 corresponds to the entire regular expression. Unused elements of *pmatch* up to *pmatch[nmatch-1]*
 57455 shall be filled with -1. If there are more than *nmatch* subexpressions in *pattern* (*pattern* itself
 57456 counts as a subexpression), then *regexexec()* shall still do the match, but shall record only the first
 57457 *nmatch* substrings.
- 57458 When matching a basic or extended regular expression, any given parenthesized subexpression
 57459 of *pattern* might participate in the match of several different substrings of *string*, or it might not
 57460 match any substring even though the pattern as a whole did match. The following rules shall be
 57461 used to determine which substrings to report in *pmatch* when matching regular expressions:
- 57462 1. If subexpression *i* in a regular expression is not contained within another subexpression,
 57463 and it participated in the match several times, then the byte offsets in *pmatch[i]* shall
 57464 delimit the last such match.
 - 57465 2. If subexpression *i* is not contained within another subexpression, and it did not
 57466 participate in an otherwise successful match, the byte offsets in *pmatch[i]* shall be -1. A
 57467 subexpression does not participate in the match when:
 57468 ' * ' or " \{ \} " appears immediately after the subexpression in a basic regular
 57469 expression, or ' * ', ' ? ', or " { } " appears immediately after the subexpression in
 57470 an extended regular expression, and the subexpression did not match (matched 0
 57471 times)
- 57472 or:
- 57473 ' | ' is used in an extended regular expression to select this subexpression or
 57474 another, and the other subexpression matched.
- 57475 3. If subexpression *i* is contained within another subexpression *j*, and *i* is not contained
 57476 within any other subexpression that is contained within *j*, and a match of subexpression *j*
 57477 is reported in *pmatch[j]*, then the match or non-match of subexpression *i* reported in
 57478 *pmatch[i]* shall be as described in 1. and 2. above, but within the substring reported in
 57479 *pmatch[j]* rather than the whole string. The offsets in *pmatch[i]* are still relative to the start
 57480 of *string*.
 - 57481 4. If subexpression *i* is contained in subexpression *j*, and the byte offsets in *pmatch[j]* are -1,
 57482 then the pointers in *pmatch[i]* shall also be -1.
 - 57483 5. If subexpression *i* matched a zero-length string, then both byte offsets in *pmatch[i]* shall be
 57484 the byte offset of the character or null terminator immediately following the zero-length
 57485 string.

57486 If, when *regexexec()* is called, the locale is different from when the regular expression was
57487 compiled, the result is undefined.

57488 If REG_NEWLINE is not set in *cflags*, then a <newline> in *pattern* or *string* shall be treated as an
57489 ordinary character. If REG_NEWLINE is set, then <newline> shall be treated as an ordinary
57490 character except as follows:

- 57491 1. A <newline> in *string* shall not be matched by a <period> outside a bracket expression or
57492 by any form of a non-matching list (see XBD [Chapter 9](#), on page 181).
- 57493 2. A <circumflex> (' ^ ') in *pattern*, when used to specify expression anchoring (see XBD
57494 [Section 9.3.8](#), on page 187), shall match the zero-length string immediately after a
57495 <newline> in *string*, regardless of the setting of REG_NOTBOL.
- 57496 3. A <dollar-sign> (' \$ ') in *pattern*, when used to specify expression anchoring, shall match
57497 the zero-length string immediately before a <newline> in *string*, regardless of the setting
57498 of REG_NOTEOL.

57499 The *regfree()* function frees any memory allocated by *regcomp()* associated with *preg*.

57500 The following constants are defined as the minimum set of error return values, although other
57501 errors listed as implementation extensions in <regex.h> are possible:

57502	REG_BADBR	Content of "\{\}" invalid: not a number, number too large, more than
57503		two numbers, first larger than second.
57504	REG_BADPAT	Invalid regular expression.
57505	REG_BADRPT	'?', '*', or '+' not preceded by valid regular expression.
57506	REG_EBRACE	"\{\}" imbalance.
57507	REG_EBRACK	"[]" imbalance.
57508	REG_ECOLLATE	Invalid collating element referenced.
57509	REG_ECTYPE	Invalid character class type referenced.
57510	REG_EESCAPE	Trailing <backslash> character in pattern.
57511	REG_EPAREN	"\(\)" or "()" imbalance.
57512	REG_ERANGE	Invalid endpoint in range expression.
57513	REG_ESPACE	Out of memory.
57514	REG_ESUBREG	Number in "\digit" invalid or in error.
57515	REG_NOMATCH	<i>regexexec()</i> failed to match.

57516 If more than one error occurs in processing a function call, any one of the possible constants may
57517 be returned, as the order of detection is unspecified.

57518 The *regerror()* function provides a mapping from error codes returned by *regcomp()* and
57519 *regexexec()* to unspecified printable strings. It generates a string corresponding to the value of the
57520 *errcode* argument, which the application shall ensure is the last non-zero value returned by
57521 *regcomp()* or *regexexec()* with the given value of *preg*. If *errcode* is not such a value, the content of
57522 the generated string is unspecified.

57523 If *preg* is a null pointer, but *errcode* is a value returned by a previous call to *regexexec()* or *regcomp()*,
57524 the *regerror()* still generates an error string corresponding to the value of *errcode*, but it might not
57525 be as detailed under some implementations.

If the *errbuf_size* argument is not 0, *regerror()* shall place the generated string into the buffer of size *errbuf_size* bytes pointed to by *errbuf*. If the string (including the terminating null) cannot fit in the buffer, *regerror()* shall truncate the string and null-terminate the result.

If *errbuf_size* is 0, *regerror()* shall ignore the *errbuf* argument, and return the size of the buffer needed to hold the generated string.

If the *preg* argument to *regexec()* or *regfree()* is not a compiled regular expression returned by *regcomp()*, the result is undefined. A *preg* is no longer treated as a compiled regular expression after it is given to *regfree()*.

57534 RETURN VALUE

Upon successful completion, the *regcomp()* function shall return 0. Otherwise, it shall return an integer value indicating an error as described in **<regex.h>**, and the content of *preg* is undefined. If a code is returned, the interpretation shall be as given in **<regex.h>**.

If *regcomp()* detects an invalid RE, it may return REG_BADPAT, or it may return one of the error codes that more precisely describes the error.

Upon successful completion, the *regexec()* function shall return 0. Otherwise, it shall return REG_NOMATCH to indicate no match.

Upon successful completion, the *regerror()* function shall return the number of bytes needed to hold the entire generated string, including the null termination. If the return value is greater than *errbuf_size*, the string returned in the buffer pointed to by *errbuf* has been truncated.

The *regfree()* function shall not return a value.

57546 ERRORS

No errors are defined.

57548 EXAMPLES

```
57549 #include <regex.h>
57550 /*
57551  * Match string against the extended regular expression in
57552  * pattern, treating errors as no match.
57553  *
57554  * Return 1 for match, 0 for no match.
57555  */
57556 int
57557 match(const char *string, char *pattern)
57558 {
57559     int    status;
57560     regex_t re;
57561     if (regcomp(&re, pattern, REG_EXTENDED|REG_NOSUB) != 0) {
57562         return(0);        /* Report error. */
57563     }
57564     status = regexec(&re, string, (size_t) 0, NULL, 0);
57565     regfree(&re);
57566     if (status != 0) {
57567         return(0);        /* Report error. */
57568     }
57569     return(1);
57570 }
```


The following demonstrates how the REG_NOTBOL flag could be used with *regexexec()* to find all substrings in a line that match a pattern supplied by a user. (For simplicity of the example, very little error checking is done.)

```
(void) regcomp (&re, pattern, 0);
/* This call to regexexec() finds the first match on the line. */
error = regexexec (&re, &buffer[0], 1, &pm, 0);
while (error == 0) { /* While matches found. */
    /* Substring found between pm.rm_so and pm.rm_eo. */
    /* This call to regexexec() finds the next match. */
    error = regexexec (&re, buffer + pm.rm_eo, 1, &pm, REG_NOTBOL);
}
```

APPLICATION USAGE

An application could use:

```
regerror(code, preg, (char *)NULL, (size_t)0)
```

to find out how big a buffer is needed for the generated string, *malloc()* a buffer to hold the string, and then call *regerror()* again to get the string. Alternatively, it could allocate a fixed, static buffer that is big enough to hold most strings, and then use *malloc()* to allocate a larger buffer if it finds that this is too small.

To match a pattern as described in XCU [Section 2.13](#) (on page 2354), use the *fnmatch()* function.

RATIONALE

The *regexexec()* function must fill in all *nmatch* elements of *pmatch*, where *nmatch* and *pmatch* are supplied by the application, even if some elements of *pmatch* do not correspond to subexpressions in *pattern*. The application developer should note that there is probably no reason for using a value of *nmatch* that is larger than *preg->re_nsub*+1.

The REG_NEWLINE flag supports a use of RE matching that is needed in some applications like text editors. In such applications, the user supplies an RE asking the application to find a line that matches the given expression. An anchor in such an RE anchors at the beginning or end of any line. Such an application can pass a sequence of <newline>-separated lines to *regexexec()* as a single long string and specify REG_NEWLINE to *regcomp()* to get the desired behavior. The application must ensure that there are no explicit <newline> characters in *pattern* if it wants to ensure that any match occurs entirely within a single line.

The REG_NEWLINE flag affects the behavior of *regexexec()*, but it is in the *cflags* parameter to *regcomp()* to allow flexibility of implementation. Some implementations will want to generate the same compiled RE in *regcomp()* regardless of the setting of REG_NEWLINE and have *regexexec()* handle anchors differently based on the setting of the flag. Other implementations will generate different compiled REs based on the REG_NEWLINE.

The REG_ICASE flag supports the operations taken by the *grep -i* option and the historical implementations of *ex* and *vi*. Including this flag will make it easier for application code to be written that does the same thing as these utilities.

The substrings reported in *pmatch[]* are defined using offsets from the start of the string rather than pointers. This allows type-safe access to both constant and non-constant strings.

The type **regoff_t** is used for the elements of *pmatch[]* to ensure that the application can represent large arrays in memory (important for an application conforming to the Shell and Utilities volume of POSIX.1-2008).

The 1992 edition of this standard required **regoff_t** to be at least as wide as **off_t**, to facilitate future extensions in which the string to be searched is taken from a file. However, these future

extensions have not appeared. The requirement rules out popular implementations with 32-bit **regoff_t** and 64-bit **off_t**, so it has been removed.

The standard developers rejected the inclusion of a *regsub()* function that would be used to do substitutions for a matched RE. While such a routine would be useful to some applications, its utility would be much more limited than the matching function described here. Both RE parsing and substitution are possible to implement without support other than that required by the ISO C standard, but matching is much more complex than substituting. The only difficult part of substitution, given the information supplied by *regexexec()*, is finding the next character in a string when there can be multi-byte characters. That is a much larger issue, and one that needs a more general solution.

The *errno* variable has not been used for error returns to avoid filling the *errno* name space for this feature.

The interface is defined so that the matched substrings *rm_sp* and *rm_ep* are in a separate **regmatch_t** structure instead of in **regex_t**. This allows a single compiled RE to be used simultaneously in several contexts; in *main()* and a signal handler, perhaps, or in multiple threads of lightweight processes. (The *preg* argument to *regexexec()* is declared with type **const**, so the implementation is not permitted to use the structure to store intermediate results.) It also allows an application to request an arbitrary number of substrings from an RE. The number of subexpressions in the RE is reported in *re_nsub* in *preg*. With this change to *regexexec()*, consideration was given to dropping the REG_NOSUB flag since the user can now specify this with a zero *nmatch* argument to *regexexec()*. However, keeping REG_NOSUB allows an implementation to use a different (perhaps more efficient) algorithm if it knows in *regcomp()* that no subexpressions need be reported. The implementation is only required to fill in *pmatch* if *nmatch* is not zero and if REG_NOSUB is not specified. Note that the **size_t** type, as defined in the ISO C standard, is unsigned, so the description of *regexexec()* does not need to address negative values of *nmatch*.

REG_NOTBOL was added to allow an application to do repeated searches for the same pattern in a line. If the pattern contains a <circumflex> character that should match the beginning of a line, then the pattern should only match when matched against the beginning of the line. Without the REG_NOTBOL flag, the application could rewrite the expression for subsequent matches, but in the general case this would require parsing the expression. The need for REG_NOTEOL is not as clear; it was added for symmetry.

The addition of the *regerror()* function addresses the historical need for conforming application programs to have access to error information more than “Function failed to compile/match your RE for unknown reasons”.

This interface provides for two different methods of dealing with error conditions. The specific error codes (REG_EBRACE, for example), defined in <**regex.h**>, allow an application to recover from an error if it is so able. Many applications, especially those that use patterns supplied by a user, will not try to deal with specific error cases, but will just use *regerror()* to obtain a human-readable error message to present to the user.

The *regerror()* function uses a scheme similar to *confstr()* to deal with the problem of allocating memory to hold the generated string. The scheme used by *strerror()* in the ISO C standard was considered unacceptable since it creates difficulties for multi-threaded applications.

The *preg* argument is provided to *regerror()* to allow an implementation to generate a more descriptive message than would be possible with *errcode* alone. An implementation might, for example, save the character offset of the offending character of the pattern in a field of *preg*, and then include that in the generated message string. The implementation may also ignore *preg*.

A REG_FILENAME flag was considered, but omitted. This flag caused *regexexec()* to match patterns as described in XCU Section 2.13 (on page 2354) instead of REs. This service is now provided by the *fnmatch()* function.

Notice that there is a difference in philosophy between the ISO POSIX-2:1993 standard and POSIX.1-2008 in how to handle a “bad” regular expression. The ISO POSIX-2:1993 standard says that many bad constructs “produce undefined results”, or that “the interpretation is undefined”. POSIX.1-2008, however, says that the interpretation of such REs is unspecified. The term “undefined” means that the action by the application is an error, of similar severity to passing a bad pointer to a function.

The *regcomp()* and *regexexec()* functions are required to accept any null-terminated string as the *pattern* argument. If the meaning of the string is “undefined”, the behavior of the function is “unspecified”. POSIX.1-2008 does not specify how the functions will interpret the pattern; they might return error codes, or they might do pattern matching in some completely unexpected way, but they should not do something like abort the process.

FUTURE DIRECTIONS

None.

SEE ALSO

fnmatch(), *glob()*

XBD Chapter 9 (on page 181), *<regex.h>*, *<sys/types.h>*

XCUI Section 2.13 (on page 2354)

CHANGE HISTORY

First released in Issue 4. Derived from the ISO POSIX-2 standard.

Issue 5

Moved from POSIX2 C-language Binding to BASE.

Issue 6

In the SYNOPSIS, the optional include of the *<sys/types.h>* header is removed.

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The requirement to include *<sys/types.h>* has been removed. Although *<sys/types.h>* was required for conforming implementations of previous POSIX specifications, it was not required for UNIX applications.

The normative text is updated to avoid use of the term “must” for application requirements.

The REG_ENOSYS constant is removed.

The **restrict** keyword is added to the *regcomp()*, *regerror()*, and *regexexec()* prototypes for alignment with the ISO/IEC 9899:1999 standard.

Issue 7

Austin Group Interpretation 1003.1-2001 #134 is applied, clarifying that if more than one error occurs in processing a function call, any one of the possible constants may be returned.

SD5-XBD-ERN-60 is applied.

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0505 [305] is applied.

+

57704 **NAME**

57705 remainder, remainderf, remainderl — remainder function

57706 **SYNOPSIS**

57707 #include <math.h>

57708 double remainder(double x, double y);

57709 float remainderf(float x, float y);

57710 long double remainderl(long double x, long double y);

57711 **DESCRIPTION**

57712 CX The functionality described on this reference page is aligned with the ISO C standard. Any
 57713 conflict between the requirements described here and the ISO C standard is unintentional. This
 57714 volume of POSIX.1-2008 defers to the ISO C standard.

57715 These functions shall return the floating-point remainder $r=x-ny$ when y is non-zero. The value
 57716 n is the integral value nearest the exact value x/y . When $|n-x/y|=\frac{1}{2}$, the value n is chosen to
 57717 be even.

57718 The behavior of *remainder()* shall be independent of the rounding mode.

57719 **RETURN VALUE**

57720 Upon successful completion, these functions shall return the floating-point remainder $r=x-ny$
 57721 when y is non-zero.

57722 On systems that do not support the IEC 60559 Floating-Point option, if y is zero, it is
 57723 implementation-defined whether a domain error occurs or zero is returned.

57724 MX If x or y is NaN, a NaN shall be returned.

57725 If x is infinite or y is 0 and the other is non-NaN, a domain error shall occur, and a NaN shall be
 57726 returned.

57727 **ERRORS**

57728 These functions shall fail if:

57729 MX **Domain Error** The x argument is $\pm\text{Inf}$, or the y argument is ± 0 and the other argument is non-
 57730 NaN.

57731 If the integer expression (*math_errhandling* & MATH_ERRNO) is non-zero,
 57732 then *errno* shall be set to [EDOM]. If the integer expression (*math_errhandling*
 57733 & MATH_ERREXCEPT) is non-zero, then the invalid floating-point exception
 57734 shall be raised.

57735 These functions may fail if:

57736 **Domain Error** The y argument is zero.

57737 If the integer expression (*math_errhandling* & MATH_ERRNO) is non-zero,
 57738 then *errno* shall be set to [EDOM]. If the integer expression (*math_errhandling*
 57739 & MATH_ERREXCEPT) is non-zero, then the invalid floating-point exception
 57740 shall be raised.

57741 **EXAMPLES**

57742 None.

57743 **APPLICATION USAGE**

57744 On error, the expressions (*math_errhandling* & MATH_ERRNO) and (*math_errhandling* &
 57745 MATH_ERREXCEPT) are independent of each other, but at least one of them must be non-zero.

57746 **RATIONALE**

57747 None.

57748 **FUTURE DIRECTIONS**

57749 None.

57750 **SEE ALSO**57751 *abs()*, *div()*, *feclearexcept()*, *fetestexcept()*, *ldiv()*

57752 XBD Section 4.19 (on page 116), <math.h>

57753 **CHANGE HISTORY**

57754 First released in Issue 4, Version 2.

57755 **Issue 5**

57756 Moved from X/OPEN UNIX extension to BASE.

57757 **Issue 6**57758 The *remainder()* function is no longer marked as an extension.

57759 The *remainderf()* and *remainderl()* functions are added for alignment with the ISO/IEC 9899:1999
 57760 standard.

57761 The DESCRIPTION, RETURN VALUE, ERRORS, and APPLICATION USAGE sections are
 57762 revised to align with the ISO/IEC 9899:1999 standard.

57763 IEC 60559:1989 standard floating-point extensions over the ISO/IEC 9899:1999 standard are
 57764 marked.

57765 **Issue 7**

57766 ISO/IEC 9899:1999 standard, Technical Corrigendum 2 #55 (SD5-XSH-ERN-82) is applied.

57767 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0506 [320] is applied.

+

57768 **NAME**

57769 remove — remove a file

57770 **SYNOPSIS**

57771 #include <stdio.h>

57772 int remove(const char *path);

57773 **DESCRIPTION**

57774 CX The functionality described on this reference page is aligned with the ISO C standard. Any
 57775 conflict between the requirements described here and the ISO C standard is unintentional. This
 57776 volume of POSIX.1-2008 defers to the ISO C standard.

57777 The *remove()* function shall cause the file named by the pathname pointed to by *path* to be no
 57778 longer accessible by that name. A subsequent attempt to open that file using that name shall fail,
 57779 unless it is created anew.

57780 CX If *path* does not name a directory, *remove(path)* shall be equivalent to *unlink(path)*.

57781 If *path* names a directory, *remove(path)* shall be equivalent to *rmdir(path)*.

57782 **RETURN VALUE**57783 CX Refer to *rmdir()* or *unlink()*.57784 **ERRORS**57785 CX Refer to *rmdir()* or *unlink()*.57786 **EXAMPLES**57787 **Removing Access to a File**57788 The following example shows how to remove access to a file named */home/cnd/old_mods*.

57789 #include <stdio.h>

57790 int status;

57791 ...

57792 status = remove("/home/cnd/old_mods");

57793 **APPLICATION USAGE**

57794 None.

57795 **RATIONALE**

57796 None.

57797 **FUTURE DIRECTIONS**

57798 None.

57799 **SEE ALSO**57800 *rmdir()*, *unlink()*

57801 XBD <stdio.h>

57802 **CHANGE HISTORY**

57803 First released in Issue 3. Included for alignment with the POSIX.1-1988 standard and the ISO C
 57804 standard.

57805 **Issue 6**

57806 Extensions beyond the ISO C standard are marked.

57807
57808

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

57809
57810
57811

- The DESCRIPTION, RETURN VALUE, and ERRORS sections are updated so that if *path* is not a directory, *remove()* is equivalent to *unlink()*, and if it is a directory, it is equivalent to *rmdir()*.

57812 **NAME**

57813 remque — remove an element from a queue

57814 **SYNOPSIS**

57815 XSI `#include <search.h>`

57816 `void remque(void *element);`

57817 **DESCRIPTION**

57818 Refer to *insque()*.

57819 **NAME**

57820 remquo, remquof, remquol — remainder functions

57821 **SYNOPSIS**

```
57822 #include <math.h>
57823 double remquo(double x, double y, int *quo);
57824 float remquof(float x, float y, int *quo);
57825 long double remquol(long double x, long double y, int *quo);
```

57826 **DESCRIPTION**

57827 CX The functionality described on this reference page is aligned with the ISO C standard. Any
 57828 conflict between the requirements described here and the ISO C standard is unintentional. This
 57829 volume of POSIX.1-2008 defers to the ISO C standard.

57830 The *remquo()*, *remquof()*, and *remquol()* functions shall compute the same remainder as the
 57831 *remainder()*, *remainderf()*, and *remainderl()* functions, respectively. In the object pointed to by *quo*,
 57832 they store a value whose sign is the sign of x/y and whose magnitude is congruent modulo 2^n to
 57833 the magnitude of the integral quotient of x/y , where n is an implementation-defined integer
 57834 greater than or equal to 3. If y is zero, the value stored in the object pointed to by *quo* is
 57835 unspecified.

57836 An application wishing to check for error situations should set *errno* to zero and call
 57837 *feclearexcept*(FE_ALL_EXCEPT) before calling these functions. On return, if *errno* is non-zero or
 57838 *fetestexcept*(FE_INVALID | FE_DIVBYZERO | FE_OVERFLOW | FE_UNDERFLOW) is non-
 57839 zero, an error has occurred.

57840 **RETURN VALUE**57841 These functions shall return $x \text{ REM } y$.

57842 On systems that do not support the IEC 60559 Floating-Point option, if y is zero, it is
 57843 implementation-defined whether a domain error occurs or zero is returned.

57844 MX If x or y is NaN, a NaN shall be returned.

57845 If x is $\pm\text{Inf}$ or y is zero and the other argument is non-NaN, a domain error shall occur, and a
 57846 NaN shall be returned.

57847 **ERRORS**

57848 These functions shall fail if:

57849 MX **Domain Error** The x argument is $\pm\text{Inf}$, or the y argument is ± 0 and the other argument is non-
 57850 NaN.

57851 If the integer expression $(\text{math\_errhandling} \ \& \ \text{MATH\_ERRNO})$ is non-zero,
 57852 then *errno* shall be set to [EDOM]. If the integer expression $(\text{math\_errhandling}$
 57853 $\ \& \ \text{MATH\_ERREXCEPT})$ is non-zero, then the invalid floating-point exception
 57854 shall be raised.

57855 These functions may fail if:

57856 **Domain Error** The y argument is zero.

57857 If the integer expression $(\text{math\_errhandling} \ \& \ \text{MATH\_ERRNO})$ is non-zero,
 57858 then *errno* shall be set to [EDOM]. If the integer expression $(\text{math\_errhandling}$
 57859 $\ \& \ \text{MATH\_ERREXCEPT})$ is non-zero, then the invalid floating-point exception
 57860 shall be raised.

57861 **EXAMPLES**

57862 None.

57863 **APPLICATION USAGE**

57864 On error, the expressions (*math_errhandling* & MATH_ERRNO) and (*math_errhandling* &
57865 MATH_ERREXCEPT) are independent of each other, but at least one of them must be non-zero.

57866 **RATIONALE**

57867 These functions are intended for implementing argument reductions which can exploit a few
57868 low-order bits of the quotient. Note that *x* may be so large in magnitude relative to *y* that an
57869 exact representation of the quotient is not practical.

57870 **FUTURE DIRECTIONS**

57871 None.

57872 **SEE ALSO**57873 *feclearexcept()*, *fetestexcept()*, *remainder()*

57874 XBD Section 4.19 (on page 116), <math.h>

57875 **CHANGE HISTORY**

57876 First released in Issue 6. Derived from the ISO/IEC 9899: 1999 standard.

57877 **Issue 7**

57878 ISO/IEC 9899: 1999 standard, Technical Corrigendum 2 #56 (SD5-XSH-ERN-83) is applied.

57879 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0507 [320] is applied. +

57880 **NAME**57881 `rename, renameat` — rename file relative to directory file descriptor57882 **SYNOPSIS**57883 `#include <stdio.h>`57884 `int rename(const char *old, const char *new);`57885 CX `int renameat(int oldfd, const char *old, int newfd,`
57886 `const char *new);`57887 **DESCRIPTION**57888 CX For `rename()`: The functionality described on this reference page is aligned with the ISO C
57889 standard. Any conflict between the requirements described here and the ISO C standard is
57890 unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.57891 The `rename()` function shall change the name of a file. The *old* argument points to the pathname
57892 CX of the file to be renamed. The *new* argument points to the new pathname of the file. If the *new*
57893 argument does not resolve to an existing directory entry for a file of type directory and the *new*
57894 argument contains at least one non-`<slash>` character and ends with one or more trailing
57895 `<slash>` characters after all symbolic links have been processed, `rename()` shall fail.57896 If either the *old* or *new* argument names a symbolic link, `rename()` shall operate on the symbolic
57897 link itself, and shall not resolve the last component of the argument. If the *old* argument and the
57898 *new* argument resolve to either the same existing directory entry or different directory entries for
57899 the same existing file, `rename()` shall return successfully and perform no other action.57900 If the *old* argument points to the pathname of a file that is not a directory, the *new* argument shall
57901 not point to the pathname of a directory. If the link named by the *new* argument exists, it shall be
57902 removed and *old* renamed to *new*. In this case, a link named *new* shall remain visible to other
57903 processes throughout the renaming operation and refer either to the file referred to by *new* or *old*
57904 before the operation began. Write access permission is required for both the directory containing
57905 *old* and the directory containing *new*.57906 If the *old* argument points to the pathname of a directory, the *new* argument shall not point to the
57907 pathname of a file that is not a directory. If the directory named by the *new* argument exists, it
57908 shall be removed and *old* renamed to *new*. In this case, a link named *new* shall exist throughout
57909 the renaming operation and shall refer either to the directory referred to by *new* or *old* before the
57910 operation began. If *new* names an existing directory, it shall be required to be an empty directory.57911 If either *pathname* argument refers to a path whose final component is either dot or dot-dot,
57912 `rename()` shall fail.57913 If the *old* argument points to a pathname of a symbolic link, the symbolic link shall be renamed.
57914 If the *new* argument points to a pathname of a symbolic link, the symbolic link shall be removed.57915 The *old* pathname shall not name an ancestor directory of the *new* pathname. Write access
57916 permission is required for the directory containing *old* and the directory containing *new*. If the
57917 *old* argument points to the pathname of a directory, write access permission may be required for
57918 the directory named by *old*, and, if it exists, the directory named by *new*.57919 If the link named by the *new* argument exists and the file's link count becomes 0 when it is
57920 removed and no process has the file open, the space occupied by the file shall be freed and the
57921 file shall no longer be accessible. If one or more processes have the file open when the last link is
57922 removed, the link shall be removed before `rename()` returns, but the removal of the file contents
57923 shall be postponed until all references to the file are closed.57924 Upon successful completion, `rename()` shall mark for update the last data modification and last

57925 file status change timestamps of the parent directory of each file.

57926 If the `rename()` function fails for any reason other than [EIO], any file named by *new* shall be
 57927 unaffected.

57928 The `renameat()` function shall be equivalent to the `rename()` function except in the case where
 57929 either *old* or *new* specifies a relative path. If *old* is a relative path, the file to be renamed is located
 57930 relative to the directory associated with the file descriptor *oldfd* instead of the current working
 57931 directory. If *new* is a relative path, the same happens only relative to the directory associated
 57932 with *newfd*. If the file descriptor was opened without O_SEARCH, the function shall check
 57933 whether directory searches are permitted using the current permissions of the directory
 57934 underlying the file descriptor. If the file descriptor was opened with O_SEARCH, the function
 57935 shall not perform the check.

57936 If `renameat()` is passed the special value AT_FDCWD in the *oldfd* or *newfd* parameter, the current
 57937 working directory shall be used in the determination of the file for the respective *path* parameter.

57938 RETURN VALUE

57939 CX Upon successful completion, the `rename()` function shall return 0. Otherwise, it shall return `-1`,
 57940 `errno` shall be set to indicate the error, and neither the file named by *old* nor the file named by
 57941 *new* shall be changed or created.

57942 CX Upon successful completion, the `renameat()` function shall return 0. Otherwise, it shall return `-1`
 57943 and set `errno` to indicate the error.

57944 ERRORS

57945 CX The `rename()` and `renameat()` functions shall fail if:

57946 CX [EACCES] A component of either path prefix denies search permission; or one of the
 57947 directories containing *old* or *new* denies write permissions; or, write
 57948 permission is required and is denied for a directory pointed to by the *old* or
 57949 *new* arguments.

57950 CX [EBUSY] The directory named by *old* or *new* is currently in use by the system or another
 57951 process, and the implementation considers this an error.

57952 CX [EEXIST] or [ENOTEMPTY]
 57953 The link named by *new* is a directory that is not an empty directory.

57954 CX [EINVAL] The *old* pathname names an ancestor directory of the *new* pathname, or either
 57955 pathname argument contains a final component that is dot or dot-dot.

57956 CX [EIO] A physical I/O error has occurred.

57957 CX [EISDIR] The *new* argument points to a directory and the *old* argument points to a file
 57958 that is not a directory.

57959 CX [ELOOP] A loop exists in symbolic links encountered during resolution of the *path*
 57960 argument.

57961 CX [EMLINK] The file named by *old* is a directory, and the link count of the parent directory
 57962 of *new* would exceed {LINK_MAX}.

57963 CX [ENAMETOOLONG]
 57964 The length of a component of a pathname is longer than {NAME_MAX}.

57965 CX [ENOENT] The link named by *old* does not name an existing file, a component of the path
 57966 prefix of *new* does not exist, or either *old* or *new* points to an empty string.

57967	CX	[ENOSPC]	The directory that would contain <i>new</i> cannot be extended.	
57968	CX	[ENOTDIR]	A component of either path prefix names an existing file that is neither a directory nor a symbolic link to a directory; or the <i>old</i> argument names a directory and the <i>new</i> argument names a non-directory file; or the <i>old</i> argument contains at least one non- <code><slash></code> character and ends with one or more trailing <code><slash></code> characters and the last pathname component names an existing file that is neither a directory nor a symbolic link to a directory; or the <i>old</i> argument names an existing non-directory file and the <i>new</i> argument names a nonexistent file, contains at least one non- <code><slash></code> character, and ends with one or more trailing <code><slash></code> characters; or the <i>new</i> argument names an existing non-directory file, contains at least one non- <code><slash></code> character, and ends with one or more trailing <code><slash></code> characters.	
57969				
57970				
57971				
57972				
57973				
57974				+
57975				+
57976				
57977				
57978				
57979	XSI	[EPERM] or [EACCES]	The <code>S_ISVTX</code> flag is set on the directory containing the file referred to by <i>old</i> and the process does not satisfy the criteria specified in XBD Section 4.2 (on page 107) with respect to <i>old</i> ; or <i>new</i> refers to an existing file, the <code>S_ISVTX</code> flag is set on the directory containing this file, and the process does not satisfy the criteria specified in XBD Section 4.2 with respect to this file.	
57980				
57981				
57982				
57983				
57984				
57985	CX	[EROFS]	The requested operation requires writing in a directory on a read-only file system.	
57986				
57987	CX	[EXDEV]	The links named by <i>new</i> and <i>old</i> are on different file systems and the implementation does not support links between file systems.	
57988				
57989	CX	In addition, the <code>renameat()</code> function shall fail if:		
57990		[EACCES]	<i>oldfd</i> or <i>newfd</i> was not opened with <code>O_SEARCH</code> and the permissions of the directory underlying <i>oldfd</i> or <i>newfd</i> respectively do not permit directory searches.	
57991				
57992				
57993		[EBADF]	The <i>old</i> argument does not specify an absolute path and the <i>oldfd</i> argument is neither <code>AT_FDCWD</code> nor a valid file descriptor open for reading or searching, or the <i>new</i> argument does not specify an absolute path and the <i>newfd</i> argument is neither <code>AT_FDCWD</code> nor a valid file descriptor open for reading or searching.	
57994				
57995				
57996				
57997				
57998		[ENOTDIR]	The <i>old</i> or <i>new</i> argument is not an absolute path and <i>oldfd</i> or <i>newfd</i> , respectively, is a file descriptor associated with a non-directory file.	+
57999				
58000	CX	The <code>rename()</code> and <code>renameat()</code> functions may fail if:		
58001	OB XSR	[EBUSY]	The file named by the <i>old</i> or <i>new</i> arguments is a named STREAM.	
58002	CX	[ELOOP]	More than <code>{SYMLOOP_MAX}</code> symbolic links were encountered during resolution of the <i>path</i> argument.	
58003				
58004	CX	[ENAMETOOLONG]	The length of a pathname exceeds <code>{PATH_MAX}</code> , or pathname resolution of a symbolic link produced an intermediate result with a length that exceeds <code>{PATH_MAX}</code> .	
58005				
58006				
58007				
58008	CX	[ETXTBSY]	The file named by <i>new</i> exists and is the last directory entry to a pure procedure (shared text) file that is being executed.	
58009				

EXAMPLES**Renaming a File**

The following example shows how to rename a file named `/home/cnd/mod1` to `/home/cnd/mod2`.

```
#include <stdio.h>

int status;
...
status = rename("/home/cnd/mod1", "/home/cnd/mod2");
```

APPLICATION USAGE

Some implementations mark for update the last file status change timestamp of renamed files and some do not. Applications which make use of the last file status change timestamp may behave differently with respect to renamed files unless they are designed to allow for either behavior.

RATIONALE

This `rename()` function is equivalent for regular files to that defined by the ISO C standard. Its inclusion here expands that definition to include actions on directories and specifies behavior when the *new* parameter names a file that already exists. That specification requires that the action of the function be atomic.

One of the reasons for introducing this function was to have a means of renaming directories while permitting implementations to prohibit the use of `link()` and `unlink()` with directories, thus constraining links to directories to those made by `mkdir()`.

The specification that if *old* and *new* refer to the same file is intended to guarantee that:

```
rename("x", "x");
```

does not remove the file.

Renaming dot or dot-dot is prohibited in order to prevent cyclical file system paths.

See also the descriptions of [ENOTEMPTY] and [ENAMETOOLONG] in `rmdir()` and [EBUSY] in `unlink()`. For a discussion of [EXDEV], see `link()`.

The purpose of the `renameat()` function is to rename files in directories other than the current working directory without exposure to race conditions. Any part of the path of a file could be changed in parallel to a call to `rename()`, resulting in unspecified behavior. By opening file descriptors for the source and target directories and using the `renameat()` function it can be guaranteed that that renamed file is located correctly and the resulting file is in the desired directory.

FUTURE DIRECTIONS

None.

SEE ALSO

[`link\(\)`](#), [`rmdir\(\)`](#), [`symlink\(\)`](#), [`unlink\(\)`](#)

XBD [Section 4.2](#) (on page 107), [`<stdio.h>`](#)

CHANGE HISTORY

First released in Issue 3. Included for alignment with the POSIX.1-1988 standard.

Issue 5

The [EBUSY] error is added to the optional part of the ERRORS section.

Issue 6

Extensions beyond the ISO C standard are marked.

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The [EIO] mandatory error condition is added.
- The [ELOOP] mandatory error condition is added.
- A second [ENAMETOOLONG] is added as an optional error condition.
- The [ETXTBSY] optional error condition is added.

The following changes were made to align with the IEEE P1003.1a draft standard:

- Details are added regarding the treatment of symbolic links.
- The [ELOOP] optional error condition is added.

The normative text is updated to avoid use of the term “must” for application requirements.

Issue 7

Austin Group Interpretation 1003.1-2001 #016 is applied, changing the definition of the [ENOTDIR] error.

Austin Group Interpretation 1003.1-2001 #076 is applied, clarifying the behavior if the final component of a path is either dot or dot-dot, and adding the associated [EINVAL] error case.

Austin Group Interpretation 1003.1-2001 #143 is applied.

Austin Group Interpretation 1003.1-2001 #145 is applied, clarifying that the [ENOENT] error condition also applies to the case in which a component of *new* does not exist.

Austin Group Interpretations 1003.1-2001 #174 and #181 are applied.

The *renameat()* function is added from The Open Group Technical Standard, 2006, Extended API Set Part 2.

Changes are made related to support for finegrained timestamps.

Changes are made to allow a directory to be opened for searching.

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0508 [324], XSH/TC1-2008/0509 [147], + XSH/TC1-2008/0510 [379], XSH/TC1-2008/0511 [278], and XSH/TC1-2008/0512 [278] are + applied.

58080 **NAME**

58081 rewind — reset the file position indicator in a stream

58082 **SYNOPSIS**

58083 #include <stdio.h>

58084 void rewind(FILE *stream);

58085 **DESCRIPTION**

58086 CX The functionality described on this reference page is aligned with the ISO C standard. Any
 58087 conflict between the requirements described here and the ISO C standard is unintentional. This
 58088 volume of POSIX.1-2008 defers to the ISO C standard.

58089 The call:

58090 rewind(stream)

58091 shall be equivalent to:

58092 (void) fseek(stream, 0L, SEEK_SET)

58093 except that *rewind()* shall also clear the error indicator.

58094 CX Since *rewind()* does not return a value, an application wishing to detect errors should clear *errno*,
 58095 then call *rewind()*, and if *errno* is non-zero, assume an error has occurred.

58096 **RETURN VALUE**58097 The *rewind()* function shall not return a value.58098 **ERRORS**58099 CX Refer to *fseek()* with the exception of [EINVAL] which does not apply.58100 **EXAMPLES**

58101 None.

58102 **APPLICATION USAGE**

58103 None.

58104 **RATIONALE**

58105 None.

58106 **FUTURE DIRECTIONS**

58107 None.

58108 **SEE ALSO**58109 Section 2.5 (on page 495), *fseek()*

58110 XBD <stdio.h>

58111 **CHANGE HISTORY**

58112 First released in Issue 1. Derived from Issue 1 of the SVID.

58113 **Issue 6**

58114 Extensions beyond the ISO C standard are marked.

58115 **Issue 7**

58116 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0513 [14] is applied.

+

58117 **NAME**

58118 rewinddir — reset the position of a directory stream to the beginning of a directory

58119 **SYNOPSIS**

58120 #include <dirent.h>

58121 void rewinddir(DIR *dirp);

58122 **DESCRIPTION**

58123 The *rewinddir()* function shall reset the position of the directory stream to which *dirp* refers to the
 58124 beginning of the directory. It shall also cause the directory stream to refer to the current state of
 58125 the corresponding directory, as a call to *opendir()* would have done. If *dirp* does not refer to a
 58126 directory stream, the effect is undefined.

58127 After a call to the *fork()* function, either the parent or child (but not both) may continue
 58128 XSI processing the directory stream using *readdir()*, *rewinddir()*, or *seekdir()*. If both the parent and
 58129 child processes use these functions, the result is undefined.

58130 **RETURN VALUE**58131 The *rewinddir()* function shall not return a value.58132 **ERRORS**

58133 No errors are defined.

58134 **EXAMPLES**

58135 None.

58136 **APPLICATION USAGE**

58137 The *rewinddir()* function should be used in conjunction with *opendir()*, *readdir()*, and *closedir()* to
 58138 examine the contents of the directory. This method is recommended for portability.

58139 **RATIONALE**

58140 None.

58141 **FUTURE DIRECTIONS**

58142 None.

58143 **SEE ALSO**58144 *closedir()*, *fdopendir()*, *readdir()*

58145 XBD <dirent.h>, <sys/types.h>

58146 **CHANGE HISTORY**

58147 First released in Issue 2.

58148 **Issue 6**

58149 In the SYNOPSIS, the optional include of the <sys/types.h> header is removed.

58150 The following new requirements on POSIX implementations derive from alignment with the
 58151 Single UNIX Specification:

- 58152 • The requirement to include <sys/types.h> has been removed. Although <sys/types.h> was
 58153 required for conforming implementations of previous POSIX specifications, it was not
 58154 required for UNIX applications.

58155 **NAME**

58156 rint, rintf, rintl — round-to-nearest integral value

58157 **SYNOPSIS**

```
58158 #include <math.h>
58159 double rint(double x);
58160 float rintf(float x);
58161 long double rintl(long double x);
```

58162 **DESCRIPTION**

58163 CX The functionality described on this reference page is aligned with the ISO C standard. Any
 58164 conflict between the requirements described here and the ISO C standard is unintentional. This
 58165 volume of POSIX.1-2008 defers to the ISO C standard.

58166 These functions shall return the integral value (represented as a **double**) nearest x in the
 58167 direction of the current rounding mode. The current rounding mode is implementation-defined.

58168 If the current rounding mode rounds toward negative infinity, then *rint()* shall be equivalent to
 58169 *floor()*. If the current rounding mode rounds toward positive infinity, then *rint()* shall be
 58170 equivalent to *ceil()*. If the current rounding mode rounds towards zero, then *rint()* shall be
 58171 MX equivalent to *trunc()*. If the current rounding mode rounds towards nearest, then *rint()* differs
 58172 from *round()* in that halfway cases are rounded to even rather than away from zero.

58173 These functions differ from the *nearbyint()*, *nearbyintf()*, and *nearbyintl()* functions only in that
 58174 they may raise the inexact floating-point exception if the result differs in value from the
 58175 argument.

58176 An application wishing to check for error situations should set *errno* to zero and call
 58177 *feclearexcept*(FE_ALL_EXCEPT) before calling these functions. On return, if *errno* is non-zero or
 58178 *fetestexcept*(FE_INVALID | FE_DIVBYZERO | FE_OVERFLOW | FE_UNDERFLOW) is non-
 58179 zero, an error has occurred.

58180 **RETURN VALUE**

58181 Upon successful completion, these functions shall return the integer (represented as a double
 58182 MX precision number) nearest x in the direction of the current rounding mode. The result shall have
 58183 the same sign as x .

58184 MX If x is NaN, a NaN shall be returned.

58185 If x is ± 0 or $\pm \text{Inf}$, x shall be returned.

58186 **ERRORS**

58187 No errors are defined.

58188 **EXAMPLES**

58189 None.

58190 **APPLICATION USAGE**

58191 The integral value returned by these functions need not be expressible as an **intmax_t**. The
 58192 return value should be tested before assigning it to an integer type to avoid the undefined
 58193 results of an integer overflow.

58194 **RATIONALE**

58195 None.

58196 **FUTURE DIRECTIONS**

58197 None.

58198 **SEE ALSO**58199 *abs()*, *ceil()*, *feclearexcept()*, *fetestexcept()*, *floor()*, *isnan()*, *nearbyint()*58200 XBD [Section 4.19](#) (on page 116), **<math.h>**58201 **CHANGE HISTORY**

58202 First released in Issue 4, Version 2.

58203 **Issue 5**

58204 Moved from X/OPEN UNIX extension to BASE.

58205 **Issue 6**

58206 The following changes are made for alignment with the ISO/IEC 9899:1999 standard:

- 58207 • The *rintf()* and *rintl()* functions are added.
- 58208 • The *rint()* function is no longer marked as an extension.
- 58209 • The DESCRIPTION, RETURN VALUE, ERRORS, and APPLICATION USAGE sections are
- 58210 revised to align with the ISO/IEC 9899:1999 standard.
- 58211 IEC 60559:1989 standard floating-point extensions over the ISO/IEC 9899:1999 standard
- 58212 are marked.

58213 **Issue 7**

58214 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0514 [346], XSH/TC1-2008/0515 [346], +
 58215 XSH/TC1-2008/0516 [346], XSH/TC1-2008/0517 [346], and XSH/TC1-2008/0518 [346] are +
 58216 applied.

NAME

rmdir — remove a directory

SYNOPSIS

```
#include <unistd.h>
```

```
int rmdir(const char *path);
```

DESCRIPTION

The *rmdir()* function shall remove a directory whose name is given by *path*. The directory shall be removed only if it is an empty directory.

If the directory is the root directory or the current working directory of any process, it is unspecified whether the function succeeds, or whether it shall fail and set *errno* to [EBUSY].

If *path* names a symbolic link, then *rmdir()* shall fail and set *errno* to [ENOTDIR].

If the *path* argument refers to a path whose final component is either dot or dot-dot, *rmdir()* shall fail.

If the directory's link count becomes 0 and no process has the directory open, the space occupied by the directory shall be freed and the directory shall no longer be accessible. If one or more processes have the directory open when the last link is removed, the dot and dot-dot entries, if present, shall be removed before *rmdir()* returns and no new entries may be created in the directory, but the directory shall not be removed until all references to the directory are closed.

If the directory is not an empty directory, *rmdir()* shall fail and set *errno* to [EEXIST] or [ENOTEMPTY].

Upon successful completion, *rmdir()* shall mark for update the last data modification and last file status change timestamps of the parent directory.

RETURN VALUE

Upon successful completion, the function *rmdir()* shall return 0. Otherwise, -1 shall be returned, and *errno* set to indicate the error. If -1 is returned, the named directory shall not be changed.

ERRORS

The *rmdir()* function shall fail if:

[EACCES] Search permission is denied on a component of the path prefix, or write permission is denied on the parent directory of the directory to be removed.

[EBUSY] The directory to be removed is currently in use by the system or some process and the implementation considers this to be an error.

[EEXIST] or [ENOTEMPTY]

The *path* argument names a directory that is not an empty directory, or there are hard links to the directory other than dot or a single entry in dot-dot.

[EINVAL] The *path* argument contains a last component that is dot.

[EIO] A physical I/O error has occurred.

[ELOOP] A loop exists in symbolic links encountered during resolution of the *path* argument.

[ENAMETOOLONG]

The length of a component of a pathname is longer than {NAME_MAX}.

[ENOENT] A component of *path* does not name an existing file, or the *path* argument names a nonexistent directory or points to an empty string.

58259 [ENOTDIR] A component of *path* names an existing file that is neither a directory nor a
 58260 symbolic link to a directory.

58261 XSI [EPERM] or [EACCES]
 58262 The S_ISVTX flag is set on the directory containing the file referred to by the
 58263 *path* argument and the process does not satisfy the criteria specified in XBD
 58264 [Section 4.2](#) (on page 107).

58265 [EROFS] The directory entry to be removed resides on a read-only file system.

58266 The `rmdir()` function may fail if:

58267 [ELOOP] More than {SYMLOOP_MAX} symbolic links were encountered during
 58268 resolution of the *path* argument.

58269 [ENAMETOOLONG]
 58270 The length of a pathname exceeds {PATH_MAX}, or pathname resolution of a
 58271 symbolic link produced an intermediate result with a length that exceeds
 58272 {PATH_MAX}.

58273 EXAMPLES

58274 Removing a Directory

58275 The following example shows how to remove a directory named `/home/cnd/mod1`.

```
58276 #include <unistd.h>
58277 int status;
58278 ...
58279 status = rmdir("/home/cnd/mod1");
```

58280 APPLICATION USAGE

58281 None.

58282 RATIONALE

58283 The `rmdir()` and `rename()` functions originated in 4.2 BSD, and they used [ENOTEMPTY] for the
 58284 condition when the directory to be removed does not exist or *new* already exists. When the 1984
 58285 /usr/group standard was published, it contained [EEXIST] instead. When these functions were
 58286 adopted into System V, the 1984 /usr/group standard was used as a reference. Therefore,
 58287 several existing applications and implementations support/use both forms, and no agreement
 58288 could be reached on either value. All implementations are required to supply both [EEXIST] and
 58289 [ENOTEMPTY] in `<errno.h>` with distinct values, so that applications can use both values in C-
 58290 language **case** statements.

58291 The meaning of deleting *pathname/dot* is unclear, because the name of the file (directory) in the
 58292 parent directory to be removed is not clear, particularly in the presence of multiple links to a
 58293 directory.

58294 The POSIX.1-1990 standard was silent with regard to the behavior of `rmdir()` when there are
 58295 multiple hard links to the directory being removed. The requirement to set *errno* to [EEXIST] or
 58296 [ENOTEMPTY] clarifies the behavior in this case.

58297 If the current working directory of the process is being removed, that should be an allowed
 58298 error.

58299 Virtually all existing implementations detect [ENOTEMPTY] or the case of dot-dot. The text in
 58300 [Section 2.3](#) (on page 481) about returning any one of the possible errors permits that behavior to
 58301 continue. The [ELOOP] error may be returned if more than {SYMLOOP_MAX} symbolic links

58302 are encountered during resolution of the *path* argument.

58303 FUTURE DIRECTIONS

58304 None.

58305 SEE ALSO

58306 [Section 2.3](#) (on page 481), *mkdir()*, *remove()*, *rename()*, *unlink()*

58307 XBD [Section 4.2](#) (on page 107), [<unistd.h>](#)

58308 CHANGE HISTORY

58309 First released in Issue 3. Included for alignment with the POSIX.1-1988 standard.

58310 Issue 6

58311 The following new requirements on POSIX implementations derive from alignment with the
58312 Single UNIX Specification:

- 58313 • The DESCRIPTION is updated to indicate the results of naming a symbolic link in *path*.
- 58314 • The [EIO] mandatory error condition is added.
- 58315 • The [ELOOP] mandatory error condition is added.
- 58316 • A second [ENAMETOOLONG] is added as an optional error condition.

58317 The following changes were made to align with the IEEE P1003.1a draft standard:

- 58318 • The [ELOOP] optional error condition is added.

58319 Issue 7

58320 Austin Group Interpretation 1003.1-2001 #143 is applied.

58321 Austin Group Interpretation 1003.1-2001 #181 is applied, updating the requirements for
58322 operations when the S_ISVTX bit is set.

58323 Changes are made related to support for finegrained timestamps.

58324 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0519 [324] is applied.

+

58325 **NAME**

58326 round, roundf, roundl — round to the nearest integer value in a floating-point format

58327 **SYNOPSIS**

```
58328 #include <math.h>
58329 double round(double x);
58330 float roundf(float x);
58331 long double roundl(long double x);
```

58332 **DESCRIPTION**

58333 CX The functionality described on this reference page is aligned with the ISO C standard. Any
 58334 conflict between the requirements described here and the ISO C standard is unintentional. This
 58335 volume of POSIX.1-2008 defers to the ISO C standard.

58336 These functions shall round their argument to the nearest integer value in floating-point format,
 58337 rounding halfway cases away from zero, regardless of the current rounding direction. -

58338 **RETURN VALUE**

58339 MX Upon successful completion, these functions shall return the rounded integer value. The result
 58340 shall have the same sign as x . +

58341 MX If x is NaN, a NaN shall be returned.

58342 If x is ± 0 or $\pm \text{Inf}$, x shall be returned. -

58343 **ERRORS**

58344 No errors are defined. |

58345 **EXAMPLES**

58346 None.

58347 **APPLICATION USAGE**

58348 The integral value returned by these functions need not be expressible as an `intmax_t`. The
 58349 return value should be tested before assigning it to an integer type to avoid the undefined
 58350 results of an integer overflow. |

58351 These functions may raise the inexact floating-point exception if the result differs in value from
 58352 the argument. |

58353 **RATIONALE**

58354 None.

58355 **FUTURE DIRECTIONS**

58356 None.

58357 **SEE ALSO**

58358 [*feclearexcept\(\)*](#), [*fetestexcept\(\)*](#)

58359 XBD [Section 4.19](#) (on page 116), [`<math.h>`](#)

58360 **CHANGE HISTORY**

58361 First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

58362 **Issue 7**

58363 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0520 [346] is applied. +

58364 **NAME**

58365 scalbln, scalblnf, scalblnl, scalbn, scalbnf, scalbnl — compute exponent using FLT_RADIX

58366 **SYNOPSIS**

```
58367 #include <math.h>
58368 double scalbln(double x, long n);
58369 float scalblnf(float x, long n);
58370 long double scalblnl(long double x, long n);
58371 double scalbn(double x, int n);
58372 float scalbnf(float x, int n);
58373 long double scalbnl(long double x, int n);
```

58374 **DESCRIPTION**

58375 CX The functionality described on this reference page is aligned with the ISO C standard. Any
 58376 conflict between the requirements described here and the ISO C standard is unintentional. This
 58377 volume of POSIX.1-2008 defers to the ISO C standard.

58378 These functions shall compute $x * \text{FLT\_RADIX}^n$ efficiently, not normally by computing
 58379 FLT_RADIX^n explicitly.

58380 An application wishing to check for error situations should set *errno* to zero and call
 58381 *feclearexcept*(FE_ALL_EXCEPT) before calling these functions. On return, if *errno* is non-zero or
 58382 *fetestexcept*(FE_INVALID | FE_DIVBYZERO | FE_OVERFLOW | FE_UNDERFLOW) is non-
 58383 zero, an error has occurred.

58384 **RETURN VALUE**

58385 Upon successful completion, these functions shall return $x * \text{FLT\_RADIX}^n$.

58386 If the result would cause overflow, a range error shall occur and these functions shall return
 58387 $\pm\text{HUGE\_VAL}$, $\pm\text{HUGE\_VALF}$, and $\pm\text{HUGE\_VALL}$ (according to the sign of *x*) as appropriate for
 58388 the return type of the function.

58389 MXX If the correct value would cause underflow, and is not representable, a range error may occur, |
 58390 MXX and *scalbln*(), *scalblnf*(), *scalblnl*(), *scalbn*(), *scalbnf*(), and *scalbnl*() shall return 0.0, or (if IEC |
 58391 60559 Floating-Point is not supported) an implementation-defined value no greater in |
 58392 magnitude than DBL_MIN, FLT_MIN, LDBL_MIN, DBL_MIN, FLT_MIN, and LDBL_MIN, |
 58393 respectively.

58394 MX If *x* is NaN, a NaN shall be returned.

58395 If *x* is ± 0 or $\pm\text{Inf}$, *x* shall be returned.

58396 If *n* is 0, *x* shall be returned.

58397 MXX If the correct value would cause underflow, and is representable, a range error may occur and
 58398 the correct value shall be returned.

58399 **ERRORS**

58400 These functions shall fail if:

58401 Range Error The result overflows.

58402 If the integer expression (*math_errhandling* & MATH_ERRNO) is non-zero,
 58403 then *errno* shall be set to [ERANGE]. If the integer expression
 58404 (*math_errhandling* & MATH_ERREXCEPT) is non-zero, then the overflow
 58405 floating-point exception shall be raised.

58406 These functions may fail if:

58407 Range Error The result underflows.

58408 If the integer expression (*math_errhandling* & MATH_ERRNO) is non-zero,
 58409 then *errno* shall be set to [ERANGE]. If the integer expression
 58410 (*math_errhandling* & MATH_ERREXCEPT) is non-zero, then the underflow
 58411 floating-point exception shall be raised.

58412 EXAMPLES

58413 None.

58414 APPLICATION USAGE

58415 On error, the expressions (*math_errhandling* & MATH_ERRNO) and (*math_errhandling* &
 58416 MATH_ERREXCEPT) are independent of each other, but at least one of them must be non-zero.

58417 RATIONALE

58418 These functions are named so as to avoid conflicting with the historical definition of the *scalb()*
 58419 function from the Single UNIX Specification. The difference is that the *scalb()* function has a
 58420 second argument of **double** instead of **int**. The *scalb()* function is not part of the ISO C standard.
 58421 The three functions whose second type is **long** are provided because the factor required to scale
 58422 from the smallest positive floating-point value to the largest finite one, on many
 58423 implementations, is too large to represent in the minimum-width **int** format.

58424 FUTURE DIRECTIONS

58425 None.

58426 SEE ALSO

58427 *feclearexcept()*, *fetestexcept()*

58428 XBD Section 4.19 (on page 116), **<math.h>**

58429 CHANGE HISTORY

58430 First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

58431 Issue 7

58432 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0521 [68] and XSH/TC1-2008/0522 +
 58433 [68] are applied.

58434 **NAME**

58435 scandir — scan a directory

58436 **SYNOPSIS**

58437 #include <dirent.h>

```
58438     int scandir(const char *dir, struct dirent ***namelist,  
58439                int (*sel)(const struct dirent *),  
58440                int (*compar)(const struct dirent **, const struct dirent **));
```

58441 **DESCRIPTION**58442 Refer to *alphasort()*.

58443 **NAME**

58444 scanf — convert formatted input

58445 **SYNOPSIS**

58446 #include <stdio.h>

58447 int scanf(const char *restrict *format*, ...);58448 **DESCRIPTION**58449 Refer to *fprintf()*.

NAME

sched_get_priority_max, sched_get_priority_min — get priority limits (**REALTIME**)

SYNOPSIS

```
PS|TPS    #include <sched.h>

int sched_get_priority_max(int policy);
int sched_get_priority_min(int policy);
```

DESCRIPTION

The *sched_get_priority_max()* and *sched_get_priority_min()* functions shall return the appropriate maximum or minimum, respectively, for the scheduling policy specified by *policy*.

The value of *policy* shall be one of the scheduling policy values defined in **<sched.h>**.

RETURN VALUE

If successful, the *sched_get_priority_max()* and *sched_get_priority_min()* functions shall return the appropriate maximum or minimum values, respectively. If unsuccessful, they shall return a value of -1 and set *errno* to indicate the error.

ERRORS

The *sched_get_priority_max()* and *sched_get_priority_min()* functions shall fail if:

[EINVAL]	The value of the <i>policy</i> parameter does not represent a defined scheduling policy.

EXAMPLES

None.

APPLICATION USAGE

None.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

sched_getparam(), *sched_setparam()*, *sched_getscheduler()*, *sched_rr_get_interval()*, *sched_setscheduler()*

XBD **<sched.h>**

CHANGE HISTORY

First released in Issue 5. Included for alignment with the POSIX Realtime Extension.

Issue 6

These functions are marked as part of the Process Scheduling option.

The [ENOSYS] error condition has been removed as stubs need not be provided if an implementation does not support the Process Scheduling option.

The [ESRCH] error condition has been removed since these functions do not take a *pid* argument.

IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/52 is applied, changing the PS margin code in the SYNOPSIS to PS|TPS.

NAME

sched_getparam — get scheduling parameters (**REALTIME**)

SYNOPSIS

```
#include <sched.h>

int sched_getparam(pid_t pid, struct sched_param *param);
```

DESCRIPTION

The *sched_getparam()* function shall return the scheduling parameters of a process specified by *pid* in the **sched_param** structure pointed to by *param*.

If a process specified by *pid* exists, and if the calling process has permission, the scheduling parameters for the process whose process ID is equal to *pid* shall be returned.

If *pid* is zero, the scheduling parameters for the calling process shall be returned. The behavior of the *sched_getparam()* function is unspecified if the value of *pid* is negative.

RETURN VALUE

Upon successful completion, the *sched_getparam()* function shall return zero. If the call to *sched_getparam()* is unsuccessful, the function shall return a value of -1 and set *errno* to indicate the error.

ERRORS

The *sched_getparam()* function shall fail if:

[EPERM] The requesting process does not have permission to obtain the scheduling parameters of the specified process.

[ESRCH] No process can be found corresponding to that specified by *pid*.

EXAMPLES

None.

APPLICATION USAGE

None.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

sched_getscheduler(), *sched_setparam()*, *sched_setscheduler()*

XBD **<sched.h>**

CHANGE HISTORY

First released in Issue 5. Included for alignment with the POSIX Realtime Extension.

Issue 6

The *sched_getparam()* function is marked as part of the Process Scheduling option.

The [ENOSYS] error condition has been removed as stubs need not be provided if an implementation does not support the Process Scheduling option.

NAME

sched_getscheduler — get scheduling policy (**REALTIME**)

SYNOPSIS

```
#include <sched.h>

int sched_getscheduler(pid_t pid);
```

DESCRIPTION

The *sched_getscheduler()* function shall return the scheduling policy of the process specified by *pid*. If the value of *pid* is negative, the behavior of the *sched_getscheduler()* function is unspecified.

The values that can be returned by *sched_getscheduler()* are defined in the **<sched.h>** header.

If a process specified by *pid* exists, and if the calling process has permission, the scheduling policy shall be returned for the process whose process ID is equal to *pid*.

If *pid* is zero, the scheduling policy shall be returned for the calling process.

RETURN VALUE

Upon successful completion, the *sched_getscheduler()* function shall return the scheduling policy of the specified process. If unsuccessful, the function shall return -1 and set *errno* to indicate the error.

ERRORS

The *sched_getscheduler()* function shall fail if:

[EPERM]	The requesting process does not have permission to determine the scheduling policy of the specified process.
[ESRCH]	No process can be found corresponding to that specified by <i>pid</i> .

EXAMPLES

None.

APPLICATION USAGE

None.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

[*sched_getparam\(\)*](#), [*sched_setparam\(\)*](#), [*sched_setscheduler\(\)*](#)

XBD **<sched.h>**

CHANGE HISTORY

First released in Issue 5. Included for alignment with the POSIX Realtime Extension.

Issue 6

The *sched_getscheduler()* function is marked as part of the Process Scheduling option.

The [ENOSYS] error condition has been removed as stubs need not be provided if an implementation does not support the Process Scheduling option.

NAME

sched_rr_get_interval — get execution time limits (**REALTIME**)

SYNOPSIS

```
PS|TPS    #include <sched.h>
```

```
int sched_rr_get_interval(pid_t pid, struct timespec *interval);
```

DESCRIPTION

The *sched_rr_get_interval()* function shall update the **timespec** structure referenced by the *interval* argument to contain the current execution time limit (that is, time quantum) for the process specified by *pid*. If *pid* is zero, the current execution time limit for the calling process shall be returned.

RETURN VALUE

If successful, the *sched_rr_get_interval()* function shall return zero. Otherwise, it shall return a value of -1 and set *errno* to indicate the error.

ERRORS

The *sched_rr_get_interval()* function shall fail if:

[ESRCH] No process can be found corresponding to that specified by *pid*.

EXAMPLES

None.

APPLICATION USAGE

None.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

sched_getparam(), *sched_get_priority_max()*, *sched_getscheduler()*, *sched_setparam()*, *sched_setscheduler()*

XBD **<sched.h>**

CHANGE HISTORY

First released in Issue 5. Included for alignment with the POSIX Realtime Extension.

Issue 6

The *sched_rr_get_interval()* function is marked as part of the Process Scheduling option.

The [ENOSYS] error condition has been removed as stubs need not be provided if an implementation does not support the Process Scheduling option.

IEEE Std 1003.1-2001/Cor 1-2002, XSH/TC1/D6/53 is applied, changing the PS margin code in the SYNOPSIS to PS|TPS.

NAME

sched_setparam — set scheduling parameters (**REALTIME**)

SYNOPSIS

```

#include <sched.h>

int sched_setparam(pid_t pid, const struct sched_param *param);

```

DESCRIPTION

The *sched_setparam()* function shall set the scheduling parameters of the process specified by *pid* to the values specified by the **sched_param** structure pointed to by *param*. The value of the *sched_priority* member in the **sched_param** structure shall be any integer within the inclusive priority range for the current scheduling policy of the process specified by *pid*. Higher numerical values for the priority represent higher priorities. If the value of *pid* is negative, the behavior of the *sched_setparam()* function is unspecified.

If a process specified by *pid* exists, and if the calling process has permission, the scheduling parameters shall be set for the process whose process ID is equal to *pid*.

If *pid* is zero, the scheduling parameters shall be set for the calling process.

The conditions under which one process has permission to change the scheduling parameters of another process are implementation-defined.

Implementations may require the requesting process to have appropriate privileges to set its own scheduling parameters or those of another process.

See [Scheduling Policies](#) (on page 506) for a description on how this function affects the scheduling of the threads within the target process.

If the current scheduling policy for the target process is not **SCHED_FIFO**, **SCHED_RR**, or **SCHED_SPORADIC**, the result is implementation-defined; this case includes the **SCHED_OTHER** policy.

The specified *sched_ss_repl_period* shall be greater than or equal to the specified *sched_ss_init_budget* for the function to succeed; if it is not, then the function shall fail.

The value of *sched_ss_max_repl* shall be within the inclusive range [1,{**SS_REPL_MAX**}] for the function to succeed; if not, the function shall fail. It is unspecified whether the *sched_ss_repl_period* and *sched_ss_init_budget* values are stored as provided by this function or are rounded to align with the resolution of the clock being used.

This function is not atomic with respect to other threads in the process. Threads may continue to execute while this function call is in the process of changing the scheduling policy for the underlying kernel-scheduled entities used by the process contention scope threads.

RETURN VALUE

If successful, the *sched_setparam()* function shall return zero.

If the call to *sched_setparam()* is unsuccessful, the priority shall remain unchanged, and the function shall return a value of **-1** and set *errno* to indicate the error.

ERRORS

The *sched_setparam()* function shall fail if:

[EINVAL] One or more of the requested scheduling parameters is outside the range defined for the scheduling policy of the specified *pid*.

58644 [EPERM] The requesting process does not have permission to set the scheduling
 58645 parameters for the specified process, or does not have appropriate privileges
 58646 to invoke *sched_setparam()*.

58647 [ESRCH] No process can be found corresponding to that specified by *pid*.

58648 EXAMPLES

58649 None.

58650 APPLICATION USAGE

58651 None.

58652 RATIONALE

58653 None.

58654 FUTURE DIRECTIONS

58655 None.

58656 SEE ALSO

58657 [Scheduling Policies](#) (on page 506), *sched_getparam()*, *sched_getscheduler()*, *sched_setscheduler()*

58658 XBD [<sched.h>](#)

58659 CHANGE HISTORY

58660 First released in Issue 5. Included for alignment with the POSIX Realtime Extension.

58661 Issue 6

58662 The *sched_setparam()* function is marked as part of the Process Scheduling option.

58663 The [ENOSYS] error condition has been removed as stubs need not be provided if an
 58664 implementation does not support the Process Scheduling option.

58665 The following new requirements on POSIX implementations derive from alignment with the
 58666 Single UNIX Specification:

58667 • In the DESCRIPTION, the effect of this function on a thread's scheduling parameters is
 58668 added.

58669 • Sections describing two-level scheduling and atomicity of the function are added.

58670 The SCHED_SPORADIC scheduling policy is added for alignment with IEEE Std 1003.1d-1999.

58671 IEEE PASC Interpretation 1003.1 #100 is applied.

58672 Issue 7

58673 Austin Group Interpretation 1003.1-2001 #061 is applied, updating the DESCRIPTION.

58674 Austin Group Interpretation 1003.1-2001 #119 is applied, clarifying the accuracy requirements
 58675 for the *sched_ss_repl_period* and *sched_ss_init_budget* values.

NAME

sched_setscheduler — set scheduling policy and parameters (**REALTIME**)

SYNOPSIS

```
#include <sched.h>

int sched_setscheduler(pid_t pid, int policy,
    const struct sched_param *param);
```

DESCRIPTION

The *sched_setscheduler()* function shall set the scheduling policy and scheduling parameters of the process specified by *pid* to *policy* and the parameters specified in the **sched_param** structure pointed to by *param*, respectively. The value of the *sched_priority* member in the **sched_param** structure shall be any integer within the inclusive priority range for the scheduling policy specified by *policy*. If the value of *pid* is negative, the behavior of the *sched_setscheduler()* function is unspecified.

The possible values for the *policy* parameter are defined in the **<sched.h>** header.

If a process specified by *pid* exists, and if the calling process has permission, the scheduling policy and scheduling parameters shall be set for the process whose process ID is equal to *pid*.

If *pid* is zero, the scheduling policy and scheduling parameters shall be set for the calling process.

The conditions under which one process has appropriate privileges to change the scheduling parameters of another process are implementation-defined.

Implementations may require that the requesting process have permission to set its own scheduling parameters or those of another process. Additionally, implementation-defined restrictions may apply as to the appropriate privileges required to set the scheduling policy of the process, or the scheduling policy of another process, to a particular value.

The *sched_setscheduler()* function shall be considered successful if it succeeds in setting the scheduling policy and scheduling parameters of the process specified by *pid* to the values specified by *policy* and the structure pointed to by *param*, respectively.

See [Scheduling Policies](#) (on page 506) for a description on how this function affects the scheduling of the threads within the target process.

SS If the current scheduling policy for the target process is not SCHED_FIFO, SCHED_RR, or SCHED_SPORADIC, the result is implementation-defined; this case includes the SCHED_OTHER policy.

SS The specified *sched_ss_repl_period* shall be greater than or equal to the specified *sched_ss_init_budget* for the function to succeed; if it is not, then the function shall fail.

The value of *sched_ss_max_repl* shall be within the inclusive range [1,{SS_REPL_MAX}] for the function to succeed; if not, the function shall fail. It is unspecified whether the *sched_ss_repl_period* and *sched_ss_init_budget* values are stored as provided by this function or are rounded to align with the resolution of the clock being used.

This function is not atomic with respect to other threads in the process. Threads may continue to execute while this function call is in the process of changing the scheduling policy and associated scheduling parameters for the underlying kernel-scheduled entities used by the process contention scope threads.

RETURN VALUE

Upon successful completion, the function shall return the former scheduling policy of the specified process. If the *sched_setscheduler()* function fails to complete successfully, the policy and scheduling parameters shall remain unchanged, and the function shall return a value of -1 and set *errno* to indicate the error.

ERRORS

The *sched_setscheduler()* function shall fail if:

- [EINVAL] The value of the *policy* parameter is invalid, or one or more of the parameters contained in *param* is outside the valid range for the specified scheduling policy.
- [EPERM] The requesting process does not have permission to set either or both of the scheduling parameters or the scheduling policy of the specified process.
- [ESRCH] No process can be found corresponding to that specified by *pid*.

EXAMPLES

None.

APPLICATION USAGE

None.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

[Scheduling Policies](#) (on page 506), *sched_getparam()*, *sched_getscheduler()*, *sched_setparam()*

XBD [<sched.h>](#)

CHANGE HISTORY

First released in Issue 5. Included for alignment with the POSIX Realtime Extension.

Issue 6

The *sched_setscheduler()* function is marked as part of the Process Scheduling option.

The [ENOSYS] error condition has been removed as stubs need not be provided if an implementation does not support the Process Scheduling option.

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- In the DESCRIPTION, the effect of this function on a thread's scheduling parameters is added.
- Sections describing two-level scheduling and atomicity of the function are added.

The SCHED_SPORADIC scheduling policy is added for alignment with IEEE Std 1003.1d-1999.

Issue 7

Austin Group Interpretation 1003.1-2001 #061 is applied, updating the DESCRIPTION.

Austin Group Interpretation 1003.1-2001 #119 is applied, clarifying the accuracy requirements for the *sched_ss_repl_period* and *sched_ss_init_budget* values.

58758 **NAME**

58759 sched_yield — yield the processor

58760 **SYNOPSIS**

58761 #include <sched.h>

58762 int sched_yield(void);

58763 **DESCRIPTION**

58764 The *sched_yield()* function shall force the running thread to relinquish the processor until it again
58765 becomes the head of its thread list. It takes no arguments.

58766 **RETURN VALUE**

58767 The *sched_yield()* function shall return 0 if it completes successfully; otherwise, it shall return a
58768 value of -1 and set *errno* to indicate the error.

58769 **ERRORS**

58770 No errors are defined.

58771 **EXAMPLES**

58772 None.

58773 **APPLICATION USAGE**

58774 The conceptual model for scheduling semantics in POSIX.1-2008 defines a set of thread lists. This
58775 set of thread lists is always present regardless of the scheduling options supported by the
58776 system. On a system where the Process Scheduling option is not supported, portable
58777 applications should not make any assumptions regarding whether threads from other processes
58778 will be on the same thread list.

58779 **RATIONALE**

58780 None.

58781 **FUTURE DIRECTIONS**

58782 None.

58783 **SEE ALSO**

58784 XBD [<sched.h>](#)

58785 **CHANGE HISTORY**

58786 First released in Issue 5. Included for alignment with the POSIX Realtime Extension and the
58787 POSIX Threads Extension.

58788 **Issue 6**

58789 The *sched_yield()* function is now marked as part of the Process Scheduling and Threads options.

58790 **Issue 7**

58791 SD5-XSH-ERN-120 is applied, adding APPLICATION USAGE.

58792 The *sched_yield()* function is moved to the Base.

58793 **NAME**

58794 seed48 — seed a uniformly distributed pseudo-random non-negative long integer generator

58795 **SYNOPSIS**58796 XSI `#include <stdlib.h>`58797 `unsigned short *seed48(unsigned short seed16v[3]);`58798 **DESCRIPTION**58799 Refer to *drand48()*.

NAME

seekdir — set the position of a directory stream

SYNOPSIS

```
XSI      #include <dirent.h>
void seekdir(DIR *dirp, long loc);
```

DESCRIPTION

The *seekdir()* function shall set the position of the next *readdir()* operation on the directory stream specified by *dirp* to the position specified by *loc*. The value of *loc* should have been returned from an earlier call to *telldir()* using the same directory stream. The new position reverts to the one associated with the directory stream when *telldir()* was performed.

If the value of *loc* was not obtained from an earlier call to *telldir()*, or if a call to *rewinddir()* occurred between the call to *telldir()* and the call to *seekdir()*, the results of subsequent calls to *readdir()* are unspecified.

RETURN VALUE

The *seekdir()* function shall not return a value.

ERRORS

No errors are defined.

EXAMPLES

None.

APPLICATION USAGE

None.

RATIONALE

The original standard developers perceived that there were restrictions on the use of the *seekdir()* and *telldir()* functions related to implementation details, and for that reason these functions need not be supported on all POSIX-conforming systems. They are required on implementations supporting the XSI option.

One of the perceived problems of implementation is that returning to a given point in a directory is quite difficult to describe formally, in spite of its intuitive appeal, when systems that use B-trees, hashing functions, or other similar mechanisms to order their directories are considered. The definition of *seekdir()* and *telldir()* does not specify whether, when using these interfaces, a given directory entry will be seen at all, or more than once.

On systems not supporting these functions, their capability can sometimes be accomplished by saving a filename found by *readdir()* and later using *rewinddir()* and a loop on *readdir()* to relocate the position from which the filename was saved.

FUTURE DIRECTIONS

None.

SEE ALSO

fdopendir(), *readdir()*, *telldir()*

XBD *<dirent.h>*, *<sys/types.h>*

CHANGE HISTORY

First released in Issue 2.

58841 **Issue 6**

58842 In the SYNOPSIS, the inclusion of `<sys/types.h>` is no longer required.

58843 **Issue 7**

58844 SD5-XSH-ERN-200 is applied, updating the DESCRIPTION to note that the value of *loc* should
58845 have been returned from an earlier call to *telldir()* using the same directory stream.

58846 NAME

58847 select — synchronous I/O multiplexing

58848 SYNOPSIS

```
58849       #include <sys/select.h>

58850       int select(int nfds, fd_set *restrict readfds,
58851                fd_set *restrict writefds, fd_set *restrict errorfds,
58852                struct timeval *restrict timeout);
```

58853 DESCRIPTION

58854 Refer to *pselect()*.

58855 NAME

58856 sem_close — close a named semaphore

58857 SYNOPSIS

```
58858 #include <semaphore.h>
58859 int sem_close(sem_t *sem);
```

58860 DESCRIPTION

58861 The *sem_close()* function shall indicate that the calling process is finished using the named
 58862 semaphore indicated by *sem*. The effects of calling *sem_close()* for an unnamed semaphore (one
 58863 created by *sem_init()*) are undefined. The *sem_close()* function shall deallocate (that is, make
 58864 available for reuse by a subsequent *sem_open()* by this process) any system resources allocated
 58865 by the system for use by this process for this semaphore. The effect of subsequent use of the
 58866 semaphore indicated by *sem* by this process is undefined. If the semaphore has not been
 58867 removed with a successful call to *sem_unlink()*, then *sem_close()* has no effect on the state of the
 58868 semaphore. If the *sem_unlink()* function has been successfully invoked for *name* after the most
 58869 recent call to *sem_open()* with O_CREAT for this semaphore, then when all processes that have
 58870 opened the semaphore close it, the semaphore is no longer accessible.

58871 RETURN VALUE

58872 Upon successful completion, a value of zero shall be returned. Otherwise, a value of -1 shall be
 58873 returned and *errno* set to indicate the error.

58874 ERRORS

58875 The *sem_close()* function may fail if:

58876 [EINVAL] The *sem* argument is not a valid semaphore descriptor.

58877 EXAMPLES

58878 None.

58879 APPLICATION USAGE

58880 None.

58881 RATIONALE

58882 None.

58883 FUTURE DIRECTIONS

58884 None.

58885 SEE ALSO

58886 [*semctl\(\)*, *semget\(\)*, *semop\(\)*, *sem_init\(\)*, *sem_open\(\)*, *sem_unlink\(\)*](#)

58887 XBD [**<semaphore.h>**](#)

58888 CHANGE HISTORY

58889 First released in Issue 5. Included for alignment with the POSIX Realtime Extension.

58890 Issue 6

58891 The *sem_close()* function is marked as part of the Semaphores option.

58892 The [ENOSYS] error condition has been removed as stubs need not be provided if an
 58893 implementation does not support the Semaphores option.

58894 IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/113 is applied, updating the ERRORS
 58895 section so that the [EINVAL] error becomes optional.

58896 **Issue 7**

58897 The *sem_close()* function is moved from the Semaphores option to the Base.

58898 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0523 [37] is applied.

+

58899 NAME

58900 sem_destroy — destroy an unnamed semaphore

58901 SYNOPSIS

```
58902 #include <semaphore.h>
58903 int sem_destroy(sem_t *sem);
```

58904 DESCRIPTION

58905 The *sem_destroy()* function shall destroy the unnamed semaphore indicated by *sem*. Only a
 58906 semaphore that was created using *sem_init()* may be destroyed using *sem_destroy()*; the effect of
 58907 calling *sem_destroy()* with a named semaphore is undefined. The effect of subsequent use of the
 58908 semaphore *sem* is undefined until *sem* is reinitialized by another call to *sem_init()*.

58909 It is safe to destroy an initialized semaphore upon which no threads are currently blocked. The
 58910 effect of destroying a semaphore upon which other threads are currently blocked is undefined.

58911 RETURN VALUE

58912 Upon successful completion, a value of zero shall be returned. Otherwise, a value of -1 shall be
 58913 returned and *errno* set to indicate the error.

58914 ERRORS

58915 The *sem_destroy()* function may fail if:

58916	[EINVAL]	The <i>sem</i> argument is not a valid semaphore.
58917	[EBUSY]	There are currently processes blocked on the semaphore.

58918 EXAMPLES

58919 None.

58920 APPLICATION USAGE

58921 None.

58922 RATIONALE

58923 None.

58924 FUTURE DIRECTIONS

58925 None.

58926 SEE ALSO

58927 [semctl\(\)](#), [semget\(\)](#), [semop\(\)](#), [sem_init\(\)](#), [sem_open\(\)](#)

58928 XBD [<semaphore.h>](#)

58929 CHANGE HISTORY

58930 First released in Issue 5. Included for alignment with the POSIX Realtime Extension.

58931 Issue 6

58932 The *sem_destroy()* function is marked as part of the Semaphores option.

58933 The [ENOSYS] error condition has been removed as stubs need not be provided if an
 58934 implementation does not support the Semaphores option.

58935 IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/114 is applied, updating the ERRORS
 58936 section so that the [EINVAL] error becomes optional.

58937 Issue 7

58938 The *sem_destroy()* function is moved from the Semaphores option to the Base.

58939 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0524 [37] is applied.

58940 **NAME**

58941 sem_getvalue — get the value of a semaphore

58942 **SYNOPSIS**

58943 #include <semaphore.h>

58944 int sem_getvalue(sem_t *restrict sem, int *restrict sval);

58945 **DESCRIPTION**

58946 The *sem_getvalue()* function shall update the location referenced by the *sval* argument to have
58947 the value of the semaphore referenced by *sem* without affecting the state of the semaphore. The
58948 updated value represents an actual semaphore value that occurred at some unspecified time
58949 during the call, but it need not be the actual value of the semaphore when it is returned to the
58950 calling process.

58951 If *sem* is locked, then the object to which *sval* points shall either be set to zero or to a negative
58952 number whose absolute value represents the number of processes waiting for the semaphore at
58953 some unspecified time during the call.

58954 **RETURN VALUE**

58955 Upon successful completion, the *sem_getvalue()* function shall return a value of zero. Otherwise,
58956 it shall return a value of -1 and set *errno* to indicate the error.

58957 **ERRORS**58958 The *sem_getvalue()* function may fail if:58959 [EINVAL] The *sem* argument does not refer to a valid semaphore.58960 **EXAMPLES**

58961 None.

58962 **APPLICATION USAGE**

58963 None.

58964 **RATIONALE**

58965 None.

58966 **FUTURE DIRECTIONS**

58967 None.

58968 **SEE ALSO**58969 *semctl()*, *semget()*, *semop()*, *sem_post()*, *sem_timedwait()*, *sem_trywait()*

58970 XBD <semaphore.h>

58971 **CHANGE HISTORY**

58972 First released in Issue 5. Included for alignment with the POSIX Realtime Extension.

58973 **Issue 6**58974 The *sem_getvalue()* function is marked as part of the Semaphores option.

58975 The [ENOSYS] error condition has been removed as stubs need not be provided if an
58976 implementation does not support the Semaphores option.

58977 The *sem_timedwait()* function is added to the SEE ALSO section for alignment with IEEE Std
58978 1003.1d-1999.

58979 The **restrict** keyword is added to the *sem_getvalue()* prototype for alignment with the
58980 ISO/IEC 9899:1999 standard.

58981 IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/54 is applied.

- 58982 IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/115 is applied, updating the ERRORS
58983 section so that the [EINVAL] error becomes optional.
- 58984 **Issue 7**
58985 The *sem_getvalue()* function is moved from the Semaphores option to the Base.
58986 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0525 [37] is applied. +

58987 **NAME**

58988 sem_init — initialize an unnamed semaphore

58989 **SYNOPSIS**

58990 #include <semaphore.h>

58991 int sem_init(sem_t *sem, int pshared, unsigned value);

58992 **DESCRIPTION**

58993 The *sem_init()* function shall initialize the unnamed semaphore referred to by *sem*. The value of
 58994 the initialized semaphore shall be *value*. Following a successful call to *sem_init()*, the semaphore
 58995 may be used in subsequent calls to *sem_wait()*, *sem_timedwait()*, *sem_trywait()*, *sem_post()*, and
 58996 *sem_destroy()*. This semaphore shall remain usable until the semaphore is destroyed.

58997 If the *pshared* argument has a non-zero value, then the semaphore is shared between processes;
 58998 in this case, any process that can access the semaphore *sem* can use *sem* for performing
 58999 *sem_wait()*, *sem_timedwait()*, *sem_trywait()*, *sem_post()*, and *sem_destroy()* operations.

59000 Only *sem* itself may be used for performing synchronization. The result of referring to copies of
 59001 *sem* in calls to *sem_wait()*, *sem_timedwait()*, *sem_trywait()*, *sem_post()*, and *sem_destroy()* is
 59002 undefined.

59003 If the *pshared* argument is zero, then the semaphore is shared between threads of the process; any
 59004 thread in this process can use *sem* for performing *sem_wait()*, *sem_timedwait()*, *sem_trywait()*,
 59005 *sem_post()*, and *sem_destroy()* operations. The use of the semaphore by threads other than those
 59006 created in the same process is undefined.

59007 Attempting to initialize an already initialized semaphore results in undefined behavior.

59008 **RETURN VALUE**

59009 Upon successful completion, the *sem_init()* function shall initialize the semaphore in *sem* and
 59010 return 0. Otherwise, it shall return -1 and set *errno* to indicate the error.

59011 **ERRORS**

59012 The *sem_init()* function shall fail if:

59013	[EINVAL]	The <i>value</i> argument exceeds {SEM_VALUE_MAX}.
59014	[ENOSPC]	A resource required to initialize the semaphore has been exhausted, or the limit on semaphores ({SEM_NSEMS_MAX}) has been reached.
59015		
59016	[EPERM]	The process lacks appropriate privileges to initialize the semaphore.

59017 **EXAMPLES**

59018 None.

59019 **APPLICATION USAGE**

59020 None.

59021 **RATIONALE**

59022 None.

59023 **FUTURE DIRECTIONS**

59024 None.

59025 **SEE ALSO**

59026 [*sem_destroy\(\)*](#), [*sem_post\(\)*](#), [*sem_timedwait\(\)*](#), [*sem_trywait\(\)*](#)

59027 XBD [**<semaphore.h>**](#)

59028 **CHANGE HISTORY**

59029 First released in Issue 5. Included for alignment with the POSIX Realtime Extension.

59030 **Issue 6**

59031 The *sem_init()* function is marked as part of the Semaphores option.

59032 The [ENOSYS] error condition has been removed as stubs need not be provided if an
59033 implementation does not support the Semaphores option.

59034 The *sem_timedwait()* function is added to the SEE ALSO section for alignment with IEEE Std
59035 1003.1d-1999.

59036 IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/116 is applied, updating the
59037 DESCRIPTION to add the *sem_timedwait()* function for alignment with IEEE Std 1003.1d-1999.

59038 **Issue 7**

59039 SD5-XSH-ERN-176 is applied.

59040 The *sem_init()* function is moved from the Semaphores option to the Base.

59041 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0526 [37] is applied.

+

NAME

sem_open — initialize and open a named semaphore

SYNOPSIS

```
#include <semaphore.h>
```

```
sem_t *sem_open(const char *name, int oflag, ...);
```

DESCRIPTION

The *sem_open()* function shall establish a connection between a named semaphore and a process. Following a call to *sem_open()* with semaphore name *name*, the process may reference the semaphore associated with *name* using the address returned from the call. This semaphore may be used in subsequent calls to *sem_wait()*, *sem_timedwait()*, *sem_trywait()*, *sem_post()*, and *sem_close()*. The semaphore remains usable by this process until the semaphore is closed by a successful call to *sem_close()*, *_exit()*, or one of the *exec* functions.

The *oflag* argument controls whether the semaphore is created or merely accessed by the call to *sem_open()*. The following flag bits may be set in *oflag*:

O_CREAT This flag is used to create a semaphore if it does not already exist. If **O_CREAT** is set and the semaphore already exists, then **O_CREAT** has no effect, except as noted under **O_EXCL**. Otherwise, *sem_open()* creates a named semaphore. The **O_CREAT** flag requires a third and a fourth argument: *mode*, which is of type **mode_t**, and *value*, which is of type **unsigned**. The semaphore is created with an initial value of *value*. Valid initial values for semaphores are less than or equal to {SEM_VALUE_MAX}.

The user ID of the semaphore shall be set to the effective user ID of the process. The group ID of the semaphore shall be set to the effective group ID of the process; however, if the *name* argument is visible in the file system, the group ID may be set to the group ID of the containing directory. The permission bits of the semaphore are set to the value of the *mode* argument except those set in the file mode creation mask of the process. When bits in *mode* other than the file permission bits are specified, the effect is unspecified.

After the semaphore named *name* has been created by *sem_open()* with the **O_CREAT** flag, other processes can connect to the semaphore by calling *sem_open()* with the same value of *name*.

O_EXCL If **O_EXCL** and **O_CREAT** are set, *sem_open()* fails if the semaphore *name* exists. The check for the existence of the semaphore and the creation of the semaphore if it does not exist are atomic with respect to other processes executing *sem_open()* with **O_EXCL** and **O_CREAT** set. If **O_EXCL** is set and **O_CREAT** is not set, the effect is undefined.

If flags other than **O_CREAT** and **O_EXCL** are specified in the *oflag* parameter, the effect is unspecified.

The *name* argument points to a string naming a semaphore object. It is unspecified whether the name appears in the file system and is visible to functions that take pathnames as arguments. The *name* argument conforms to the construction rules for a pathname, except that the interpretation of <slash> characters other than the leading <slash> character in *name* is implementation-defined, and that the length limits for the *name* argument are implementation-defined and need not be the same as the pathname limits {PATH_MAX} and {NAME_MAX}. If *name* begins with the <slash> character, then processes calling *sem_open()* with the same value of *name* shall refer to the same semaphore object, as long as that name has not been removed. If *name* does not begin with the <slash> character, the effect is implementation-defined.

59089 If a process makes multiple successful calls to *sem_open()* with the same value for *name*, the same
 59090 semaphore address shall be returned for each such successful call, provided that there have been
 59091 no calls to *sem_unlink()* for this semaphore, and at least one previous successful *sem_open()* call
 59092 for this semaphore has not been matched with a *sem_close()* call.

59093 References to copies of the semaphore produce undefined results.

59094 RETURN VALUE

59095 Upon successful completion, the *sem_open()* function shall return the address of the semaphore.
 59096 Otherwise, it shall return a value of SEM_FAILED and set *errno* to indicate the error. The symbol
 59097 SEM_FAILED is defined in the **<semaphore.h>** header. No successful return from *sem_open()*
 59098 shall return the value SEM_FAILED.

59099 ERRORS

59100 If any of the following conditions occur, the *sem_open()* function shall return SEM_FAILED and
 59101 set *errno* to the corresponding value:

59102 [EACCES] The named semaphore exists and the permissions specified by *oflag* are
 59103 denied, or the named semaphore does not exist and permission to create the
 59104 named semaphore is denied.

59105 [EEXIST] O_CREAT and O_EXCL are set and the named semaphore already exists.

59106 [EINTR] The *sem_open()* operation was interrupted by a signal.

59107 [EINVAL] The *sem_open()* operation is not supported for the given name, or O_CREAT
 59108 was specified in *oflag* and *value* was greater than {SEM_VALUE_MAX}.

59109 [EMFILE] Too many semaphore descriptors or file descriptors are currently in use by this
 59110 process.

59111 [ENFILE] Too many semaphores are currently open in the system.

59112 [ENOENT] O_CREAT is not set and the named semaphore does not exist.

59113 [ENOMEM] There is insufficient memory for the creation of the new named semaphore.

59114 [ENOSPC] There is insufficient space on a storage device for the creation of the new
 59115 named semaphore.

59116 If any of the following conditions occur, the *sem_open()* function may return SEM_FAILED and
 59117 set *errno* to the corresponding value:

59118 [ENAMETOOLONG]

59119 The length of the *name* argument exceeds {_POSIX_PATH_MAX} on systems
 59120 XSI that do not support the XSI option or exceeds {_XOPEN_PATH_MAX} on XSI
 59121 systems, or has a pathname component that is longer than
 59122 XSI {_POSIX_NAME_MAX} on systems that do not support the XSI option or
 59123 longer than {_XOPEN_NAME_MAX} on XSI systems.

EXAMPLES

None.

APPLICATION USAGE

None.

RATIONALE

Early drafts required an error return value of -1 with the type **sem_t** * for the *sem_open()* function, which is not guaranteed to be portable across implementations. The revised text provides the symbolic error code SEM_FAILED to eliminate the type conflict.

FUTURE DIRECTIONS

A future version might require the *sem_open()* and *sem_unlink()* functions to have semantics similar to normal file system operations.

SEE ALSO

semctl(), *semget()*, *semop()*, *sem_close()*, *sem_post()*, *sem_timedwait()*, *sem_trywait()*, *sem_unlink()*

XBD <**semaphore.h**>

CHANGE HISTORY

First released in Issue 5. Included for alignment with the POSIX Realtime Extension.

Issue 6

The *sem_open()* function is marked as part of the Semaphores option.

The [ENOSYS] error condition has been removed as stubs need not be provided if an implementation does not support the Semaphores option.

The *sem_timedwait()* function is added to the SEE ALSO section for alignment with IEEE Std 1003.1d-1999.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/117 is applied, updating the DESCRIPTION to add the *sem_timedwait()* function for alignment with IEEE Std 1003.1d-1999.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/118 is applied, updating the DESCRIPTION to describe the conditions to return the same semaphore address on a call to *sem_open()*. The words “and at least one previous successful *sem_open()* call for this semaphore has not been matched with a *sem_close()* call” are added.

Issue 7

Austin Group Interpretation 1003.1-2001 #066 is applied, updating the [ENOSPC] error case and adding the [ENOMEM] error case.

Austin Group Interpretation 1003.1-2001 #077 is applied, clarifying the *name* argument and adding [ENAMETOOLONG] as a “may fail” error.

Austin Group Interpretation 1003.1-2001 #141 is applied, adding FUTURE DIRECTIONS.

SD5-XSH-ERN-170 is applied, updating the DESCRIPTION to clarify the wording for setting the user ID and group ID of the semaphore.

The *sem_open()* function is moved from the Semaphores option to the Base.

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0527 [37] is applied.

59162 **NAME**

59163 sem_post — unlock a semaphore

59164 **SYNOPSIS**

59165 #include <semaphore.h>

59166 int sem_post(sem_t *sem);

59167 **DESCRIPTION**

59168 The *sem_post()* function shall unlock the semaphore referenced by *sem* by performing a
 59169 semaphore unlock operation on that semaphore.

59170 If the semaphore value resulting from this operation is positive, then no threads were blocked
 59171 waiting for the semaphore to become unlocked; the semaphore value is simply incremented.

59172 If the value of the semaphore resulting from this operation is zero, then one of the threads
 59173 blocked waiting for the semaphore shall be allowed to return successfully from its call to
 59174 PS *sem_wait()*. If the Process Scheduling option is supported, the thread to be unblocked shall be
 59175 chosen in a manner appropriate to the scheduling policies and parameters in effect for the
 59176 blocked threads. In the case of the schedulers SCHED_FIFO and SCHED_RR, the highest
 59177 priority waiting thread shall be unblocked, and if there is more than one highest priority thread
 59178 blocked waiting for the semaphore, then the highest priority thread that has been waiting the
 59179 longest shall be unblocked. If the Process Scheduling option is not defined, the choice of a thread
 59180 to unblock is unspecified.

59181 SS If the Process Sporadic Server option is supported, and the scheduling policy is
 59182 SCHED_SPORADIC, the semantics are as per SCHED_FIFO above.

59183 The *sem_post()* function shall be async-signal-safe and may be invoked from a signal-catching
 59184 function.

59185 **RETURN VALUE**

59186 If successful, the *sem_post()* function shall return zero; otherwise, the function shall return -1
 59187 and set *errno* to indicate the error.

59188 **ERRORS**

59189 The *sem_post()* function may fail if:

59190 [EINVAL] The *sem* argument does not refer to a valid semaphore.

59191 **EXAMPLES**

59192 See *sem_timedwait()*.

59193 **APPLICATION USAGE**

59194 None.

59195 **RATIONALE**

59196 None.

59197 **FUTURE DIRECTIONS**

59198 None.

59199 **SEE ALSO**

59200 *semctl()*, *semget()*, *semop()*, *sem_timedwait()*, *sem_trywait()*

59201 XBD Section 4.11 (on page 110), <semaphore.h>

59202 **CHANGE HISTORY**

59203 First released in Issue 5. Included for alignment with the POSIX Realtime Extension.

59204 **Issue 6**

59205 The *sem_post()* function is marked as part of the Semaphores option.

59206 The [ENOSYS] error condition has been removed as stubs need not be provided if an
59207 implementation does not support the Semaphores option.

59208 The *sem_timedwait()* function is added to the SEE ALSO section for alignment with IEEE Std
59209 1003.1d-1999.

59210 SCHED_SPORADIC is added to the list of scheduling policies for which the thread that is to be
59211 unblocked is specified for alignment with IEEE Std 1003.1d-1999.

59212 IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/119 is applied, updating the ERRORS
59213 section so that the [EINVAL] error becomes optional.

59214 **Issue 7**

59215 Austin Group Interpretation 1003.1-2001 #156 is applied.

59216 The *sem_post()* function is moved from the Semaphores option to the Base.

59217 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0528 [37] is applied.

+

NAME

sem_timedwait — lock a semaphore

SYNOPSIS

```
#include <semaphore.h>
#include <time.h>

int sem_timedwait(sem_t *restrict sem,
                  const struct timespec *restrict abstime);
```

DESCRIPTION

The *sem_timedwait()* function shall lock the semaphore referenced by *sem* as in the *sem_wait()* function. However, if the semaphore cannot be locked without waiting for another process or thread to unlock the semaphore by performing a *sem_post()* function, this wait shall be terminated when the specified timeout expires.

The timeout shall expire when the absolute time specified by *abstime* passes, as measured by the clock on which timeouts are based (that is, when the value of that clock equals or exceeds *abstime*), or if the absolute time specified by *abstime* has already been passed at the time of the call.

The timeout shall be based on the CLOCK_REALTIME clock. The resolution of the timeout shall be the resolution of the clock on which it is based. The **timespec** data type is defined as a structure in the **<time.h>** header.

Under no circumstance shall the function fail with a timeout if the semaphore can be locked immediately. The validity of the *abstime* need not be checked if the semaphore can be locked immediately.

RETURN VALUE

The *sem_timedwait()* function shall return zero if the calling process successfully performed the semaphore lock operation on the semaphore designated by *sem*. If the call was unsuccessful, the state of the semaphore shall be unchanged, and the function shall return a value of -1 and set *errno* to indicate the error.

ERRORS

The *sem_timedwait()* function shall fail if:

[EINVAL] The process or thread would have blocked, and the *abstime* parameter specified a nanoseconds field value less than zero or greater than or equal to 1 000 million.

[ETIMEDOUT] The semaphore could not be locked before the specified timeout expired.

The *sem_timedwait()* function may fail if:

[EDEADLK] A deadlock condition was detected.

[EINTR] A signal interrupted this function.

[EINVAL] The *sem* argument does not refer to a valid semaphore.

EXAMPLES

The program shown below operates on an unnamed semaphore. The program expects two command-line arguments. The first argument specifies a seconds value that is used to set an alarm timer to generate a SIGALRM signal. This handler performs a *sem_post*(3) to increment the semaphore that is being waited on in *main*() using *sem_timedwait*(). The second command-line argument specifies the length of the timeout, in seconds, for *sem_timedwait*().

```

#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include <semaphore.h>
#include <time.h>
#include <assert.h>
#include <errno.h>
#include <signal.h>

sem_t sem;

static void
handler(int sig)
{
    int sav_errno = errno;
    static const char info_msg[] = "sem_post() from handler\n";
    write(STDOUT_FILENO, info_msg, sizeof info_msg - 1);
    if (sem_post(&sem) == -1) {
        static const char err_msg[] = "sem_post() failed\n";
        write(STDERR_FILENO, err_msg, sizeof err_msg - 1);
        _exit(EXIT_FAILURE);
    }
    errno = sav_errno;
}

int
main(int argc, char *argv[])
{
    struct sigaction sa;
    struct timespec ts;
    int s;

    if (argc != 3) {
        fprintf(stderr, "Usage: %s <alarm-secs> <wait-secs>\n",
            argv[0]);
        exit(EXIT_FAILURE);
    }

    if (sem_init(&sem, 0, 0) == -1) {
        perror("sem_init");
        exit(EXIT_FAILURE);
    }

    /* Establish SIGALRM handler; set alarm timer using argv[1] */
    sa.sa_handler = handler;
    sigemptyset(&sa.sa_mask);
    sa.sa_flags = 0;
    if (sigaction(SIGALRM, &sa, NULL) == -1) {

```

```

59303         perror("sigaction");
59304         exit(EXIT_FAILURE);
59305     }
59306     alarm(atoi(argv[1]));
59307     /* Calculate relative interval as current time plus
59308        number of seconds given argv[2] */
59309     if (clock_gettime(CLOCK_REALTIME, &ts) == -1) {
59310         perror("clock_gettime");
59311         exit(EXIT_FAILURE);
59312     }
59313     ts.tv_sec += atoi(argv[2]);
59314     printf("main() about to call sem_timedwait()\n");
59315     while ((s = sem_timedwait(&sem, &ts)) == -1 && errno == EINTR)
59316         continue;          /* Restart if interrupted by handler */
59317     /* Check what happened */
59318     if (s == -1) {
59319         if (errno == ETIMEDOUT)
59320             printf("sem_timedwait() timed out\n");
59321         else
59322             perror("sem_timedwait");
59323     } else
59324         printf("sem_timedwait() succeeded\n");
59325     exit((s == 0) ? EXIT_SUCCESS : EXIT_FAILURE);
59326 }

```

APPLICATION USAGE

Applications using these functions may be subject to priority inversion, as discussed in XBD [Section 3.287](#) (on page 79).

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

[*sem_post\(\)*](#), [*sem_trywait\(\)*](#), [*semctl\(\)*](#), [*semget\(\)*](#), [*semop\(\)*](#), [*time\(\)*](#)

XBD [Section 3.287](#) (on page 79), [**<semaphore.h>**](#), [**<time.h>**](#)

CHANGE HISTORY

First released in Issue 6. Derived from IEEE Std 1003.1d-1999.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/120 is applied, updating the ERRORS section so that the [EINVAL] error becomes optional.

Issue 7

The *sem_timedwait()* function is moved from the Semaphores option to the Base.

Functionality relating to the Timers option is moved to the Base.

An example is added.

59345

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0529 [138] is applied.

+

NAME

sem_trywait, sem_wait — lock a semaphore

SYNOPSIS

```
#include <semaphore.h>

int sem_trywait(sem_t *sem);
int sem_wait(sem_t *sem);
```

DESCRIPTION

The *sem_trywait()* function shall lock the semaphore referenced by *sem* only if the semaphore is currently not locked; that is, if the semaphore value is currently positive. Otherwise, it shall not lock the semaphore.

The *sem_wait()* function shall lock the semaphore referenced by *sem* by performing a semaphore lock operation on that semaphore. If the semaphore value is currently zero, then the calling thread shall not return from the call to *sem_wait()* until it either locks the semaphore or the call is interrupted by a signal.

Upon successful return, the state of the semaphore shall be locked and shall remain locked until the *sem_post()* function is executed and returns successfully.

The *sem_wait()* function is interruptible by the delivery of a signal.

RETURN VALUE

The *sem_trywait()* and *sem_wait()* functions shall return zero if the calling process successfully performed the semaphore lock operation on the semaphore designated by *sem*. If the call was unsuccessful, the state of the semaphore shall be unchanged, and the function shall return a value of -1 and set *errno* to indicate the error.

ERRORS

The *sem_trywait()* function shall fail if:

[EAGAIN] The semaphore was already locked, so it cannot be immediately locked by the *sem_trywait()* operation.

The *sem_trywait()* and *sem_wait()* functions may fail if:

[EDEADLK] A deadlock condition was detected.

[EINTR] A signal interrupted this function.

[EINVAL] The *sem* argument does not refer to a valid semaphore.

EXAMPLES

None.

APPLICATION USAGE

Applications using these functions may be subject to priority inversion, as discussed in XBD [Section 3.287](#) (on page 79).

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

[*semctl\(\)*](#), [*semget\(\)*](#), [*semop\(\)*](#), [*sem_post\(\)*](#), [*sem_timedwait\(\)*](#)

XBD [Section 3.287](#) (on page 79), [Section 4.11](#) (on page 110), [**<semaphore.h>**](#)

CHANGE HISTORY

First released in Issue 5. Included for alignment with the POSIX Realtime Extension.

Issue 6

The *sem_trywait()* and *sem_wait()* functions are marked as part of the Semaphores option.

The [ENOSYS] error condition has been removed as stubs need not be provided if an implementation does not support the Semaphores option.

The *sem_timedwait()* function is added to the SEE ALSO section for alignment with IEEE Std 1003.1d-1999.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/121 is applied, updating the ERRORS section so that the [EINVAL] error becomes optional.

Issue 7

SD5-XSH-ERN-54 is applied, removing the *sem_wait()* function from the “shall fail” error cases.

The *sem_trywait()* and *sem_wait()* functions are moved from the Semaphores option to the Base.

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0530 [37] is applied.

+

59402 **NAME**

59403 sem_unlink — remove a named semaphore

59404 **SYNOPSIS**

59405 #include <semaphore.h>

59406 int sem_unlink(const char *name);

59407 **DESCRIPTION**

59408 The *sem_unlink()* function shall remove the semaphore named by the string *name*. If the
 59409 semaphore named by *name* is currently referenced by other processes, then *sem_unlink()* shall
 59410 have no effect on the state of the semaphore. If one or more processes have the semaphore open
 59411 when *sem_unlink()* is called, destruction of the semaphore is postponed until all references to the
 59412 semaphore have been destroyed by calls to *sem_close()*, *_exit()*, or *exec*. Calls to *sem_open()* to
 59413 recreate or reconnect to the semaphore refer to a new semaphore after *sem_unlink()* is called. The
 59414 *sem_unlink()* call shall not block until all references have been destroyed; it shall return
 59415 immediately.

59416 **RETURN VALUE**

59417 Upon successful completion, the *sem_unlink()* function shall return a value of 0. Otherwise, the
 59418 semaphore shall not be changed and the function shall return a value of -1 and set *errno* to
 59419 indicate the error.

59420 **ERRORS**59421 The *sem_unlink()* function shall fail if:

59422 [EACCES] Permission is denied to unlink the named semaphore.

59423 [ENOENT] The named semaphore does not exist.

59424 The *sem_unlink()* function may fail if:

59425 [ENAMETOOLONG]

59426 The length of the *name* argument exceeds {_POSIX_PATH_MAX} on systems
 59427 XSI that do not support the XSI option or exceeds {_XOPEN_PATH_MAX} on XSI
 59428 systems, or has a pathname component that is longer than
 59429 XSI {_POSIX_NAME_MAX} on systems that do not support the XSI option or
 59430 longer than {_XOPEN_NAME_MAX} on XSI systems. A call to *sem_unlink()*
 59431 with a *name* argument that contains the same semaphore name as was
 59432 previously used in a successful *sem_open()* call shall not give an
 59433 [ENAMETOOLONG] error.

59434 **EXAMPLES**

59435 None.

59436 **APPLICATION USAGE**

59437 None.

59438 **RATIONALE**

59439 None.

59440 **FUTURE DIRECTIONS**

59441 A future version might require the *sem_open()* and *sem_unlink()* functions to have semantics
 59442 similar to normal file system operations.

59443 **SEE ALSO**59444 *semctl()*, *semget()*, *semop()*, *sem_close()*, *sem_open()*

59445 XBD <semaphore.h>

59446 **CHANGE HISTORY**

59447 First released in Issue 5. Included for alignment with the POSIX Realtime Extension.

59448 **Issue 6**

59449 The *sem_unlink()* function is marked as part of the Semaphores option.

59450 The [ENOSYS] error condition has been removed as stubs need not be provided if an
59451 implementation does not support the Semaphores option.

59452 **Issue 7**

59453 Austin Group Interpretation 1003.1-2001 #077 is applied, changing [ENAMETOOLONG] from a
59454 “shall fail” to a “may fail” error.

59455 Austin Group Interpretation 1003.1-2001 #141 is applied, adding FUTURE DIRECTIONS.

59456 The *sem_unlink()* function is moved from the Semaphores option to the Base.

59457 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0531 [37] is applied.

+

59458 **NAME**

59459 sem_wait — lock a semaphore

59460 **SYNOPSIS**

59461 #include <semaphore.h>

59462 int sem_wait(sem_t *sem);

59463 **DESCRIPTION**59464 Refer to *sem_trywait()*.

59465 **NAME**

59466 semctl — XSI semaphore control operations

59467 **SYNOPSIS**

```
59468 XSI      #include <sys/sem.h>
59469      int semctl(int semid, int semnum, int cmd, ...);
```

59470 **DESCRIPTION**

59471 The *semctl()* function operates on XSI semaphores (see XBD [Section 4.16](#), on page 113). It is
 59472 unspecified whether this function interoperates with the realtime interprocess communication
 59473 facilities defined in [Section 2.8](#) (on page 502).

59474 The *semctl()* function provides a variety of semaphore control operations as specified by *cmd*.
 59475 The fourth argument is optional and depends upon the operation requested. If required, it is of
 59476 type **union semun**, which the application shall explicitly declare:

```
59477 union semun {
59478     int val;
59479     struct semid_ds *buf;
59480     unsigned short *array;
59481 } arg;
```

59482 The following semaphore control operations as specified by *cmd* are executed with respect to the
 59483 semaphore specified by *semid* and *semnum*. The level of permission required for each operation
 59484 is shown with each command; see [Section 2.7](#) (on page 501). The symbolic names for the values
 59485 of *cmd* are defined in the **<sys/sem.h>** header:

59486	GETVAL	Return the value of <i>semval</i> ; see <sys/sem.h> . Requires read permission.
59487	SETVAL	Set the value of <i>semval</i> to <i>arg.val</i> , where <i>arg</i> is the value of the fourth argument 59488 to <i>semctl()</i> . When this command is successfully executed, the <i>semadj</i> value 59489 corresponding to the specified semaphore in all processes is cleared. Also, the 59490 <i>sem_ctime</i> timestamp shall be set to the current time, as described in Section 59491 2.7.1 (on page 501). Requires alter permission; see Section 2.7 (on page 501).

59492	GETPID	Return the value of <i>sempid</i> . Requires read permission.

59493	GETNCNT	Return the value of <i>semncnt</i> . Requires read permission.

59494	GETZCNT	Return the value of <i>semzcnt</i> . Requires read permission.

59495 The following values of *cmd* operate on each *semval* in the set of semaphores:

59496	GETALL	Return the value of <i>semval</i> for each semaphore in the semaphore set and place 59497 into the array pointed to by <i>arg.array</i> , where <i>arg</i> is the fourth argument to 59498 <i>semctl()</i> . Requires read permission.

59499	SETALL	Set the value of <i>semval</i> for each semaphore in the semaphore set according to 59500 the array pointed to by <i>arg.array</i> , where <i>arg</i> is the fourth argument to <i>semctl()</i> . 59501 When this command is successfully executed, the <i>semadj</i> values corresponding 59502 to each specified semaphore in all processes are cleared. Also, the <i>sem_ctime</i> 59503 timestamp shall be set to the current time, as described in Section 2.7.1 (on 59504 page 501). Requires alter permission.

59505 The following values of *cmd* are also available:

59506 IPC_STAT Place the current value of each member of the **semid_ds** data structure
59507 associated with *semid* into the structure pointed to by *arg.buf*, where *arg* is the
59508 fourth argument to *semctl()*. The contents of this structure are defined in
59509 **<sys/sem.h>**. Requires read permission.

59510 IPC_SET Set the value of the following members of the **semid_ds** data structure
59511 associated with *semid* to the corresponding value found in the structure
59512 pointed to by *arg.buf*, where *arg* is the fourth argument to *semctl()*:

59513 sem_perm.uid
59514 sem_perm.gid
59515 sem_perm.mode

59516 The mode bits specified in [Section 2.7.1](#) (on page 501) are copied into the
59517 corresponding bits of the *sem_perm.mode* associated with *semid*. The stored
59518 values of any other bits are unspecified. The *sem_ctime* timestamp shall be set
59519 to the current time, as described in [Section 2.7.1](#) (on page 501).

59520 This command can only be executed by a process that has an effective user ID
59521 equal to either that of a process with appropriate privileges or to the value of
59522 *sem_perm.cuid* or *sem_perm.uid* in the **semid_ds** data structure associated with
59523 *semid*.

59524 IPC_RMID Remove the semaphore identifier specified by *semid* from the system and
59525 destroy the set of semaphores and **semid_ds** data structure associated with it.
59526 This command can only be executed by a process that has an effective user ID
59527 equal to either that of a process with appropriate privileges or to the value of
59528 *sem_perm.cuid* or *sem_perm.uid* in the **semid_ds** data structure associated with
59529 *semid*.

59530 RETURN VALUE

59531 If successful, the value returned by *semctl()* depends on *cmd* as follows:

59532 GETVAL The value of *semval*.
59533 GETPID The value of *sempid*.
59534 GETNCNT The value of *semmcnt*.
59535 GETZCNT The value of *semzcnt*.
59536 All others 0.

59537 Otherwise, *semctl()* shall return -1 and set *errno* to indicate the error.

59538 ERRORS

59539 The *semctl()* function shall fail if:

59540 [EACCES] Operation permission is denied to the calling process; see [Section 2.7](#) (on page
59541 501).
59542 [EINVAL] The value of *semid* is not a valid semaphore identifier, or the value of *semnum*
59543 is less than 0 or greater than or equal to *sem_nsems*, or the value of *cmd* is not a
59544 valid command.
59545 [EPERM] The argument *cmd* is equal to IPC_RMID or IPC_SET and the effective user ID
59546 of the calling process is not equal to that of a process with appropriate
59547 privileges and it is not equal to the value of *sem_perm.cuid* or *sem_perm.uid* in

59548 the data structure associated with *semid*.
 59549 [ERANGE] The argument *cmd* is equal to SETVAL or SETALL and the value to which
 59550 *semval* is to be set is greater than the system-imposed maximum.

59551 EXAMPLES

59552 Refer to *semop()*.

59553 APPLICATION USAGE

59554 The fourth parameter in the SYNOPSIS section is now specified as ". . ." in order to avoid a
 59555 clash with the ISO C standard when referring to the union *semun* (as defined in Issue 3) and for
 59556 backwards-compatibility.

59557 The POSIX Realtime Extension defines alternative interfaces for interprocess communication.
 59558 Application developers who need to use IPC should design their applications so that modules
 59559 using the IPC routines described in [Section 2.7](#) (on page 501) can be easily modified to use the
 59560 alternative interfaces.

59561 RATIONALE

59562 None.

59563 FUTURE DIRECTIONS

59564 None.

59565 SEE ALSO

59566 [Section 2.7](#) (on page 501), [Section 2.8](#) (on page 502), *semget()*, *semop()*, *sem_close()*, *sem_destroy()*,
 59567 *sem_getvalue()*, *sem_init()*, *sem_open()*, *sem_post()*, *sem_trywait()*, *sem_unlink()*

59568 XBD [Section 4.16](#) (on page 113), [<sys/sem.h>](#)

59569 CHANGE HISTORY

59570 First released in Issue 2. Derived from Issue 2 of the SVID.

59571 Issue 5

59572 The note about use of POSIX Realtime Extension IPC routines has been moved from FUTURE
 59573 DIRECTIONS to the APPLICATION USAGE section.

59574 Issue 7

59575 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0532 [345], XSH/TC1-2008/0533 [345], +
 59576 XSH/TC1-2008/0534 [345], and XSH/TC1-2008/0535 [335] are applied.

NAME

semget — get set of XSI semaphores

SYNOPSIS

```
XSI    #include <sys/sem.h>
      int semget(key_t key, int nsems, int semflg);
```

DESCRIPTION

The *semget()* function operates on XSI semaphores (see XBD [Section 4.16](#), on page 113). It is unspecified whether this function interoperates with the realtime interprocess communication facilities defined in [Section 2.8](#) (on page 502).

The *semget()* function shall return the semaphore identifier associated with *key*.

A semaphore identifier with its associated **semid_ds** data structure and its associated set of *nsems* semaphores (see **<sys/sem.h>**) is created for *key* if one of the following is true:

- The argument *key* is equal to `IPC_PRIVATE`.
- The argument *key* does not already have a semaphore identifier associated with it and (*semflg* & `IPC_CREAT`) is non-zero.

Upon creation, the **semid_ds** data structure associated with the new semaphore identifier is initialized as follows:

- In the operation permissions structure *sem_perm.cuid*, *sem_perm.uid*, *sem_perm.cgid*, and *sem_perm.gid* shall be set to the effective user ID and effective group ID, respectively, of the calling process.
- The low-order 9 bits of *sem_perm.mode* shall be set to the low-order 9 bits of *semflg*.
- The variable *sem_nsems* shall be set to the value of *nsems*.
- The variable *sem_otime* shall be set to 0 and *sem_ctime* shall be set to the current time, as described in [Section 2.7.1](#) (on page 501).
- The data structure associated with each semaphore in the set need not be initialized. The *semctl()* function with the command `SETVAL` or `SETALL` can be used to initialize each semaphore.

RETURN VALUE

Upon successful completion, *semget()* shall return a non-negative integer, namely a semaphore identifier; otherwise, it shall return `-1` and set *errno* to indicate the error.

ERRORS

The *semget()* function shall fail if:

[EACCES]	A semaphore identifier exists for <i>key</i> , but operation permission as specified by the low-order 9 bits of <i>semflg</i> would not be granted; see Section 2.7 (on page 501).
[EEXIST]	A semaphore identifier exists for the argument <i>key</i> but ((<i>semflg</i> & <code>IPC_CREAT</code>) && (<i>semflg</i> & <code>IPC_EXCL</code>)) is non-zero.
[EINVAL]	The value of <i>nsems</i> is either less than or equal to 0 or greater than the system-imposed limit, or a semaphore identifier exists for the argument <i>key</i> , but the number of semaphores in the set associated with it is less than <i>nsems</i> and <i>nsems</i> is not equal to 0.

59618 [ENOENT] A semaphore identifier does not exist for the argument *key* and (*semflg*
 59619 &IPC_CREAT) is equal to 0.

59620 [ENOSPC] A semaphore identifier is to be created but the system-imposed limit on the
 59621 maximum number of allowed semaphores system-wide would be exceeded.

59622 EXAMPLES

59623 Refer to *semop()*.

59624 APPLICATION USAGE

59625 The POSIX Realtime Extension defines alternative interfaces for interprocess communication.
 59626 Application developers who need to use IPC should design their applications so that modules
 59627 using the IPC routines described in [Section 2.7](#) (on page 501) can be easily modified to use the
 59628 alternative interfaces.

59629 RATIONALE

59630 None.

59631 FUTURE DIRECTIONS

59632 A future version may require that the value of the *semval*, *sempid*, *semncnt*, and *semzcnt* members
 59633 of all semaphores in a semaphore set be initialized to zero when a call to *semget()* creates a
 59634 semaphore set. Many semaphore implementations already do this and it greatly simplifies what
 59635 an application must do to initialize a semaphore set.

59636 SEE ALSO

59637 [Section 2.7](#) (on page 501), [Section 2.8](#) (on page 502), *ftok()*, *semctl()*, *semop()*, *sem_close()*,
 59638 *sem_destroy()*, *sem_getvalue()*, *sem_init()*, *sem_open()*, *sem_post()*, *sem_trywait()*, *sem_unlink()*
 59639 XBD [Section 4.16](#) (on page 113), [<sys/sem.h>](#)

59640 CHANGE HISTORY

59641 First released in Issue 2. Derived from Issue 2 of the SVID.

59642 Issue 5

59643 The note about use of POSIX Realtime Extension IPC routines has been moved from FUTURE
 59644 DIRECTIONS to a new APPLICATION USAGE section.

59645 Issue 6

59646 IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/122 is applied, updating the
 59647 DESCRIPTION from “each semaphore in the set shall not be initialized” to “each semaphore in
 59648 the set need not be initialized”.

59649 Issue 7

59650 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0536 [335,439] and +
 59651 XSH/TC1-2008/0537 [344] are applied.

NAME

semop — XSI semaphore operations

SYNOPSIS

```
XSI    #include <sys/sem.h>
int semop(int semid, struct sembuf *sops, size_t nsops);
```

DESCRIPTION

The *semop()* function operates on XSI semaphores (see XBD [Section 4.16](#), on page 113). It is unspecified whether this function interoperates with the realtime interprocess communication facilities defined in [Section 2.8](#) (on page 502).

The *semop()* function shall perform atomically a user-defined array of semaphore operations in array order on the set of semaphores associated with the semaphore identifier specified by the argument *semid*.

The argument *sops* is a pointer to a user-defined array of semaphore operation structures. The implementation shall not modify elements of this array unless the application uses implementation-defined extensions.

The argument *nsops* is the number of such structures in the array.

Each structure, **sembuf**, includes the following members:

Member Type	Member Name	Description
unsigned short	<i>sem_num</i>	Semaphore number.
short	<i>sem_op</i>	Semaphore operation.
short	<i>sem_flg</i>	Operation flags.

Each semaphore operation specified by *sem_op* is performed on the corresponding semaphore specified by *semid* and *sem_num*.

The variable *sem_op* specifies one of three semaphore operations:

1. If *sem_op* is a negative integer and the calling process has alter permission, one of the following shall occur:
 - If *semval* (see **<sys/sem.h>**) is greater than or equal to the absolute value of *sem_op*, the absolute value of *sem_op* is subtracted from *semval*. Also, if (*sem_flg* & SEM_UNDO) is non-zero, the absolute value of *sem_op* shall be added to the *semadj* value of the calling process for the specified semaphore.
 - If *semval* is less than the absolute value of *sem_op* and (*sem_flg* & IPC_NOWAIT) is non-zero, *semop()* shall return immediately.
 - If *semval* is less than the absolute value of *sem_op* and (*sem_flg* & IPC_NOWAIT) is 0, *semop()* shall increment the *semncnt* associated with the specified semaphore and suspend execution of the calling thread until one of the following conditions occurs:
 - The value of *semval* becomes greater than or equal to the absolute value of *sem_op*. When this occurs, the value of *semncnt* associated with the specified semaphore shall be decremented, the absolute value of *sem_op* shall be subtracted from *semval* and, if (*sem_flg* & SEM_UNDO) is non-zero, the absolute value of *sem_op* shall be added to the *semadj* value of the calling process for the specified semaphore.

- 59693 — The *semid* for which the calling thread is awaiting action is removed from the
59694 system. When this occurs, *errno* shall be set to [EIDRM] and -1 shall be
59695 returned.
- 59696 — The calling thread receives a signal that is to be caught. When this occurs, the
59697 value of *semncnt* associated with the specified semaphore shall be
59698 decremented, and the calling thread shall resume execution in the manner
59699 prescribed in [sigaction\(\)](#).
- 59700 2. If *sem_op* is a positive integer and the calling process has alter permission, the value of
59701 *sem_op* shall be added to *semval* and, if (*sem_flg* & SEM_UNDO) is non-zero, the value of
59702 *sem_op* shall be subtracted from the *semadj* value of the calling process for the specified
59703 semaphore.
- 59704 3. If *sem_op* is 0 and the calling process has read permission, one of the following shall occur:
59705
 - If *semval* is 0, *semop()* shall return immediately.
 - 59706 • If *semval* is non-zero and (*sem_flg* & IPC_NOWAIT) is non-zero, *semop()* shall return
59707 immediately.
 - 59708 • If *semval* is non-zero and (*sem_flg* & IPC_NOWAIT) is 0, *semop()* shall increment the
59709 *semzcnt* associated with the specified semaphore and suspend execution of the
59710 calling thread until one of the following occurs:
 - 59711 — The value of *semval* becomes 0, at which time the value of *semzcnt* associated
59712 with the specified semaphore shall be decremented.
 - 59713 — The *semid* for which the calling thread is awaiting action is removed from the
59714 system. When this occurs, *errno* shall be set to [EIDRM] and -1 shall be
59715 returned.
 - 59716 — The calling thread receives a signal that is to be caught. When this occurs, the
59717 value of *semzcnt* associated with the specified semaphore shall be
59718 decremented, and the calling thread shall resume execution in the manner
59719 prescribed in [sigaction\(\)](#).
- 59720 Upon successful completion, the value of *sempid* for each semaphore specified in the array
59721 pointed to by *sops* shall be set to the process ID of the calling process. Also, the *sem_otime*
59722 timestamp shall be set to the current time, as described in [Section 2.7.1](#) (on page 501).

59723 RETURN VALUE

59724 Upon successful completion, *semop()* shall return 0; otherwise, it shall return -1 and set *errno* to
59725 indicate the error.

59726 ERRORS

59727 The *semop()* function shall fail if:

- 59728 [E2BIG] The value of *nsops* is greater than the system-imposed maximum.
- 59729 [EACCES] Operation permission is denied to the calling process; see [Section 2.7](#) (on page
59730 501).
- 59731 [EAGAIN] The operation would result in suspension of the calling process but (*sem_flg*
59732 & IPC_NOWAIT) is non-zero.
- 59733 [EFBIG] The value of *sem_num* is greater than or equal to the number of semaphores in
59734 the set associated with *semid*.

59735	[EIDRM]	The semaphore identifier <i>semid</i> is removed from the system.
59736	[EINTR]	The <i>semop()</i> function was interrupted by a signal.
59737	[EINVAL]	The value of <i>semid</i> is not a valid semaphore identifier, or the number of individual semaphores for which the calling process requests a SEM_UNDO would exceed the system-imposed limit.
59738		
59739		
59740	[ENOSPC]	The limit on the number of individual processes requesting a SEM_UNDO would be exceeded.
59741		
59742	[ERANGE]	An operation would cause a <i>semval</i> to overflow the system-imposed limit, or an operation would cause a <i>semadj</i> value to overflow the system-imposed limit.
59743		
59744		

59745 EXAMPLES

59746 Setting Values in Semaphores

59747 The following example sets the values of the two semaphores associated with the *semid* identifier
 59748 to the values contained in the *sb* array.

```

59749 #include <sys/sem.h>
59750 ...
59751 int semid;
59752 struct sembuf sb[2];
59753 int nsops = 2;
59754 int result;

59755 /* Code to initialize semid. */
59756 ...
59757 /* Adjust value of semaphore in the semaphore array semid. */
59758 sb[0].sem_num = 0;
59759 sb[0].sem_op = -1;
59760 sb[0].sem_flg = SEM_UNDO | IPC_NOWAIT;
59761 sb[1].sem_num = 1;
59762 sb[1].sem_op = 1;
59763 sb[1].sem_flg = 0;
59764 result = semop(semid, sb, nsops);

```

59765 Creating a Semaphore Identifier

59766 The following example gets a unique semaphore key using the *ftok()* function, then gets a
 59767 semaphore ID associated with that key using the *semget()* function (the first call also tests to
 59768 make sure the semaphore exists). If the semaphore does not exist, the program creates it, as
 59769 shown by the second call to *semget()*. In creating the semaphore for the queuing process, the
 59770 program attempts to create one semaphore with read/write permission for all. It also uses the
 59771 IPC_EXCL flag, which forces *semget()* to fail if the semaphore already exists.

59772 After creating the semaphore, the program uses calls to *semctl()* and *semop()* to initialize it to the
 59773 values in the *sbuf* array. The number of processes that can execute concurrently without queuing
 59774 is initially set to 2. The final call to *semget()* creates a semaphore identifier that can be used later
 59775 in the program.

59776 Processes that obtain *semid* without creating it check that *sem_otime* is non-zero, to ensure that
 59777 the creating process has completed the *semop()* initialization.

```

59778 The final call to semop() acquires the semaphore and waits until it is free; the SEM_UNDO
59779 option releases the semaphore when the process exits, waiting until there are less than two
59780 processes running concurrently.

59781 #include <stdio.h>
59782 #include <sys/sem.h>
59783 #include <sys/stat.h>
59784 #include <errno.h>
59785 #include <stdlib.h>
59786 ...
59787 key_t semkey;
59788 int semid;
59789 struct sembuf sbuf;
59790 union semun {
59791     int val;
59792     struct semid_ds *buf;
59793     unsigned short *array;
59794 } arg;
59795 struct semid_ds ds;
59796 ...
59797 /* Get unique key for semaphore. */
59798 if ((semkey = ftok("/tmp", 'a')) == (key_t) -1) {
59799     perror("IPC error: ftok"); exit(1);
59800 }

59801 /* Get semaphore ID associated with this key. */
59802 if ((semid = semget(semkey, 0, 0)) == -1) {

59803     /* Semaphore does not exist - Create. */
59804     if ((semid = semget(semkey, 1, IPC_CREAT | IPC_EXCL | S_IRUSR |
59805         S_IWUSR | S_IRGRP | S_IWGRP | S_IROTH | S_IWOTH)) != -1)
59806     {
59807         /* Initialize the semaphore. */
59808         arg.val = 0;
59809         sbuf.sem_num = 0;
59810         sbuf.sem_op = 2; /* This is the number of runs without queuing. */
59811         sbuf.sem_flg = 0;
59812         if (semctl(semid, 0, SETVAL, arg) == -1
59813             || semop(semid, &sbuf, 1) == -1) {
59814             perror("IPC error: semop"); exit(1);
59815         }
59816     }
59817     else if (errno == EEXIST) {
59818         if ((semid = semget(semkey, 0, 0)) == -1) {
59819             perror("IPC error 1: semget"); exit(1);
59820         }
59821         goto check_init;
59822     }
59823     else {
59824         perror("IPC error 2: semget"); exit(1);
59825     }
59826 }
59827 else

```

```

59828     {
59829         /* Check that semid has completed initialization. */
59830         /* An application can use a retry loop at this point rather than
59831            exiting. */
59832         check_init:
59833         arg.buf = &ds;
59834         if (semctl(semid, 0, IPC_STAT, arg) < 0) {
59835             perror("IPC error 3: semctl"); exit(1);
59836         }
59837         if (ds.sem_otime == 0) {
59838             perror("IPC error 4: semctl"); exit(1);
59839         }
59840     }
59841     ...
59842     sbuf.sem_num = 0;
59843     sbuf.sem_op = -1;
59844     sbuf.sem_flg = SEM_UNDO;
59845     if (semop(semid, &sbuf, 1) == -1) {
59846         perror("IPC Error: semop"); exit(1);
59847     }

```

APPLICATION USAGE

The POSIX Realtime Extension defines alternative interfaces for interprocess communication. Application developers who need to use IPC should design their applications so that modules using the IPC routines described in [Section 2.7](#) (on page 501) can be easily modified to use the alternative interfaces.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

[Section 2.7](#) (on page 501), [Section 2.8](#) (on page 502), [exec](#), [exit\(\)](#), [fork\(\)](#), [semctl\(\)](#), [semget\(\)](#), [sem_close\(\)](#), [sem_destroy\(\)](#), [sem_getvalue\(\)](#), [sem_init\(\)](#), [sem_open\(\)](#), [sem_post\(\)](#), [sem_trywait\(\)](#), [sem_unlink\(\)](#)

XBD [Section 4.16](#) (on page 113), [<sys/ipc.h>](#), [<sys/sem.h>](#), [<sys/types.h>](#)

CHANGE HISTORY

First released in Issue 2. Derived from Issue 2 of the SVID.

Issue 5

The note about use of POSIX Realtime Extension IPC routines has been moved from FUTURE DIRECTIONS to a new APPLICATION USAGE section.

Issue 7

SD5-XSH-ERN-171 is applied, updating the DESCRIPTION to clarify the order in which the operations in *sops* will be performed when there are multiple operations.

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0538 [329,429], XSH/TC1-2008/0539 [345,428], XSH/TC1-2008/0540 [329,429], XSH/TC1-2008/0541 [335], and XSH/TC1-2008/0542 [291,429] are applied.

59873 **NAME**

59874 send — send a message on a socket

59875 **SYNOPSIS**

59876 #include <sys/socket.h>

59877 ssize_t send(int *socket*, const void **buffer*, size_t *length*, int *flags*);59878 **DESCRIPTION**

59879 The *send()* function shall initiate transmission of a message from the specified socket to its peer.
 59880 The *send()* function shall send a message only when the socket is connected. If the socket is a
 59881 connectionless-mode socket, the message shall be sent to the pre-specified peer address.

59882 The *send()* function takes the following arguments:59883 *socket* Specifies the socket file descriptor.59884 *buffer* Points to the buffer containing the message to send.59885 *length* Specifies the length of the message in bytes.

59886 *flags* Specifies the type of message transmission. Values of this argument are
 59887 formed by logically OR'ing zero or more of the following flags:

59888 MSG_EOR Terminates a record (if supported by the protocol).

59889 MSG_OOB Sends out-of-band data on sockets that support out-of-
 59890 band communications. The significance and semantics
 59891 of out-of-band data are protocol-specific.

59892 MSG_NOSIGNAL Requests not to send the SIGPIPE signal if an attempt to
 59893 send is made on a stream-oriented socket that is no
 59894 longer connected. The [EPIPE] error shall still be
 59895 returned.

59896 The length of the message to be sent is specified by the *length* argument. If the message is too
 59897 long to pass through the underlying protocol, *send()* shall fail and no data shall be transmitted.

59898 Successful completion of a call to *send()* does not guarantee delivery of the message. A return
 59899 value of -1 indicates only locally-detected errors.

59900 If space is not available at the sending socket to hold the message to be transmitted, and the
 59901 socket file descriptor does not have O_NONBLOCK set, *send()* shall block until space is
 59902 available. If space is not available at the sending socket to hold the message to be transmitted,
 59903 and the socket file descriptor does have O_NONBLOCK set, *send()* shall fail. The *select()* and
 59904 *poll()* functions can be used to determine when it is possible to send more data.

59905 The socket in use may require the process to have appropriate privileges to use the *send()*
 59906 function.

59907 **RETURN VALUE**

59908 Upon successful completion, *send()* shall return the number of bytes sent. Otherwise, -1 shall be
 59909 returned and *errno* set to indicate the error.

59910 **ERRORS**59911 The *send()* function shall fail if:

59912 [EAGAIN] or [EWOULDBLOCK]

59913 The socket's file descriptor is marked O_NONBLOCK and the requested
 59914 operation would block.

59915	[EBADF]	The <i>socket</i> argument is not a valid file descriptor.
59916	[ECONNRESET]	A connection was forcibly closed by a peer.
59917	[EDESTADDRREQ]	
59918		The socket is not connection-mode and no peer address is set.
59919	[EINTR]	A signal interrupted <i>send()</i> before any data was transmitted.
59920	[EMSGSIZE]	The message is too large to be sent all at once, as the socket requires.
59921	[ENOTCONN]	The socket is not connected.
59922	[ENOTSOCK]	The <i>socket</i> argument does not refer to a socket.
59923	[EOPNOTSUPP]	The <i>socket</i> argument is associated with a socket that does not support one or more of the values set in <i>flags</i> .
59924		
59925	[EPIPE]	The socket is shut down for writing, or the socket is connection-mode and is no longer connected. In the latter case, and if the socket is of type SOCK_STREAM or SOCK_SEQPACKET and the MSG_NOSIGNAL flag is not set, the SIGPIPE signal is generated to the calling thread.
59926		
59927		
59928		
59929		The <i>send()</i> function may fail if:
59930	[EACCES]	The calling process does not have appropriate privileges.
59931	[EIO]	An I/O error occurred while reading from or writing to the file system.
59932	[ENETDOWN]	The local network interface used to reach the destination is down.
59933	[ENETUNREACH]	
59934		No route to the network is present.
59935	[ENOBUFS]	Insufficient resources were available in the system to perform the operation.

59936 EXAMPLES

59937 None.

59938 APPLICATION USAGE

59939 If the *socket* argument refers to a connection-mode socket, the *send()* function is equivalent to
 59940 *sendto()* (with any value for the *dest_addr* and *dest_len* arguments, as they are ignored in this
 59941 case). If the *socket* argument refers to a socket and the *flags* argument is 0, the *send()* function is
 59942 equivalent to *write()*.

59943 RATIONALE

59944 None.

59945 FUTURE DIRECTIONS

59946 None.

59947 SEE ALSO

59948 *connect()*, *getsockopt()*, *poll()*, *pselect()*, *recv()*, *recvfrom()*, *recvmsg()*, *sendmsg()*, *sendto()*,
 59949 *setsockopt()*, *shutdown()*, *socket()*, *write()*

59950 XBD <sys/socket.h>

59951 CHANGE HISTORY

59952 First released in Issue 6. Derived from the XNS, Issue 5.2 specification.

59953 **Issue 7**

59954 Austin Group Interpretation 1003.1-2001 #035 is applied, updating the DESCRIPTION to clarify
59955 the behavior when the socket is a connectionless-mode socket.

59956 The MSG_NOSIGNAL flag is added from The Open Group Technical Standard, 2006, Extended
59957 API Set Part 2.

59958 The [EPIPE] error is modified.

59959 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0543 [463] is applied.

+

59960 **NAME**59961 `sendmsg` — send a message on a socket using a message structure59962 **SYNOPSIS**59963 `#include <sys/socket.h>`59964 `ssize_t sendmsg(int socket, const struct msghdr *message, int flags);`59965 **DESCRIPTION**

59966 The `sendmsg()` function shall send a message through a connection-mode or connectionless-mode socket. If the socket is a connectionless-mode socket, the message shall be sent to the address specified by **msghdr** if no pre-specified peer address has been set. If a peer address has been pre-specified, either the message shall be sent to the address specified in **msghdr** (overriding the pre-specified peer address), or the function shall return `-1` and set `errno` to `[EISCONN]`. If the socket is connection-mode, the destination address in **msghdr** shall be ignored.

59973 The `sendmsg()` function takes the following arguments:

59974	<i>socket</i>	Specifies the socket file descriptor.
59975	<i>message</i>	Points to a msghdr structure, containing both the destination address and the buffers for the outgoing message. The length and format of the address depend on the address family of the socket. The <i>msg_flags</i> member is ignored.
59976		
59977		
59978	<i>flags</i>	Specifies the type of message transmission. The application may specify 0 or the following flag:
59979		
59980	<code>MSG_EOR</code>	Terminates a record (if supported by the protocol).
59981	<code>MSG_OOB</code>	Sends out-of-band data on sockets that support out-of-band data. The significance and semantics of out-of-band data are protocol-specific.
59982		
59983		
59984	<code>MSG_NOSIGNAL</code>	Requests not to send the SIGPIPE signal if an attempt to send is made on a stream-oriented socket that is no longer connected. The <code>[EPIPE]</code> error shall still be returned.
59985		
59986		
59987		

59988 The *msg_iov* and *msg_iovlen* fields of *message* specify zero or more buffers containing the data to be sent. *msg_iov* points to an array of **iovec** structures; *msg_iovlen* shall be set to the dimension of this array. In each **iovec** structure, the *iov_base* field specifies a storage area and the *iov_len* field gives its size in bytes. Some of these sizes can be zero. The data from each storage area indicated by *msg_iov* is sent in turn.

59993 Successful completion of a call to `sendmsg()` does not guarantee delivery of the message. A return value of `-1` indicates only locally-detected errors.

59995 If space is not available at the sending socket to hold the message to be transmitted and the socket file descriptor does not have `O_NONBLOCK` set, the `sendmsg()` function shall block until space is available. If space is not available at the sending socket to hold the message to be transmitted and the socket file descriptor does have `O_NONBLOCK` set, the `sendmsg()` function shall fail.

60000 If the socket protocol supports broadcast and the specified address is a broadcast address for the socket protocol, `sendmsg()` shall fail if the `SO_BROADCAST` option is not set for the socket.

60002 The socket in use may require the process to have appropriate privileges to use the `sendmsg()` function.

60004 **RETURN VALUE**

60005 Upon successful completion, *sendmsg()* shall return the number of bytes sent. Otherwise, *-1*
60006 shall be returned and *errno* set to indicate the error.

60007 **ERRORS**

60008 The *sendmsg()* function shall fail if:

60009 [EAGAIN] or [EWOULDBLOCK]

60010 The socket's file descriptor is marked O_NONBLOCK and the requested
60011 operation would block.

60012 [EAFNOSUPPORT]

60013 Addresses in the specified address family cannot be used with this socket.

60014 [EBADF] The *socket* argument is not a valid file descriptor.

60015 [ECONNRESET] A connection was forcibly closed by a peer.

60016 [EINTR] A signal interrupted *sendmsg()* before any data was transmitted.

60017 [EINVAL] The sum of the *iov_len* values overflows an *ssize_t*.

60018 [EMSGSIZE] The message is too large to be sent all at once (as the socket requires), or the
60019 *msg_iovlen* member of the **msghdr** structure pointed to by *message* is less than
60020 or equal to 0 or is greater than {IOV_MAX}.

60021 [ENOTCONN] The socket is connection-mode but is not connected.

60022 [ENOTSOCK] The *socket* argument does not refer to a socket.

60023 [EOPNOTSUPP] The *socket* argument is associated with a socket that does not support one or
60024 more of the values set in *flags*.

60025 [EPIPE] The socket is shut down for writing, or the socket is connection-mode and is
60026 no longer connected. In the latter case, and if the socket is of type
60027 SOCK_STREAM or SOCK_SEQPACKET and the MSG_NOSIGNAL flag is not
60028 set, the SIGPIPE signal is generated to the calling thread.

60029 If the address family of the socket is AF_UNIX, then *sendmsg()* shall fail if:

60030 [EIO] An I/O error occurred while reading from or writing to the file system.

60031 [ELOOP] A loop exists in symbolic links encountered during resolution of the pathname
60032 in the socket address.

60033 [ENAMETOOLONG]

60034 The length of a component of a pathname is longer than {NAME_MAX}.

60035 [ENOENT] A component of the pathname does not name an existing file or the path name
60036 is an empty string.

60037 [ENOTDIR] A component of the path prefix of the pathname in the socket address names
60038 an existing file that is neither a directory nor a symbolic link to a directory, or
60039 the pathname in the socket address contains at least one non-*<slash>* character
60040 and ends with one or more trailing *<slash>* characters and the last pathname
60041 component names an existing file that is neither a directory nor a symbolic
60042 link to a directory.

60043 The *sendmsg()* function may fail if:

60044 [EACCES] Search permission is denied for a component of the path prefix; or write access
60045 to the named socket is denied.

60046 [EDESTADDRREQ]
60047 The socket is not connection-mode and does not have its peer address set, and
60048 no destination address was specified.

60049 [EHOSTUNREACH]
60050 The destination host cannot be reached (probably because the host is down or
60051 a remote router cannot reach it).

60052 [EIO] An I/O error occurred while reading from or writing to the file system.

60053 [EISCONN] A destination address was specified and the socket is already connected.

60054 [ENETDOWN] The local network interface used to reach the destination is down.

60055 [ENETUNREACH]
60056 No route to the network is present.

60057 [ENOBUFS] Insufficient resources were available in the system to perform the operation.

60058 [ENOMEM] Insufficient memory was available to fulfill the request.

60059 If the address family of the socket is AF_UNIX, then *sendmsg()* may fail if:

60060 [ELOOP] More than {SYMLOOP_MAX} symbolic links were encountered during
60061 resolution of the pathname in the socket address.

60062 [ENAMETOOLONG]
60063 The length of a pathname exceeds {PATH_MAX}, or pathname resolution of a
60064 symbolic link produced an intermediate result with a length that exceeds
60065 {PATH_MAX}.

EXAMPLES

60066 Done.

APPLICATION USAGE

60068 The *select()* and *poll()* functions can be used to determine when it is possible to send more data.

RATIONALE

60070 None.

FUTURE DIRECTIONS

60072 None.

SEE ALSO

60075 *getsockopt()*, *poll()*, *pselect()*, *recv()*, *recvfrom()*, *recvmsg()*, *send()*, *sendto()*, *setsockopt()*,
60076 *shutdown()*, *socket()*

60077 XBD <sys/socket.h>

CHANGE HISTORY

60078 First released in Issue 6. Derived from the XNS, Issue 5.2 specification.

60080 The wording of the mandatory [ELOOP] error condition is updated, and a second optional
60081 [ELOOP] error condition is added.

Issue 7

- 60082 Austin Group Interpretation 1003.1-2001 #073 is applied, updating the DESCRIPTION.
- 60083
- 60084 Austin Group Interpretation 1003.1-2001 #143 is applied.
- 60085 The MSG_NOSIGNAL flag is added from The Open Group Technical Standard, 2006, Extended
- 60086 API Set Part 2.
- 60087 The [EPIPE] error is modified.
- 60088 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0544 [324] is applied. +

60089 **NAME**60090 `sendto` — send a message on a socket60091 **SYNOPSIS**

```
60092 #include <sys/socket.h>
60093 ssize_t sendto(int socket, const void *message, size_t length,
60094               int flags, const struct sockaddr *dest_addr,
60095               socklen_t dest_len);
```

60096 **DESCRIPTION**

60097 The `sendto()` function shall send a message through a connection-mode or connectionless-mode
 60098 socket.

60099 If the socket is a connectionless-mode socket, the message shall be sent to the address specified
 60100 by `dest_addr` if no pre-specified peer address has been set. If a peer address has been pre-
 60101 specified, either the message shall be sent to the address specified by `dest_addr` (overriding the
 60102 pre-specified peer address), or the function shall return `-1` and set `errno` to `[EISCONN]`.

60103 If the socket is connection-mode, `dest_addr` shall be ignored.

60104 The `sendto()` function takes the following arguments:

60105	<i>socket</i>	Specifies the socket file descriptor.
60106	<i>message</i>	Points to a buffer containing the message to be sent.
60107	<i>length</i>	Specifies the size of the message in bytes.
60108	<i>flags</i>	Specifies the type of message transmission. Values of this argument are 60109 formed by logically OR'ing zero or more of the following flags:
60110		MSG_EOR Terminates a record (if supported by the protocol).
60111		MSG_OOB Sends out-of-band data on sockets that support out-of- 60112 band data. The significance and semantics of out-of- 60113 band data are protocol-specific.
60114		MSG_NOSIGNAL Requests not to send the SIGPIPE signal if an attempt to 60115 send is made on a stream-oriented socket that is no 60116 longer connected. The <code>[EPIPE]</code> error shall still be 60117 returned.
60118	<i>dest_addr</i>	Points to a sockaddr structure containing the destination address. The length 60119 and format of the address depend on the address family of the socket.
60120	<i>dest_len</i>	Specifies the length of the sockaddr structure pointed to by the <code>dest_addr</code> 60121 argument.

60122 If the socket protocol supports broadcast and the specified address is a broadcast address for the
 60123 socket protocol, `sendto()` shall fail if the `SO_BROADCAST` option is not set for the socket.

60124 The `dest_addr` argument specifies the address of the target.

60125 The `length` argument specifies the length of the message.

60126 Successful completion of a call to `sendto()` does not guarantee delivery of the message. A return
 60127 value of `-1` indicates only locally-detected errors.

60128 If space is not available at the sending socket to hold the message to be transmitted and the
 60129 socket file descriptor does not have `O_NONBLOCK` set, `sendto()` shall block until space is
 60130 available. If space is not available at the sending socket to hold the message to be transmitted

60131 and the socket file descriptor does have O_NONBLOCK set, *sendto()* shall fail.
 60132 The socket in use may require the process to have appropriate privileges to use the *sendto()*
 60133 function.

60134 **RETURN VALUE**

60135 Upon successful completion, *sendto()* shall return the number of bytes sent. Otherwise, -1 shall
 60136 be returned and *errno* set to indicate the error.

60137 **ERRORS**

60138 The *sendto()* function shall fail if:

60139 [EAFNOSUPPORT]

60140 Addresses in the specified address family cannot be used with this socket.

60141 [EAGAIN] or [EWOULDBLOCK]

60142 The socket's file descriptor is marked O_NONBLOCK and the requested
 60143 operation would block.

60144 [EBADF] The *socket* argument is not a valid file descriptor.

60145 [ECONNRESET] A connection was forcibly closed by a peer.

60146 [EINTR] A signal interrupted *sendto()* before any data was transmitted.

60147 [EMSGSIZE] The message is too large to be sent all at once, as the socket requires.

60148 [ENOTCONN] The socket is connection-mode but is not connected.

60149 [ENOTSOCK] The *socket* argument does not refer to a socket.

60150 [EOPNOTSUPP] The *socket* argument is associated with a socket that does not support one or
 60151 more of the values set in *flags*.

60152 [EPIPE] The socket is shut down for writing, or the socket is connection-mode and is
 60153 no longer connected. In the latter case, and if the socket is of type
 60154 SOCK_STREAM or SOCK_SEQPACKET and the MSG_NOSIGNAL flag is not
 60155 set, the SIGPIPE signal is generated to the calling thread.

60156 If the address family of the socket is AF_UNIX, then *sendto()* shall fail if:

60157 [EIO] An I/O error occurred while reading from or writing to the file system.

60158 [ELOOP] A loop exists in symbolic links encountered during resolution of the pathname
 60159 in the socket address.

60160 [ENAMETOOLONG]

60161 The length of a component of a pathname is longer than {NAME_MAX}.

60162 [ENOENT] A component of the pathname does not name an existing file or the pathname
 60163 is an empty string.

60164 [ENOTDIR] A component of the path prefix of the pathname in the socket address names
 60165 an existing file that is neither a directory nor a symbolic link to a directory, or
 60166 the pathname in the socket address contains at least one non-*<slash>* character
 60167 and ends with one or more trailing *<slash>* characters and the last pathname
 60168 component names an existing file that is neither a directory nor a symbolic
 60169 link to a directory.

60170 The *sendto()* function may fail if:

60171 [EACCES] Search permission is denied for a component of the path prefix; or write access
60172 to the named socket is denied.

60173 [EDESTADDRREQ]
60174 The socket is not connection-mode and does not have its peer address set, and
60175 no destination address was specified.

60176 [EHOSTUNREACH]
60177 The destination host cannot be reached (probably because the host is down or
60178 a remote router cannot reach it).

60179 [EINVAL] The *dest_len* argument is not a valid length for the address family.

60180 [EIO] An I/O error occurred while reading from or writing to the file system.

60181 [EISCONN] A destination address was specified and the socket is already connected.

60182 [ENETDOWN] The local network interface used to reach the destination is down.

60183 [ENETUNREACH]
60184 No route to the network is present.

60185 [ENOBUFS] Insufficient resources were available in the system to perform the operation.

60186 [ENOMEM] Insufficient memory was available to fulfill the request.

60187 If the address family of the socket is AF_UNIX, then *sendto()* may fail if:

60188 [ELOOP] More than {SYMLOOP_MAX} symbolic links were encountered during
60189 resolution of the pathname in the socket address.

60190 [ENAMETOOLONG]
60191 The length of a pathname exceeds {PATH_MAX}, or pathname resolution of a
60192 symbolic link produced an intermediate result with a length that exceeds
60193 {PATH_MAX}.

60194 EXAMPLES

60195 None.

60196 APPLICATION USAGE

60197 The *select()* and *poll()* functions can be used to determine when it is possible to send more data.

60198 RATIONALE

60199 None.

60200 FUTURE DIRECTIONS

60201 None.

60202 SEE ALSO

60203 *getsockopt()*, *poll()*, *pselect()*, *recv()*, *recvfrom()*, *recvmsg()*, *send()*, *sendmsg()*, *setsockopt()*,
60204 *shutdown()*, *socket()*

60205 XBD <sys/socket.h>

60206 CHANGE HISTORY

60207 First released in Issue 6. Derived from the XNS, Issue 5.2 specification.

60208 The wording of the mandatory [ELOOP] error condition is updated, and a second optional
60209 [ELOOP] error condition is added.

60210 **Issue 7**

60211 Austin Group Interpretations 1003.1-2001 #035 and #073 are applied, updating the [EISCONN]
60212 error and the DESCRIPTION.

60213 Austin Group Interpretation 1003.1-2001 #143 is applied, clarifying the [ENAMETOOLONG]
60214 error condition.

60215 The MSG_NOSIGNAL flag is added from The Open Group Technical Standard, 2006, Extended
60216 API Set Part 2.

60217 The [EPIPE] error is modified.

60218 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0545 [324] is applied.

+

NAME

setbuf — assign buffering to a stream

SYNOPSIS

```
#include <stdio.h>
```

```
void setbuf(FILE *restrict stream, char *restrict buf);
```

DESCRIPTION

CX The functionality described on this reference page is aligned with the ISO C standard. Any conflict between the requirements described here and the ISO C standard is unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

Except that it returns no value, the function call:

```
setbuf(stream, buf)
```

shall be equivalent to:

```
setvbuf(stream, buf, _IOFBF, BUFSIZ)
```

if *buf* is not a null pointer, or to:

```
setvbuf(stream, buf, _IONBF, BUFSIZ)
```

if *buf* is a null pointer.

RETURN VALUE

The *setbuf()* function shall not return a value.

ERRORS

Although the *setvbuf()* interface may set *errno* in defined ways, the value of *errno* after a call to *setbuf()* is unspecified.

EXAMPLES

None.

APPLICATION USAGE

A common source of error is allocating buffer space as an “automatic” variable in a code block, and then failing to close the stream in the same block.

With *setbuf()*, allocating a buffer of BUFSIZ bytes does not necessarily imply that all of BUFSIZ bytes are used for the buffer area.

Since *errno* is not required to be unchanged on success, in order to correctly detect and possibly recover from errors, applications should use *setvbuf()* instead of *setbuf()*.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

[Section 2.5](#) (on page 495), *fopen()*, *setvbuf()*

XBD [<stdio.h>](#)

CHANGE HISTORY

First released in Issue 1. Derived from Issue 1 of the SVID.

60258 **Issue 6**

60259 The prototype for *setbuf()* is updated for alignment with the ISO/IEC 9899:1999 standard.

60260 **Issue 7**

60261 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0546 [397], XSH/TC1-2008/0547 [397], +
60262 and XSH/TC1-2008/0548 [14] are applied.

NAME

setegid — set the effective group ID

SYNOPSIS

```
#include <unistd.h>
```

```
int setegid(gid_t gid);
```

DESCRIPTION

If *gid* is equal to the real group ID or the saved set-group-ID, or if the process has appropriate privileges, *setegid()* shall set the effective group ID of the calling process to *gid*; the real group ID, saved set-group-ID, and any supplementary group IDs shall remain unchanged.

The *setegid()* function shall not affect the supplementary group list in any way.

RETURN VALUE

Upon successful completion, 0 shall be returned; otherwise, -1 shall be returned and *errno* set to indicate the error.

ERRORS

The *setegid()* function shall fail if:

[EINVAL]	The value of the <i>gid</i> argument is invalid and is not supported by the implementation.
[EPERM]	The process does not have appropriate privileges and <i>gid</i> does not match the real group ID or the saved set-group-ID.

EXAMPLES

None.

APPLICATION USAGE

None.

RATIONALE

Refer to the RATIONALE section in *setuid()*.

FUTURE DIRECTIONS

None.

SEE ALSO

exec, *getegid()*, *geteuid()*, *getgid()*, *getuid()*, *seteuid()*, *setgid()*, *setregid()*, *setreuid()*, *setuid()*

XBD [<sys/types.h>](#), [<unistd.h>](#)

CHANGE HISTORY

First released in Issue 6. Derived from the IEEE P1003.1a draft standard.

NAME

setenv — add or change environment variable

SYNOPSIS

```
#include <stdlib.h>

int setenv(const char *envname, const char *envval, int overwrite);
```

DESCRIPTION

The *setenv()* function shall update or add a variable in the environment of the calling process. The *envname* argument points to a string containing the name of an environment variable to be added or altered. The environment variable shall be set to the value to which *envval* points. The function shall fail if *envname* points to a string which contains an '=' character. If the environment variable named by *envname* already exists and the value of *overwrite* is non-zero, the function shall return success and the environment shall be updated. If the environment variable named by *envname* already exists and the value of *overwrite* is zero, the function shall return success and the environment shall remain unchanged.

The *setenv()* function shall update the list of pointers to which *environ* points.

The strings described by *envname* and *envval* are copied by this function.

The *setenv()* function need not be thread-safe.

RETURN VALUE

Upon successful completion, zero shall be returned. Otherwise, -1 shall be returned, *errno* set to indicate the error, and the environment shall be unchanged.

ERRORS

The *setenv()* function shall fail if:

[EINVAL] The *envname* argument points to an empty string or points to a string containing an '=' character.

[ENOMEM] Insufficient memory was available to add a variable or its value to the environment.

EXAMPLES

None.

APPLICATION USAGE

See *exec()* for restrictions on changing the environment in multi-threaded applications.

RATIONALE

Unanticipated results may occur if *setenv()* changes the external variable *environ*. In particular, if the optional *envp* argument to *main()* is present, it is not changed, and thus may point to an obsolete copy of the environment (as may any other copy of *environ*). However, other than the aforementioned restriction, the standard developers intended that the traditional method of walking through the environment by way of the *environ* pointer must be supported.

It was decided that *setenv()* should be required by this version because it addresses a piece of missing functionality, and does not impose a significant burden on the implementor.

There was considerable debate as to whether the System V *putenv()* function or the BSD *setenv()* function should be required as a mandatory function. The *setenv()* function was chosen because it permitted the implementation of the *unsetenv()* function to delete environmental variables, without specifying an additional interface. The *putenv()* function is available as part of the XSI option.

60338 The standard developers considered requiring that *setenv()* indicate an error when a call to it
60339 would result in exceeding {ARG_MAX}. The requirement was rejected since the condition might
60340 be temporary, with the application eventually reducing the environment size. The ultimate
60341 success or failure depends on the size at the time of a call to *exec*, which returns an indication of
60342 this error condition.

60343 See also the RATIONALE section in *getenv()*.

+

60344 FUTURE DIRECTIONS

60345 None.

60346 SEE ALSO

60347 *exec*, *getenv()*, *putenv()*, *unsetenv()*

60348 XBD *<stdlib.h>*, *<sys/types.h>*, *<unistd.h>*

60349 CHANGE HISTORY

60350 First released in Issue 6. Derived from the IEEE P1003.1a draft standard.

60351 IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/55 is applied, adding references to *exec* in
60352 the APPLICATION USAGE and SEE ALSO sections.

60353 Issue 7

60354 Austin Group Interpretation 1003.1-2001 #156 is applied.

60355 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0549 [167], XSH/TC1-2008/0550 [185], +
60356 XSH/TC1-2008/0551 [167], and XSH/TC1-2008/0552 [38] are applied.

NAME

seteuid — set effective user ID

SYNOPSIS

```
#include <unistd.h>
```

```
int seteuid(uid_t uid);
```

DESCRIPTION

If *uid* is equal to the real user ID or the saved set-user-ID, or if the process has appropriate privileges, *seteuid()* shall set the effective user ID of the calling process to *uid*; the real user ID and saved set-user-ID shall remain unchanged.

The *seteuid()* function shall not affect the supplementary group list in any way.

RETURN VALUE

Upon successful completion, 0 shall be returned; otherwise, -1 shall be returned and *errno* set to indicate the error.

ERRORS

The *seteuid()* function shall fail if:

[EINVAL] The value of the *uid* argument is invalid and is not supported by the implementation.

[EPERM] The process does not have appropriate privileges and *uid* does not match the real user ID or the saved set-user-ID.

EXAMPLES

None.

APPLICATION USAGE

None.

RATIONALE

Refer to the RATIONALE section in *setuid()*.

FUTURE DIRECTIONS

None.

SEE ALSO

exec, *getegid()*, *geteuid()*, *getgid()*, *getuid()*, *setegid()*, *setgid()*, *setregid()*, *setreuid()*, *setuid()*

XBD *<sys/types.h>*, *<unistd.h>*

CHANGE HISTORY

First released in Issue 6. Derived from the IEEE P1003.1a draft standard.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/123 is applied, making an editorial correction to the [EPERM] error in the ERRORS section.

60391 **NAME**

60392 setgid — set-group-ID

60393 **SYNOPSIS**

60394 #include <unistd.h>

60395 int setgid(gid_t *gid*);60396 **DESCRIPTION**60397 If the process has appropriate privileges, *setgid()* shall set the real group ID, effective group ID,
60398 and the saved set-group-ID of the calling process to *gid*.60399 If the process does not have appropriate privileges, but *gid* is equal to the real group ID or the
60400 saved set-group-ID, *setgid()* shall set the effective group ID to *gid*; the real group ID and saved
60401 set-group-ID shall remain unchanged.60402 The *setgid()* function shall not affect the supplementary group list in any way.

60403 Any supplementary group IDs of the calling process shall remain unchanged.

60404 **RETURN VALUE**60405 Upon successful completion, 0 is returned. Otherwise, -1 shall be returned and *errno* set to
60406 indicate the error.60407 **ERRORS**60408 The *setgid()* function shall fail if:60409 [EINVAL] The value of the *gid* argument is invalid and is not supported by the
60410 implementation.60411 [EPERM] The process does not have appropriate privileges and *gid* does not match the
60412 real group ID or the saved set-group-ID.60413 **EXAMPLES**

60414 None.

60415 **APPLICATION USAGE**

60416 None.

60417 **RATIONALE**60418 Refer to the RATIONALE section in *setuid()*.60419 **FUTURE DIRECTIONS**

60420 None.

60421 **SEE ALSO**60422 *exec*, *getegid()*, *geteuid()*, *getgid()*, *getuid()*, *setegid()*, *seteuid()*, *setregid()*, *setreuid()*, *setuid()*

60423 XBD <sys/types.h>, <unistd.h>

60424 **CHANGE HISTORY**

60425 First released in Issue 1. Derived from Issue 1 of the SVID.

60426 **Issue 6**

60427 In the SYNOPSIS, the optional include of the <sys/types.h> header is removed.

60428 The following new requirements on POSIX implementations derive from alignment with the
60429 Single UNIX Specification:

- 60430 • The requirement to include <sys/types.h> has been removed. Although <sys/types.h> was
60431 required for conforming implementations of previous POSIX specifications, it was not
60432 required for UNIX applications.

- 60433 • Functionality associated with `_POSIX_SAVED_IDS` is now mandated. This is a FIPS
60434 requirement.

60435 The following changes were made to align with the IEEE P1003.1a draft standard:

- 60436 • The effects of `setgid()` in processes without appropriate privileges are changed.
- 60437 • A requirement that the supplementary group list is not affected is added.

60438 **NAME**

60439 setgrent — reset the group database to the first entry

60440 **SYNOPSIS**

```
60441 XSI      #include <grp.h>
60442          void setgrent(void);
```

60443 **DESCRIPTION**60444 Refer to *endgrent()*.

60445 **NAME**
60446 sethostent — network host database functions

60447 **SYNOPSIS**
60448 #include <netdb.h>
60449 void sethostent(int *stayopen*);

60450 **DESCRIPTION**
60451 Refer to *endhostent()*.

60452 **NAME**

60453 setitimer — set the value of an interval timer

60454 **SYNOPSIS**

60455 OB XSI #include <sys/time.h>

```
60456       int setitimer(int which, const struct itimerval *restrict value,  
60457                    struct itimerval *restrict ovalue);
```

60458 **DESCRIPTION**60459 Refer to *getitimer()*.

NAME

setjmp — set jump point for a non-local goto

SYNOPSIS

```
#include <setjmp.h>
```

```
int setjmp(jmp_buf env);
```

DESCRIPTION

The functionality described on this reference page is aligned with the ISO C standard. Any conflict between the requirements described here and the ISO C standard is unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

A call to *setjmp()* shall save the calling environment in its *env* argument for later use by *longjmp()*.

It is unspecified whether *setjmp()* is a macro or a function. If a macro definition is suppressed in order to access an actual function, or a program defines an external identifier with the name *setjmp*, the behavior is undefined.

An application shall ensure that an invocation of *setjmp()* appears in one of the following contexts only:

- The entire controlling expression of a selection or iteration statement
- One operand of a relational or equality operator with the other operand an integral constant expression, with the resulting expression being the entire controlling expression of a selection or iteration statement
- The operand of a unary '!' operator with the resulting expression being the entire controlling expression of a selection or iteration
- The entire expression of an expression statement (possibly cast to **void**)

If the invocation appears in any other context, the behavior is undefined.

RETURN VALUE

If the return is from a direct invocation, *setjmp()* shall return 0. If the return is from a call to *longjmp()*, *setjmp()* shall return a non-zero value.

ERRORS

No errors are defined.

EXAMPLES

None.

APPLICATION USAGE

In general, *sigsetjmp()* is more useful in dealing with errors and interrupts encountered in a low-level subroutine of a program.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

longjmp(), *sigsetjmp()*

XBD <setjmp.h>

60501	CHANGE HISTORY
60502	First released in Issue 1. Derived from Issue 1 of the SVID.
60503	Issue 6
60504	The normative text is updated to avoid use of the term “must” for application requirements.

NAME

setkey — set encoding key (**CRYPT**)

SYNOPSIS

```
XSI      #include <stdlib.h>
void setkey(const char *key);
```

DESCRIPTION

The *setkey()* function provides access to an implementation-defined encoding algorithm. The argument of *setkey()* is an array of length 64 bytes containing only the bytes with numerical value of 0 and 1. If this string is divided into groups of 8, the low-order bit in each group is ignored; this gives a 56-bit key which is used by the algorithm. This is the key that shall be used with the algorithm to encode a string *block* passed to *encrypt()*.

The *setkey()* function shall not change the setting of *errno* if successful. An application wishing to check for error situations should set *errno* to 0 before calling *setkey()*. If *errno* is non-zero on return, an error has occurred.

The *setkey()* function need not be thread-safe.

RETURN VALUE

No values are returned.

ERRORS

The *setkey()* function shall fail if:

[ENOSYS] The functionality is not supported on this implementation.

EXAMPLES

None.

APPLICATION USAGE

Decoding need not be implemented in all environments. This is related to government restrictions in some countries on encryption and decryption routines. Historical practice has been to ship a different version of the encryption library without the decryption feature in the routines supplied. Thus the exported version of *encrypt()* does encoding but not decoding.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

crypt(), *encrypt()*

XBD *<stdlib.h>*

CHANGE HISTORY

First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 5

The DESCRIPTION is updated to indicate that *errno* is not changed if the function is successful.

Issue 7

Austin Group Interpretation 1003.1-2001 #156 is applied.

60545 **NAME**

60546 setlocale — set program locale

60547 **SYNOPSIS**

60548 #include <locale.h>

60549 char *setlocale(int *category*, const char **locale*);60550 **DESCRIPTION**

60551 CX The functionality described on this reference page is aligned with the ISO C standard. Any
 60552 conflict between the requirements described here and the ISO C standard is unintentional. This
 60553 volume of POSIX.1-2008 defers to the ISO C standard.

60554 The *setlocale()* function selects the appropriate piece of the global locale, as specified by the
 60555 *category* and *locale* arguments, and can be used to change or query the entire global locale or
 60556 portions thereof. The value LC_ALL for *category* names the entire global locale; other values for
 60557 *category* name only a part of the global locale:

60558 LC_COLLATE Affects the behavior of regular expressions and the collation functions.

60559 LC_CTYPE Affects the behavior of regular expressions, character classification, character
 60560 conversion functions, and wide-character functions.

60561 CX LC_MESSAGES Affects the affirmative and negative response expressions returned by
 60562 *nl_langinfo()* and the way message catalogs are located. It may also affect the
 60563 behavior of functions that return or write message strings.

60564 LC_MONETARY Affects the behavior of functions that handle monetary values.

60565 LC_NUMERIC Affects the behavior of functions that handle numeric values.

60566 LC_TIME Affects the behavior of the time conversion functions.

60567 The *locale* argument is a pointer to a character string containing the required setting of *category*.
 60568 The contents of this string are implementation-defined. In addition, the following preset values
 60569 of *locale* are defined for all settings of *category*:

60570 CX "POSIX" Specifies the minimal environment for C-language translation called the
 60571 POSIX locale. The POSIX locale is the default global locale at entry to *main()*.

60572 "C" Equivalent to "POSIX".

60573 CX "" Specifies an implementation-defined native environment. The determination
 60574 of the name of the new locale for the specified category depends on the value
 60575 of the associated environment variables, *LC_** and *LANG*; see XBD Chapter 7
 60576 (on page 135) and Chapter 8 (on page 173).

60577 A null pointer Directs *setlocale()* to query the current global locale setting and return the
 60578 name of the locale if *category* is not LC_ALL, or a string which encodes the
 60579 locale name(s) for all of the individual categories if *category* is LC_ALL.

60580 CX Setting all of the categories of the global locale is similar to successively setting each individual
 60581 category of the global locale, except that all error checking is done before any actions are
 60582 performed. To set all the categories of the global locale, *setlocale()* can be invoked as:

60583 setlocale(LC_ALL, "");

60584 In this case, *setlocale()* shall first verify that the values of all the environment variables it needs
 60585 according to the precedence rules (described in XBD Chapter 8, on page 173) indicate supported
 60586 locales. If the value of any of these environment variable searches yields a locale that is not
 60587 supported (and non-null), *setlocale()* shall return a null pointer and the global locale shall not be

changed. If all environment variables name supported locales, *setlocale()* shall proceed as if it had been called for each category, using the appropriate value from the associated environment variable or from the implementation-defined default if there is no such value.

The global locale established using *setlocale()* shall only be used in threads for which no current locale has been set using *uselocale()* or whose current locale has been set to the global locale using *uselocale(LC_GLOBAL_LOCALE)*.

The implementation shall behave as if no function defined in this volume of POSIX.1-2008 calls *setlocale()*.

RETURN VALUE

Upon successful completion, *setlocale()* shall return the string associated with the specified category for the new locale. Otherwise, *setlocale()* shall return a null pointer and the global locale shall not be changed.

A null pointer for *locale* shall cause *setlocale()* to return a pointer to the string associated with the specified *category* for the current global locale. The global locale shall not be changed.

The string returned by *setlocale()* is such that a subsequent call with that string and its associated *category* shall restore that part of the global locale. The application shall not modify the string returned. The returned string pointer might be invalidated or the string content might be overwritten by a subsequent call to *setlocale()*.

ERRORS

No errors are defined.

EXAMPLES

None.

APPLICATION USAGE

The following code illustrates how a program can initialize the international environment for one language, while selectively modifying the global locale such that regular expressions and string operations can be applied to text recorded in a different language:

```
setlocale(LC_ALL, "De");
setlocale(LC_COLLATE, "Fr@dict");
```

Internationalized programs can initiate language operation according to environment variable settings (see XBD [Section 8.2](#), on page 174) by calling *setlocale()* as follows:

```
setlocale(LC_ALL, "");
```

Changing the setting of *LC_MESSAGES* has no effect on catalogs that have already been opened by calls to *catopen()*.

In order to make use of different locale settings while multiple threads are running, applications should use *uselocale()* in preference to *setlocale()*.

RATIONALE

References to the international environment or locale in the following text relate to the global locale for the process. This can be overridden for individual threads using *uselocale()*.

The ISO C standard defines a collection of functions to support internationalization. One of the most significant aspects of these functions is a facility to set and query the *international environment*. The international environment is a repository of information that affects the behavior of certain functionality, namely:

1. Character handling
2. Collating
3. Date/time formatting
4. Numeric editing
5. Monetary formatting
6. Messaging

The `setlocale()` function provides the application developer with the ability to set all or portions, called *categories*, of the international environment. These categories correspond to the areas of functionality mentioned above. The syntax for `setlocale()` is as follows:

```
char *setlocale(int category, const char *locale);
```

where *category* is the name of one of following categories, namely:

```
LC_COLLATE
LC_CTYPE
LC_MESSAGES
LC_MONETARY
LC_NUMERIC
LC_TIME
```

In addition, a special value called `LC_ALL` directs `setlocale()` to set all categories.

There are two primary uses of `setlocale()`:

1. Querying the international environment to find out what it is set to
2. Setting the international environment, or *locale*, to a specific value

The behavior of `setlocale()` in these two areas is described below. Since it is difficult to describe the behavior in words, examples are used to illustrate the behavior of specific uses.

To query the international environment, `setlocale()` is invoked with a specific category and the null pointer as the locale. The null pointer is a special directive to `setlocale()` that tells it to query rather than set the international environment. The following syntax is used to query the name of the international environment:

```
setlocale({LC_ALL, LC_COLLATE, LC_CTYPE, LC_MESSAGES, LC_MONETARY, \
          LC_NUMERIC, LC_TIME}, (char *) NULL);
```

The `setlocale()` function shall return the string corresponding to the current international environment. This value may be used by a subsequent call to `setlocale()` to reset the international environment to this value. However, it should be noted that the return value from `setlocale()` may be a pointer to a static area within the function and is not guaranteed to remain unchanged (that is, it may be modified by a subsequent call to `setlocale()`). Therefore, if the purpose of calling `setlocale()` is to save the value of the current international environment so it can be changed and reset later, the return value should be copied to an array of `char` in the calling program.

There are three ways to set the international environment with `setlocale()`:

`setlocale(category, string)`

This usage sets a specific *category* in the international environment to a specific value corresponding to the value of the *string*. A specific example is provided below:

```
60671     setlocale(LC_ALL, "fr_FR.ISO-8859-1");
```

60672 In this example, all categories of the international environment are set to the locale
60673 corresponding to the string "fr_FR.ISO-8859-1", or to the French language as spoken in
60674 France using the ISO/IEC 8859-1:1998 standard codeset.

60675 If the string does not correspond to a valid locale, *setlocale()* shall return a null pointer and
60676 the international environment is not changed. Otherwise, *setlocale()* shall return the name of
60677 the locale just set.

```
60678     setlocale(category, "C")
```

60679 The ISO C standard states that one locale must exist on all conforming implementations.
60680 The name of the locale is C and corresponds to a minimal international environment needed
60681 to support the C programming language.

```
60682     setlocale(category, "")
```

60683 This sets a specific category to an implementation-defined default. This corresponds to the
60684 value of the environment variables.

60685 FUTURE DIRECTIONS

60686 None.

60687 SEE ALSO

60688 *catopen()*, *exec*, *fprintf()*, *fscanf()*, *isalnum()*, *isalpha()*, *isblank()*, *isctrl()*, *isdigit()*, *isgraph()*,
60689 *islower()*, *isprint()*, *ispunct()*, *isspace()*, *isupper()*, *iswalnum()*, *iswalpha()*, *iswblank()*, *iswcntrl()*,
60690 *iswctype()*, *iswdigit()*, *iswgraph()*, *iswlower()*, *iswprint()*, *iswpunct()*, *iswspace()*, *iswupper()*,
60691 *iswxdigit()*, *isxdigit()*, *localeconv()*, *mblen()*, *mbstowcs()*, *mbtowc()*, *nl_langinfo()*, *perror()*,
60692 *psiginfo()*, *setlocale()*, *strcoll()*, *strerror()*, *strfmon()*, *strsignal()*, *strtod()*, *strxfrm()*, *tolower()*,
60693 *toupper()*, *towlower()*, *towupper()*, *uselocale()*, *wcscoll()*, *wcstod()*, *wcstombs()*, *wcsxfrm()*, *wctomb()*

60694 XBD [Chapter 7](#) (on page 135), [Chapter 8](#) (on page 173), [<langinfo.h>](#), [<locale.h>](#)

60695 CHANGE HISTORY

60696 First released in Issue 3.

60697 Issue 5

60698 The DESCRIPTION is updated for alignment with the POSIX Threads Extension.

60699 Issue 6

60700 Extensions beyond the ISO C standard are marked.

60701 The normative text is updated to avoid use of the term “must” for application requirements.

60702 IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/124 is applied, updating the
60703 DESCRIPTION to clarify the behavior of:

```
60704     setlocale(LC_ALL, "");
```

60705 Issue 7

60706 Functionality relating to the Threads option is moved to the Base.

60707 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0553 [302], XSH/TC1-2008/0554 [303], +
60708 XSH/TC1-2008/0555 [302], XSH/TC1-2008/0556 [302], XSH/TC1-2008/0557 [302], +
60709 XSH/TC1-2008/0558 [302], XSH/TC1-2008/0559 [302], XSH/TC1-2008/0560 [288], +
60710 XSH/TC1-2008/0561 [302], XSH/TC1-2008/0562 [302], XSH/TC1-2008/0563 [302], +
60711 XSH/TC1-2008/0564 [302], XSH/TC1-2008/0565 [302], XSH/TC1-2008/0566 [302], +
60712 XSH/TC1-2008/0567 [288], XSH/TC1-2008/0568 [288], and XSH/TC1-2008/0569 [303] are +
60713 applied.

60714 **NAME**

60715 setlogmask — set the log priority mask

60716 **SYNOPSIS**60717 XSI `#include <syslog.h>`60718 `int setlogmask(int maskpri);`60719 **DESCRIPTION**60720 Refer to *closelog()*.

60721 **NAME**

60722 setnetent — network database function

60723 **SYNOPSIS**

60724 #include <netdb.h>

60725 void setnetent(int *stayopen*);

60726 **DESCRIPTION**

60727 Refer to *endnetent()*.

60728 **NAME**

60729 setpgid — set process group ID for job control

60730 **SYNOPSIS**

60731 #include <unistd.h>

60732 int setpgid(pid_t pid, pid_t pgid);

60733 **DESCRIPTION**60734 The *setpgid()* function shall either join an existing process group or create a new process group within the session of the calling process.

60736 The process group ID of a session leader shall not change.

60737 Upon successful completion, the process group ID of the process with a process ID that matches *pid* shall be set to *pgid*.60739 As a special case, if *pid* is 0, the process ID of the calling process shall be used. Also, if *pgid* is 0, the process ID of the indicated process shall be used.60741 **RETURN VALUE**60742 Upon successful completion, *setpgid()* shall return 0; otherwise, -1 shall be returned and *errno* shall be set to indicate the error.60744 **ERRORS**60745 The *setpgid()* function shall fail if:60746 [EACCES] The value of the *pid* argument matches the process ID of a child process of the calling process and the child process has successfully executed one of the *exec* functions.60749 [EINVAL] The value of the *pgid* argument is less than 0, or is not a value supported by the implementation.60751 [EPERM] The process indicated by the *pid* argument is a session leader.60752 [EPERM] The value of the *pid* argument matches the process ID of a child process of the calling process and the child process is not in the same session as the calling process.60755 [EPERM] The value of the *pgid* argument is valid but does not match the process ID of the process indicated by the *pid* argument and there is no process with a process group ID that matches the value of the *pgid* argument in the same session as the calling process.60759 [ESRCH] The value of the *pid* argument does not match the process ID of the calling process or of a child process of the calling process.60761 **EXAMPLES**

60762 None.

60763 **APPLICATION USAGE**

60764 None.

60765 **RATIONALE**60766 The *setpgid()* function shall group processes together for the purpose of signaling, placement in foreground or background, and other job control actions.60768 The *setpgid()* function is similar to the *setpgrp()* function of 4.2 BSD, except that 4.2 BSD allowed the specified new process group to assume any value. This presents certain security problems and is more flexible than necessary to support job control.

To provide tighter security, *setpgid()* only allows the calling process to join a process group already in use inside its session or create a new process group whose process group ID was equal to its process ID.

When a job control shell spawns a new job, the processes in the job must be placed into a new process group via *setpgid()*. There are two timing constraints involved in this action:

1. The new process must be placed in the new process group before the appropriate program is launched via one of the *exec* functions.
2. The new process must be placed in the new process group before the shell can correctly send signals to the new process group.

To address these constraints, the following actions are performed. The new processes call *setpgid()* to alter their own process groups after *fork()* but before *exec*. This satisfies the first constraint. Under 4.3 BSD, the second constraint is satisfied by the synchronization property of *vfork()*; that is, the shell is suspended until the child has completed the *exec*, thus ensuring that the child has completed the *setpgid()*. A new version of *fork()* with this same synchronization property was considered, but it was decided instead to merely allow the parent shell process to adjust the process group of its child processes via *setpgid()*. Both timing constraints are now satisfied by having both the parent shell and the child attempt to adjust the process group of the child process; it does not matter which succeeds first.

Since it would be confusing to an application to have its process group change after it began executing (that is, after *exec*), and because the child process would already have adjusted its process group before this, the [EACCES] error was added to disallow this.

One non-obvious use of *setpgid()* is to allow a job control shell to return itself to its original process group (the one in effect when the job control shell was executed). A job control shell does this before returning control back to its parent when it is terminating or suspending itself as a way of restoring its job control “state” back to what its parent would expect. (Note that the original process group of the job control shell typically matches the process group of its parent, but this is not necessarily always the case.)

FUTURE DIRECTIONS

None.

SEE ALSO

exec, *getpgrp()*, *setsid()*, *tcsetpgrp()*

XBD [`<sys/types.h>`](#), [`<unistd.h>`](#)

CHANGE HISTORY

First released in Issue 3. Included for alignment with the POSIX.1-1988 standard.

Issue 6

In the SYNOPSIS, the optional include of the [`<sys/types.h>`](#) header is removed.

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The requirement to include [`<sys/types.h>`](#) has been removed. Although [`<sys/types.h>`](#) was required for conforming implementations of previous POSIX specifications, it was not required for UNIX applications.
- The *setpgid()* function is mandatory since `_POSIX_JOB_CONTROL` is required to be defined in this version. This is a FIPS requirement.

60814
60815
60816
60817

IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/56 is applied, changing the wording in the DESCRIPTION from “the process group ID of the indicated process shall be used” to “the process ID of the indicated process shall be used”. This change reverts the wording to as in the ISO POSIX-1: 1996 standard; it appeared to be an unintentional change.

60818 **NAME**

60819 setpgrp — set the process group ID

60820 **SYNOPSIS**

```
60821 OB XSI  #include <unistd.h>
60822         pid_t setpgrp(void);
```

60823 **DESCRIPTION**

60824 If the calling process is not already a session leader, *setpgrp()* sets the process group ID of the
60825 calling process to the process ID of the calling process. If *setpgrp()* creates a new session, then the
60826 new session has no controlling terminal.

60827 The *setpgrp()* function has no effect when the calling process is a session leader.

60828 **RETURN VALUE**

60829 Upon completion, *setpgrp()* shall return the process group ID.

60830 **ERRORS**

60831 No errors are defined.

60832 **EXAMPLES**

60833 None.

60834 **APPLICATION USAGE**

60835 It is unspecified whether this function behaves as *setpgid(0,0)* or *setsid()* unless the process is
60836 already a session leader. Therefore, applications are encouraged to use *setpgid()* or *setsid()* as
60837 appropriate.

60838 **RATIONALE**

60839 None.

60840 **FUTURE DIRECTIONS**

60841 The *setpgrp()* function may be removed in a future version.

60842 **SEE ALSO**

60843 *exec*, *fork()*, *getpid()*, *getsid()*, *kill()*, *setpgid()*, *setsid()*

60844 XBD [<unistd.h>](#)

60845 **CHANGE HISTORY**

60846 First released in Issue 4, Version 2.

60847 **Issue 5**

60848 Moved from X/OPEN UNIX extension to BASE.

60849 **Issue 7**

60850 The *setpgrp()* function is marked obsolescent.

60851 **NAME**

60852 setpriority — set the nice value

60853 **SYNOPSIS**

```
60854 XSI      #include <sys/resource.h>
60855          int setpriority(int which, id_t who, int nice);
```

60856 **DESCRIPTION**60857 Refer to *getpriority()*.

60858 **NAME**

60859 setprotoent — network protocol database functions

60860 **SYNOPSIS**

60861 #include <netdb.h>

60862 void setprotoent(int *stayopen*);

60863 **DESCRIPTION**

60864 Refer to *endprotoent()*.

60865 **NAME**

60866 setpwent — user database function

60867 **SYNOPSIS**

60868 XSI #include <pwd.h>

60869 void setpwent(void);

60870 **DESCRIPTION**60871 Refer to *endpwent()*.

NAME

setregid — set real and effective group IDs

SYNOPSIS

```
XSI      #include <unistd.h>
int setregid(gid_t rgid, gid_t egid);
```

DESCRIPTION

The *setregid()* function shall set the real and effective group IDs of the calling process.

If *rgid* is -1 , the real group ID shall not be changed; if *egid* is -1 , the effective group ID shall not be changed.

The real and effective group IDs may be set to different values in the same call.

Only a process with appropriate privileges can set the real group ID and the effective group ID to any valid value.

A non-privileged process can set either the real group ID to the saved set-group-ID from one of the *exec* family of functions, or the effective group ID to the saved set-group-ID or the real group ID.

If the real group ID is being set (*rgid* is not -1), or the effective group ID is being set to a value not equal to the real group ID, then the saved set-group-ID of the current process shall be set equal to the new effective group ID.

Any supplementary group IDs of the calling process remain unchanged.

RETURN VALUE

Upon successful completion, 0 shall be returned. Otherwise, -1 shall be returned and *errno* set to indicate the error, and neither of the group IDs are changed.

ERRORS

The *setregid()* function shall fail if:

[EINVAL]	The value of the <i>rgid</i> or <i>egid</i> argument is invalid or out-of-range.
[EPERM]	The process does not have appropriate privileges and a change other than changing the real group ID to the saved set-group-ID, or changing the effective group ID to the real group ID or the saved set-group-ID, was requested.

EXAMPLES

None.

APPLICATION USAGE

If a non-privileged set-group-ID process sets its effective group ID to its real group ID, it can only set its effective group ID back to the previous value if *rgid* was -1 in the *setregid()* call, since the saved-group-ID is not changed in that case. If *rgid* was equal to the real group ID in the *setregid()* call, then the saved set-group-ID will also have been changed to the real user ID.

RATIONALE

Earlier versions of this standard did not specify whether the saved set-group-ID was affected by *setregid()* calls. This version specifies common existing practice that constitutes an important security feature. The ability to set both the effective group ID and saved set-group-ID to be the same as the real group ID means that any security weakness in code that is executed after that point cannot result in malicious code being executed with the previous effective group ID. Privileged applications could already do this using just *setgid()*, but for non-privileged

60915 applications the only standard method available is to use this feature of *setregid()*.

60916 **FUTURE DIRECTIONS**

60917 None.

60918 **SEE ALSO**

60919 *exec*, *getegid()*, *geteuid()*, *getgid()*, *getuid()*, *setegid()*, *seteuid()*, *setgid()*, *setreuid()*, *setuid()*

60920 XBD <**unistd.h**>

60921 **CHANGE HISTORY**

60922 First released in Issue 4, Version 2.

60923 **Issue 5**

60924 Moved from X/OPEN UNIX extension to BASE.

60925 The DESCRIPTION is updated to indicate that the saved set-group-ID can be set by any of the
60926 *exec* family of functions, not just *execve()*.

60927 **Issue 7**

60928 SD5-XSH-ERN-177 is applied, adding the ability to set both the effective group ID and saved set-
60929 group-ID to be the same as the real group ID.

NAME

setreuid — set real and effective user IDs

SYNOPSIS

```
XSI      #include <unistd.h>
int setreuid(uid_t ruid, uid_t euid);
```

DESCRIPTION

The *setreuid()* function shall set the real and effective user IDs of the current process to the values specified by the *ruid* and *euid* arguments. If *ruid* or *euid* is *-1*, the corresponding effective or real user ID of the current process shall be left unchanged.

A process with appropriate privileges can set either ID to any value. An unprivileged process can only set the effective user ID if the *euid* argument is equal to either the real, effective, or saved user ID of the process.

If the real user ID is being set (*ruid* is not *-1*), or the effective user ID is being set to a value not equal to the real user ID, then the saved set-user-ID of the current process shall be set equal to the new effective user ID.

It is unspecified whether a process without appropriate privileges is permitted to change the real user ID to match the current effective user ID or saved set-user-ID of the process.

RETURN VALUE

Upon successful completion, 0 shall be returned. Otherwise, *-1* shall be returned and *errno* set to indicate the error.

ERRORS

The *setreuid()* function shall fail if:

[EINVAL]	The value of the <i>ruid</i> or <i>euid</i> argument is invalid or out-of-range.
[EPERM]	The current process does not have appropriate privileges, and either an attempt was made to change the effective user ID to a value other than the real user ID or the saved set-user-ID or an attempt was made to change the real user ID to a value not permitted by the implementation.

EXAMPLES**Setting the Effective User ID to the Real User ID**

The following example sets the effective user ID of the calling process to the real user ID, so that files created later will be owned by the current user. It also sets the saved set-user-ID to the real user ID, so any future attempt to set the effective user ID back to its previous value will fail.

```
#include <unistd.h>
#include <sys/types.h>
...
setreuid(getuid(), getuid());
...
```

APPLICATION USAGE

None.

RATIONALE

Earlier versions of this standard did not specify whether the saved set-user-ID was affected by *setreuid()* calls. This version specifies common existing practice that constitutes an important security feature. The ability to set both the effective user ID and saved set-user-ID to be the same as the real user ID means that any security weakness in code that is executed after that point cannot result in malicious code being executed with the previous effective user ID. Privileged applications could already do this using just *setuid()*, but for non-privileged applications the only standard method available is to use this feature of *setreuid()*.

FUTURE DIRECTIONS

None.

SEE ALSO

getegid(), *geteuid()*, *getgid()*, *getuid()*, *setegid()*, *seteuid()*, *setgid()*, *setregid()*, *setuid()*

XBD <**unistd.h**>

CHANGE HISTORY

First released in Issue 4, Version 2.

Issue 5

Moved from X/OPEN UNIX extension to BASE.

Issue 7

SD5-XSH-ERN-177 is applied, adding the ability to set both the effective user ID and the saved set-user-ID to be the same as the real user ID.

60989 NAME

60990 setrlimit — control maximum resource consumption

60991 SYNOPSIS

60992 XSI `#include <sys/resource.h>`

60993 `int setrlimit(int resource, const struct rlimit *rlp);`

60994 DESCRIPTION

60995 Refer to *getrlimit()*.

60996 **NAME**

60997 setservent — network services database functions

60998 **SYNOPSIS**

60999 #include <netdb.h>

61000 void setservent(int *stayopen*);61001 **DESCRIPTION**61002 Refer to *endservent()*.

61003 **NAME**

61004 setsid — create session and set process group ID

61005 **SYNOPSIS**

61006 #include <unistd.h>

61007 pid_t setsid(void);

61008 **DESCRIPTION**

61009 The *setsid()* function shall create a new session, if the calling process is not a process group
 61010 leader. Upon return the calling process shall be the session leader of this new session, shall be
 61011 the process group leader of a new process group, and shall have no controlling terminal. The
 61012 process group ID of the calling process shall be set equal to the process ID of the calling process.
 61013 The calling process shall be the only process in the new process group and the only process in
 61014 the new session.

61015 **RETURN VALUE**

61016 Upon successful completion, *setsid()* shall return the value of the new process group ID of the
 61017 calling process. Otherwise, it shall return -1 and set *errno* to indicate the error.

61018 **ERRORS**61019 The *setsid()* function shall fail if:

61020 [EPERM] The calling process is already a process group leader, or the process group ID
 61021 of a process other than the calling process matches the process ID of the
 61022 calling process.

61023 **EXAMPLES**

61024 None.

61025 **APPLICATION USAGE**

61026 None.

61027 **RATIONALE**

61028 The *setsid()* function is similar to the *setpgid()* function of System V. System V, without job
 61029 control, groups processes into process groups and creates new process groups via *setpgid()*; only
 61030 one process group may be part of a login session.

61031 Job control allows multiple process groups within a login session. In order to limit job control
 61032 actions so that they can only affect processes in the same login session, this volume of
 61033 POSIX.1-2008 adds the concept of a session that is created via *setsid()*. The *setsid()* function also
 61034 creates the initial process group contained in the session. Additional process groups can be
 61035 created via the *setpgid()* function. A System V process group would correspond to a POSIX
 61036 System Interfaces session containing a single POSIX process group. Note that this function
 61037 requires that the calling process not be a process group leader. The usual way to ensure this is
 61038 true is to create a new process with *fork()* and have it call *setsid()*. The *fork()* function
 61039 guarantees that the process ID of the new process does not match any existing process group ID.

61040 **FUTURE DIRECTIONS**

61041 None.

61042 **SEE ALSO**61043 *getsid()*, *setpgid()*, *setpgid()*

61044 XBD <sys/types.h>, <unistd.h>

CHANGE HISTORY

61045 First released in Issue 3. Included for alignment with the POSIX.1-1988 standard.

Issue 6

61047 In the SYNOPSIS, the optional include of the `<sys/types.h>` header is removed.

61049 The following new requirements on POSIX implementations derive from alignment with the
61050 Single UNIX Specification:

- 61051 • The requirement to include `<sys/types.h>` has been removed. Although `<sys/types.h>` was
61052 required for conforming implementations of previous POSIX specifications, it was not
61053 required for UNIX applications.

Issue 7

61054 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0570 [421] is applied.

+

NAME

setsockopt — set the socket options

SYNOPSIS

```
#include <sys/socket.h>

int setsockopt(int socket, int level, int option_name,
               const void *option_value, socklen_t option_len);
```

DESCRIPTION

The *setsockopt()* function shall set the option specified by the *option_name* argument, at the protocol level specified by the *level* argument, to the value pointed to by the *option_value* argument for the socket associated with the file descriptor specified by the *socket* argument.

The *level* argument specifies the protocol level at which the option resides. To set options at the socket level, specify the *level* argument as SOL_SOCKET. To set options at other levels, supply the appropriate *level* identifier for the protocol controlling the option. For example, to indicate that an option is interpreted by the TCP (Transport Control Protocol), set *level* to IPPROTO_TCP as defined in the **<netinet/in.h>** header.

The *option_name* argument specifies a single option to set. It can be one of the socket-level options defined in **<sys/socket.h>** and described in [Section 2.10.16](#) (on page 527). If *option_name* is equal to SO_RCVTIMEO or SO_SNDTIMEO and the implementation supports setting the option, it is unspecified whether the **struct timeval** pointed to by *option_value* is stored as provided by this function or is rounded up to align with the resolution of the clock being used. If *setsockopt()* is called with *option_name* equal to SO_ACCEPTCONN, SO_ERROR, or SO_TYPE, the behavior is unspecified.

RETURN VALUE

Upon successful completion, *setsockopt()* shall return 0. Otherwise, -1 shall be returned and *errno* set to indicate the error.

ERRORS

The *setsockopt()* function shall fail if:

- [EBADF] The *socket* argument is not a valid file descriptor.
- [EDOM] The send and receive timeout values are too big to fit into the timeout fields in the socket structure.
- [EINVAL] The specified option is invalid at the specified socket level or the socket has been shut down.
- [EISCONN] The socket is already connected, and a specified option cannot be set while the socket is connected.
- [ENOPROTOOPT] The option is not supported by the protocol.
- [ENOTSOCK] The *socket* argument does not refer to a socket.

The *setsockopt()* function may fail if:

- [ENOMEM] There was insufficient memory available for the operation to complete.
- [ENOBUFS] Insufficient resources are available in the system to complete the call.

61096 EXAMPLES

61097 None.

61098 APPLICATION USAGE

61099 The *setsockopt()* function provides an application program with the means to control socket
61100 behavior. An application program can use *setsockopt()* to allocate buffer space, control timeouts,
61101 or permit socket data broadcasts. The **<sys/socket.h>** header defines the socket-level options
61102 available to *setsockopt()*.

61103 Options may exist at multiple protocol levels. The SO_ options are always present at the
61104 uppermost socket level.

61105 RATIONALE

61106 None.

61107 FUTURE DIRECTIONS

61108 None.

61109 SEE ALSO

61110 [Section 2.10](#) (on page 522), *bind()*, *endprotoent()*, *getsockopt()*, *socket()*

61111 XBD **<netinet/in.h>**, **<sys/socket.h>**

61112 CHANGE HISTORY

61113 First released in Issue 6. Derived from the XNS, Issue 5.2 specification.

61114 IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/125 is applied, updating the SO_LINGER
61115 option in the DESCRIPTION to refer to the calling thread rather than the process.

61116 Issue 7

61117 Austin Group Interpretation 1003.1-2001 #158 is applied, removing text relating to socket options
61118 that is now in [Section 2.10.16](#) (on page 527).

61119 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0571 [369] is applied.

+

61120 NAME

61121 setstate — switch pseudo-random number generator state arrays

61122 SYNOPSIS

```
61123 XSI       #include <stdlib.h>  
61124       char *setstate(char *state);
```

61125 DESCRIPTION

61126 Refer to *initstate()*.

61127 **NAME**

61128 setuid — set user ID

61129 **SYNOPSIS**

61130 #include <unistd.h>

61131 int setuid(uid_t uid);

61132 **DESCRIPTION**

61133 If the process has appropriate privileges, *setuid()* shall set the real user ID, effective user ID, and
 61134 the saved set-user-ID of the calling process to *uid*.

61135 If the process does not have appropriate privileges, but *uid* is equal to the real user ID or the
 61136 saved set-user-ID, *setuid()* shall set the effective user ID to *uid*; the real user ID and saved set-
 61137 user-ID shall remain unchanged.

61138 The *setuid()* function shall not affect the supplementary group list in any way.

61139 **RETURN VALUE**

61140 Upon successful completion, 0 shall be returned. Otherwise, -1 shall be returned and *errno* set to
 61141 indicate the error.

61142 **ERRORS**

61143 The *setuid()* function shall fail, return -1, and set *errno* to the corresponding value if one or more
 61144 of the following are true:

61145 [EINVAL] The value of the *uid* argument is invalid and not supported by the
 61146 implementation.

61147 [EPERM] The process does not have appropriate privileges and *uid* does not match the
 61148 real user ID or the saved set-user-ID.

61149 **EXAMPLES**

61150 None.

61151 **APPLICATION USAGE**

61152 None.

61153 **RATIONALE**

61154 The various behaviors of the *setuid()* and *setgid()* functions when called by non-privileged
 61155 processes reflect the behavior of different historical implementations. For portability, it is
 61156 recommended that new non-privileged applications use the *seteuid()* and *setegid()* functions
 61157 instead.

61158 The saved set-user-ID capability allows a program to regain the effective user ID established at
 61159 the last *exec* call. Similarly, the saved set-group-ID capability allows a program to regain the
 61160 effective group ID established at the last *exec* call. These capabilities are derived from System V.
 61161 Without them, a program might have to run as superuser in order to perform the same
 61162 functions, because superuser can write on the user's files. This is a problem because such a
 61163 program can write on any user's files, and so must be carefully written to emulate the
 61164 permissions of the calling process properly. In System V, these capabilities have traditionally
 61165 been implemented only via the *setuid()* and *setgid()* functions for non-privileged processes. The
 61166 fact that the behavior of those functions was different for privileged processes made them
 61167 difficult to use. The POSIX.1-1990 standard defined the *setuid()* function to behave differently
 61168 for privileged and unprivileged users. When the caller had appropriate privileges, the function
 61169 set the real user ID, effective user ID, and saved set-user ID of the calling process on
 61170 implementations that supported it. When the caller did not have appropriate privileges, the
 61171 function set only the effective user ID, subject to permission checks. The former use is generally
 61172 needed for utilities like *login* and *su*, which are not conforming applications and thus outside the

scope of POSIX.1-2008. These utilities wish to change the user ID irrevocably to a new value, generally that of an unprivileged user. The latter use is needed for conforming applications that are installed with the set-user-ID bit and need to perform operations using the real user ID.

POSIX.1-2008 augments the latter functionality with a mandatory feature named `_POSIX_SAVED_IDS`. This feature permits a set-user-ID application to switch its effective user ID back and forth between the values of its *exec*-time real user ID and effective user ID. Unfortunately, the POSIX.1-1990 standard did not permit a conforming application using this feature to work properly when it happened to be executed with (implementation-defined) appropriate privileges. Furthermore, the application did not even have a means to tell whether it had this privilege. Since the saved set-user-ID feature is quite desirable for applications, as evidenced by the fact that NIST required it in FIPS 151-2, it has been mandated by POSIX.1-2008. However, there are implementors who have been reluctant to support it given the limitation described above.

The 4.3BSD system handles the problem by supporting separate functions: *setuid()* (which always sets both the real and effective user IDs, like *setuid()* in POSIX.1-2008 for privileged users), and *seteuid()* (which always sets just the effective user ID, like *setuid()* in POSIX.1-2008 for non-privileged users). This separation of functionality into distinct functions seems desirable. 4.3BSD does not support the saved set-user-ID feature. It supports similar functionality of switching the effective user ID back and forth via *setreuid()*, which permits reversing the real and effective user IDs. This model seems less desirable than the saved set-user-ID because the real user ID changes as a side-effect. The current 4.4BSD includes saved effective IDs and uses them for *seteuid()* and *setegid()* as described above. The *setreuid()* and *setregid()* functions will be deprecated or removed.

The solution here is:

- Require that all implementations support the functionality of the saved set-user-ID, which is set by the *exec* functions and by privileged calls to *setuid()*.
- Add the *seteuid()* and *setegid()* functions as portable alternatives to *setuid()* and *setgid()* for non-privileged and privileged processes.

Historical systems have provided two mechanisms for a set-user-ID process to change its effective user ID to be the same as its real user ID in such a way that it could return to the original effective user ID: the use of the *setuid()* function in the presence of a saved set-user-ID, or the use of the BSD *setreuid()* function, which was able to swap the real and effective user IDs. The changes included in POSIX.1-2008 provide a new mechanism using *seteuid()* in conjunction with a saved set-user-ID. Thus, all implementations with the new *seteuid()* mechanism will have a saved set-user-ID for each process, and most of the behavior controlled by `_POSIX_SAVED_IDS` has been changed to agree with the case where the option was defined. The *kill()* function is an exception. Implementors of the new *seteuid()* mechanism will generally be required to maintain compatibility with the older mechanisms previously supported by their systems. However, compatibility with this use of *setreuid()* and with the `_POSIX_SAVED_IDS` behavior of *kill()* is unfortunately complicated. If an implementation with a saved set-user-ID allows a process to use *setreuid()* to swap its real and effective user IDs, but were to leave the saved set-user-ID unmodified, the process would then have an effective user ID equal to the original real user ID, and both real and saved set-user-ID would be equal to the original effective user ID. In that state, the real user would be unable to kill the process, even though the effective user ID of the process matches that of the real user, if the *kill()* behavior of `_POSIX_SAVED_IDS` was used. This is obviously not acceptable. The alternative choice, which is used in at least one implementation, is to change the saved set-user-ID to the effective user ID during most calls to *setreuid()*. The standard developers considered that alternative to be less correct than the retention of the old behavior of *kill()* in such systems. Current conforming applications shall

61222 accommodate either behavior from *kill()*, and there appears to be no strong reason for *kill()* to
61223 check the saved set-user-ID rather than the effective user ID.

61224 FUTURE DIRECTIONS

61225 None.

61226 SEE ALSO

61227 *exec*, *getegid()*, *geteuid()*, *getgid()*, *getuid()*, *setegid()*, *seteuid()*, *setgid()*, *setregid()*, *setreuid()*

61228 XBD **<sys/types.h>**, **<unistd.h>**

61229 CHANGE HISTORY

61230 First released in Issue 1. Derived from Issue 1 of the SVID.

61231 Issue 6

61232 In the SYNOPSIS, the optional include of the **<sys/types.h>** header is removed.

61233 The following new requirements on POSIX implementations derive from alignment with the
61234 Single UNIX Specification:

- 61235 • The requirement to include **<sys/types.h>** has been removed. Although **<sys/types.h>** was
61236 required for conforming implementations of previous POSIX specifications, it was not
61237 required for UNIX applications.
- 61238 • The functionality associated with `_POSIX_SAVED_IDS` is now mandatory. This is a FIPS
61239 requirement.

61240 The following changes were made to align with the IEEE P1003.1a draft standard:

- 61241 • The effects of *setuid()* in processes without appropriate privileges are changed.
- 61242 • A requirement that the supplementary group list is not affected is added.

61243 **NAME**

61244 setutxent — reset the user accounting database to the first entry

61245 **SYNOPSIS**

```
61246 XSI      #include <utmpx.h>
61247         void setutxent(void);
```

61248 **DESCRIPTION**61249 Refer to *endutxent()*.

61250 **NAME**

61251 setvbuf — assign buffering to a stream

61252 **SYNOPSIS**

61253 #include <stdio.h>

```
61254       int setvbuf(FILE *restrict stream, char *restrict buf, int type,
61255                   size_t size);
```

61256 **DESCRIPTION**

61257 CX The functionality described on this reference page is aligned with the ISO C standard. Any
 61258 conflict between the requirements described here and the ISO C standard is unintentional. This
 61259 volume of POSIX.1-2008 defers to the ISO C standard.

61260 The *setvbuf()* function may be used after the stream pointed to by *stream* is associated with an
 61261 open file but before any other operation (other than an unsuccessful call to *setvbuf()*) is
 61262 performed on the stream. The argument *type* determines how *stream* shall be buffered, as
 61263 follows:

- 61264 • {_IOFBF} shall cause input/output to be fully buffered.
- 61265 • {_IOLBF} shall cause input/output to be line buffered.
- 61266 • {_IONBF} shall cause input/output to be unbuffered.

61267 If *buf* is not a null pointer, the array it points to may be used instead of a buffer allocated by
 61268 *setvbuf()* and the argument *size* specifies the size of the array; otherwise, *size* may determine the
 61269 size of a buffer allocated by the *setvbuf()* function. The contents of the array at any time are
 61270 unspecified.

61271 For information about streams, see [Section 2.5](#) (on page 495).

61272 **RETURN VALUE**

61273 Upon successful completion, *setvbuf()* shall return 0. Otherwise, it shall return a non-zero value
 61274 CX if an invalid value is given for *type* or if the request cannot be honored, and may set *errno* to
 61275 indicate the error.

61276 **ERRORS**

61277 The *setvbuf()* function may fail if:

61278 CX [EBADF] The file descriptor underlying *stream* is not valid.

61279 **EXAMPLES**

61280 None.

61281 **APPLICATION USAGE**

61282 A common source of error is allocating buffer space as an “automatic” variable in a code block,
 61283 and then failing to close the stream in the same block.

61284 With *setvbuf()*, allocating a buffer of *size* bytes does not necessarily imply that all of *size* bytes are
 61285 used for the buffer area.

61286 Applications should note that many implementations only provide line buffering on input from
 61287 terminal devices.

61288 **RATIONALE**

61289 None.

61290 **FUTURE DIRECTIONS**

61291 None.

61292 **SEE ALSO**61293 [Section 2.5](#) (on page 495), [fopen\(\)](#), [setbuf\(\)](#)61294 XBD [<stdio.h>](#)61295 **CHANGE HISTORY**

61296 First released in Issue 1. Derived from Issue 1 of the SVID.

61297 **Issue 6**

61298 Extensions beyond the ISO C standard are marked.

61299 The *setvbuf()* prototype is updated for alignment with the ISO/IEC 9899:1999 standard.

61300 **NAME**61301 shm_open — open a shared memory object (**REALTIME**)61302 **SYNOPSIS**

```
61303 SHM    #include <sys/mman.h>
61304        int shm_open(const char *name, int oflag, mode_t mode);
```

61305 **DESCRIPTION**

61306 The *shm_open()* function shall establish a connection between a shared memory object and a file
 61307 descriptor. It shall create an open file description that refers to the shared memory object and a
 61308 file descriptor that refers to that open file description. The file descriptor is used by other
 61309 functions to refer to that shared memory object. The *name* argument points to a string naming a
 61310 shared memory object. It is unspecified whether the name appears in the file system and is
 61311 visible to other functions that take pathnames as arguments. The *name* argument conforms to the
 61312 construction rules for a pathname, except that the interpretation of <slash> characters other than
 61313 the leading <slash> character in *name* is implementation-defined, and that the length limits for
 61314 the *name* argument are implementation-defined and need not be the same as the pathname limits
 61315 {PATH_MAX} and {NAME_MAX}. If *name* begins with the <slash> character, then processes
 61316 calling *shm_open()* with the same value of *name* refer to the same shared memory object, as long
 61317 as that name has not been removed. If *name* does not begin with the <slash> character, the effect
 61318 is implementation-defined.

61319 If successful, *shm_open()* shall return a file descriptor for the shared memory object that is the
 61320 lowest numbered file descriptor not currently open for that process. The open file description is
 61321 new, and therefore the file descriptor does not share it with any other processes. It is unspecified
 61322 whether the file offset is set. The FD_CLOEXEC file descriptor flag associated with the new file
 61323 descriptor is set.

61324 The file status flags and file access modes of the open file description are according to the value
 61325 of *oflag*. The *oflag* argument is the bitwise-inclusive OR of the following flags defined in the
 61326 <fcntl.h> header. Applications specify exactly one of the first two values (access modes) below
 61327 in the value of *oflag*:

61328 O_RDONLY Open for read access only.
 61329 O_RDWR Open for read or write access.

61330 Any combination of the remaining flags may be specified in the value of *oflag*:

61331 O_CREAT If the shared memory object exists, this flag has no effect, except as noted
 61332 under O_EXCL below. Otherwise, the shared memory object is created. The
 61333 user ID of the shared memory object shall be set to the effective user ID of the
 61334 process. The group ID of the shared memory object shall be set to the effective
 61335 group ID of the process; however, if the *name* argument is visible in the file
 61336 system, the group ID may be set to the group ID of the containing directory.
 61337 The permission bits of the shared memory object shall be set to the value of
 61338 the *mode* argument except those set in the file mode creation mask of the
 61339 process. When bits in *mode* other than the file permission bits are set, the effect
 61340 is unspecified. The *mode* argument does not affect whether the shared memory
 61341 object is opened for reading, for writing, or for both. The shared memory
 61342 object has a size of zero.

61343 O_EXCL If O_EXCL and O_CREAT are set, *shm_open()* fails if the shared memory
 61344 object exists. The check for the existence of the shared memory object and the
 61345 creation of the object if it does not exist is atomic with respect to other

61346		processes executing <i>shm_open()</i> naming the same shared memory object with
61347		O_EXCL and O_CREAT set. If O_EXCL is set and O_CREAT is not set, the
61348		result is undefined.
61349	O_TRUNC	If the shared memory object exists, and it is successfully opened O_RDWR, the
61350		object shall be truncated to zero length and the mode and owner shall be
61351		unchanged by this function call. The result of using O_TRUNC with
61352		O_RDONLY is undefined.
61353		When a shared memory object is created, the state of the shared memory object, including all
61354		data associated with the shared memory object, persists until the shared memory object is
61355		unlinked and all other references are gone. It is unspecified whether the name and shared
61356		memory object state remain valid after a system reboot.
61357	RETURN VALUE	
61358		Upon successful completion, the <i>shm_open()</i> function shall return a non-negative integer
61359		representing the lowest numbered unused file descriptor. Otherwise, it shall return -1 and set
61360		<i>errno</i> to indicate the error.
61361	ERRORS	
61362		The <i>shm_open()</i> function shall fail if:
61363	[EACCES]	The shared memory object exists and the permissions specified by <i>oflag</i> are
61364		denied, or the shared memory object does not exist and permission to create
61365		the shared memory object is denied, or O_TRUNC is specified and write
61366		permission is denied.
61367	[EEXIST]	O_CREAT and O_EXCL are set and the named shared memory object already
61368		exists.
61369	[EINTR]	The <i>shm_open()</i> operation was interrupted by a signal.
61370	[EINVAL]	The <i>shm_open()</i> operation is not supported for the given name.
61371	[EMFILE]	All file descriptors available to the process are currently open.
61372	[ENFILE]	Too many shared memory objects are currently open in the system.
61373	[ENOENT]	O_CREAT is not set and the named shared memory object does not exist.
61374	[ENOSPC]	There is insufficient space for the creation of the new shared memory object.
61375		The <i>shm_open()</i> function may fail if:
61376	[ENAMETOOLONG]	
61377		The length of the <i>name</i> argument exceeds {_POSIX_PATH_MAX} on systems
61378	XSI	that do not support the XSI option or exceeds {_XOPEN_PATH_MAX} on XSI
61379		systems, or has a pathname component that is longer than
61380	XSI	{_POSIX_NAME_MAX} on systems that do not support the XSI option or
61381		longer than {_XOPEN_NAME_MAX} on XSI systems.

61382 **EXAMPLES**61383 **Creating and Mapping a Shared Memory Object**

61384 The following code segment demonstrates the use of *shm_open()* to create a shared memory
 61385 object which is then sized using *ftruncate()* before being mapped into the process address space
 61386 using *mmap()*:

```
61387 #include <unistd.h>
61388 #include <sys/mman.h>
61389 ...
61390 #define MAX_LEN 10000
61391 struct region {          /* Defines "structure" of shared memory */
61392     int len;
61393     char buf[MAX_LEN];
61394 };
61395 struct region *rptr;
61396 int fd;
61397
61398 /* Create shared memory object and set its size */
61399 fd = shm_open("/myregion", O_CREAT | O_RDWR, S_IRUSR | S_IWUSR);
61400 if (fd == -1)
61401     /* Handle error */;
61402
61403 if (ftruncate(fd, sizeof(struct region)) == -1)
61404     /* Handle error */;
61405
61406 /* Map shared memory object */
61407 rptr = mmap(NULL, sizeof(struct region),
61408             PROT_READ | PROT_WRITE, MAP_SHARED, fd, 0);
61409 if (rptr == MAP_FAILED)
61410     /* Handle error */;
61411
61412 /* Now we can refer to mapped region using fields of rptr;
61413    for example, rptr->len */
61414 ...
```

61411 **APPLICATION USAGE**

61412 None.

61413 **RATIONALE**

61414 When the Memory Mapped Files option is supported, the normal *open()* call is used to obtain a
 61415 descriptor to a file to be mapped according to existing practice with *mmap()*. When the Shared
 61416 Memory Objects option is supported, the *shm_open()* function shall obtain a descriptor to the
 61417 shared memory object to be mapped.

61418 There is ample precedent for having a file descriptor represent several types of objects. In the
 61419 POSIX.1-1990 standard, a file descriptor can represent a file, a pipe, a FIFO, a tty, or a directory.
 61420 Many implementations simply have an operations vector, which is indexed by the file descriptor
 61421 type and does very different operations. Note that in some cases the file descriptor passed to
 61422 generic operations on file descriptors is returned by *open()* or *creat()* and in some cases returned
 61423 by alternate functions, such as *pipe()*. The latter technique is used by *shm_open()*.

61424 Note that such shared memory objects can actually be implemented as mapped files. In both
 61425 cases, the size can be set after the open using *ftruncate()*. The *shm_open()* function itself does not

create a shared object of a specified size because this would duplicate an extant function that set the size of an object referenced by a file descriptor.

On implementations where memory objects are implemented using the existing file system, the *shm_open()* function may be implemented using a macro that invokes *open()*, and the *shm_unlink()* function may be implemented using a macro that invokes *unlink()*.

For implementations without a permanent file system, the definition of the name of the memory objects is allowed not to survive a system reboot. Note that this allows systems with a permanent file system to implement memory objects as data structures internal to the implementation as well.

On implementations that choose to implement memory objects using memory directly, a *shm_open()* followed by an *ftruncate()* and *close()* can be used to preallocate a shared memory area and to set the size of that preallocation. This may be necessary for systems without virtual memory hardware support in order to ensure that the memory is contiguous.

The set of valid open flags to *shm_open()* was restricted to *O_RDONLY*, *O_RDWR*, *O_CREAT*, and *O_TRUNC* because these could be easily implemented on most memory mapping systems. This volume of POSIX.1-2008 is silent on the results if the implementation cannot supply the requested file access because of implementation-defined reasons, including hardware ones.

The error conditions [EACCES] and [ENOTSUP] are provided to inform the application that the implementation cannot complete a request.

[EACCES] indicates for implementation-defined reasons, probably hardware-related, that the implementation cannot comply with a requested mode because it conflicts with another requested mode. An example might be that an application desires to open a memory object two times, mapping different areas with different access modes. If the implementation cannot map a single area into a process space in two places, which would be required if different access modes were required for the two areas, then the implementation may inform the application at the time of the second open.

[ENOTSUP] indicates for implementation-defined reasons, probably hardware-related, that the implementation cannot comply with a requested mode at all. An example would be that the hardware of the implementation cannot support write-only shared memory areas.

On all implementations, it may be desirable to restrict the location of the memory objects to specific file systems for performance (such as a RAM disk) or implementation-defined reasons (shared memory supported directly only on certain file systems). The *shm_open()* function may be used to enforce these restrictions. There are a number of methods available to the application to determine an appropriate name of the file or the location of an appropriate directory. One way is from the environment via *getenv()*. Another would be from a configuration file.

This volume of POSIX.1-2008 specifies that memory objects have initial contents of zero when created. This is consistent with current behavior for both files and newly allocated memory. For those implementations that use physical memory, it would be possible that such implementations could simply use available memory and give it to the process uninitialized. This, however, is not consistent with standard behavior for the uninitialized data area, the stack, and of course, files. Finally, it is highly desirable to set the allocated memory to zero for security reasons. Thus, initializing memory objects to zero is required.

FUTURE DIRECTIONS

A future version might require the *shm_open()* and *shm_unlink()* functions to have semantics similar to normal file system operations.

SEE ALSO

61471 *close()*, *dup()*, *exec*, *fcntl()*, *mmap()*, *shmat()*, *shmctl()*, *shmdt()*, *shm_unlink()*, *umask()*
61472
61473 XBD <fcntl.h>, <sys/mman.h>

CHANGE HISTORY

61474 First released in Issue 5. Included for alignment with the POSIX Realtime Extension.

Issue 6

61476 The *shm_open()* function is marked as part of the Shared Memory Objects option.

61478 The [ENOSYS] error condition has been removed as stubs need not be provided if an
61479 implementation does not support the Shared Memory Objects option.

61480 IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/126 is applied, adding the example to the
61481 EXAMPLES section.

Issue 7

61482 Austin Group Interpretation 1003.1-2001 #077 is applied, clarifying the *name* argument and
61483 changing [ENAMETOOLONG] from a “shall fail” to a “may fail” error.

61485 Austin Group Interpretation 1003.1-2001 #141 is applied, adding FUTURE DIRECTIONS.

61486 SD5-XSH-ERN-170 is applied, updating the DESCRIPTION to clarify the wording for setting the
61487 user ID and group ID of the shared memory object.

61488 **NAME**61489 shm_unlink — remove a shared memory object (**REALTIME**)61490 **SYNOPSIS**

```
61491 SHM    #include <sys/mman.h>
61492      int shm_unlink(const char *name);
```

61493 **DESCRIPTION**

61494 The *shm_unlink()* function shall remove the name of the shared memory object named by the
 61495 string pointed to by *name*.

61496 If one or more references to the shared memory object exist when the object is unlinked, the
 61497 name shall be removed before *shm_unlink()* returns, but the removal of the memory object
 61498 contents shall be postponed until all open and map references to the shared memory object have
 61499 been removed.

61500 Even if the object continues to exist after the last *shm_unlink()*, reuse of the name shall
 61501 subsequently cause *shm_open()* to behave as if no shared memory object of this name exists (that
 61502 is, *shm_open()* will fail if O_CREAT is not set, or will create a new shared memory object if
 61503 O_CREAT is set).

61504 **RETURN VALUE**

61505 Upon successful completion, a value of zero shall be returned. Otherwise, a value of -1 shall be
 61506 returned and *errno* set to indicate the error. If -1 is returned, the named shared memory object
 61507 shall not be changed by this function call.

61508 **ERRORS**

61509 The *shm_unlink()* function shall fail if:

61510 [EACCES] Permission is denied to unlink the named shared memory object.

61511 [ENOENT] The named shared memory object does not exist.

61512 The *shm_unlink()* function may fail if:

61513 [ENAMETOOLONG]

61514 The length of the *name* argument exceeds {_POSIX_PATH_MAX} on systems
 61515 XSI that do not support the XSI option or exceeds {_XOPEN_PATH_MAX} on XSI
 61516 systems, or has a pathname component that is longer than
 61517 XSI {_POSIX_NAME_MAX} on systems that do not support the XSI option or
 61518 longer than {_XOPEN_NAME_MAX} on XSI systems. A call to *shm_unlink()*
 61519 with a *name* argument that contains the same shared memory object name as
 61520 was previously used in a successful *shm_open()* call shall not give an
 61521 [ENAMETOOLONG] error.

61522 **EXAMPLES**

61523 None.

61524 **APPLICATION USAGE**

61525 Names of memory objects that were allocated with *open()* are deleted with *unlink()* in the usual
 61526 fashion. Names of memory objects that were allocated with *shm_open()* are deleted with
 61527 *shm_unlink()*. Note that the actual memory object is not destroyed until the last close and
 61528 unmap on it have occurred if it was already in use.

61529 RATIONALE

61530 None.

61531 FUTURE DIRECTIONS

61532 A future version might require the *shm_open()* and *shm_unlink()* functions to have semantics
61533 similar to normal file system operations.

61534 SEE ALSO

61535 *close()*, *mmap()*, *munmap()*, *shmat()*, *shmctl()*, *shmdt()*, *shm_open()*

61536 XBD <sys/mman.h>

61537 CHANGE HISTORY

61538 First released in Issue 5. Included for alignment with the POSIX Realtime Extension.

61539 Issue 6

61540 The *shm_unlink()* function is marked as part of the Shared Memory Objects option.

61541 In the DESCRIPTION, text is added to clarify that reusing the same name after a *shm_unlink()*
61542 will not attach to the old shared memory object.

61543 The [ENOSYS] error condition has been removed as stubs need not be provided if an
61544 implementation does not support the Shared Memory Objects option.

61545 Issue 7

61546 Austin Group Interpretation 1003.1-2001 #077 is applied, changing [ENAMETOOLONG] from a
61547 “shall fail” to a “may fail” error.

61548 Austin Group Interpretation 1003.1-2001 #141 is applied, adding FUTURE DIRECTIONS.

61549 **NAME**

61550 shmat — XSI shared memory attach operation

61551 **SYNOPSIS**

```
61552 XSI      #include <sys/shm.h>
61553      void *shmat(int shmid, const void *shmaddr, int shmflg);
```

61554 **DESCRIPTION**

61555 The *shmat()* function operates on XSI shared memory (see XBD [Section 3.342](#), on page 88). It is
 61556 unspecified whether this function interoperates with the realtime interprocess communication
 61557 facilities defined in [Section 2.8](#) (on page 502).

61558 The *shmat()* function attaches the shared memory segment associated with the shared memory
 61559 identifier specified by *shmid* to the address space of the calling process. The segment is attached
 61560 at the address specified by one of the following criteria:

- 61561 • If *shmaddr* is a null pointer, the segment is attached at the first available address as selected
 61562 by the system.
- 61563 • If *shmaddr* is not a null pointer and (*shmflg* &SHM_RND) is non-zero, the segment is
 61564 attached at the address given by (*shmaddr* - ((uintptr_t)*shmaddr* %SHMLBA)). The character
 61565 ' % ' is the C-language remainder operator.
- 61566 • If *shmaddr* is not a null pointer and (*shmflg* &SHM_RND) is 0, the segment is attached at
 61567 the address given by *shmaddr*.
- 61568 • The segment is attached for reading if (*shmflg* &SHM_RDONLY) is non-zero and the
 61569 calling process has read permission; otherwise, if it is 0 and the calling process has read
 61570 and write permission, the segment is attached for reading and writing.

61571 **RETURN VALUE**

61572 Upon successful completion, *shmat()* shall increment the value of *shm_nattch* in the data
 61573 structure associated with the shared memory ID of the attached shared memory segment and
 61574 return the segment's start address. Also, the *shm_atime* timestamp shall be set to the current +
 61575 time, as described in [Section 2.7.1](#) (on page 501).

61576 Otherwise, the shared memory segment shall not be attached, *shmat()* shall return -1, and *errno*
 61577 shall be set to indicate the error.

61578 **ERRORS**

61579 The *shmat()* function shall fail if:

61580 61581	[EACCES]	Operation permission is denied to the calling process; see Section 2.7 (on page 501).
61582 61583 61584 61585 61586	[EINVAL]	The value of <i>shmid</i> is not a valid shared memory identifier, the <i>shmaddr</i> is not a null pointer, and the value of (<i>shmaddr</i> - ((uintptr_t) <i>shmaddr</i> %SHMLBA)) is an illegal address for attaching shared memory; or the <i>shmaddr</i> is not a null pointer, (<i>shmflg</i> &SHM_RND) is 0, and the value of <i>shmaddr</i> is an illegal address for attaching shared memory.
61587 61588	[EMFILE]	The number of shared memory segments attached to the calling process would exceed the system-imposed limit.
61589 61590	[ENOMEM]	The available data space is not large enough to accommodate the shared memory segment.

61591 EXAMPLES

61592 None.

61593 APPLICATION USAGE

61594 The POSIX Realtime Extension defines alternative interfaces for interprocess communication.
61595 Application developers who need to use IPC should design their applications so that modules
61596 using the IPC routines described in [Section 2.7](#) (on page 501) can be easily modified to use the
61597 alternative interfaces.

61598 RATIONALE

61599 None.

61600 FUTURE DIRECTIONS

61601 None.

61602 SEE ALSO

61603 [Section 2.7](#) (on page 501), [Section 2.8](#) (on page 502), *exec*, *exit()*, *fork()*, *shmctl()*, *shmdt()*,
61604 *shmget()*, *shm_open()*, *shm_unlink()*
61605 XBD [Section 3.342](#) (on page 88), [<sys/shm.h>](#)

61606 CHANGE HISTORY

61607 First released in Issue 2. Derived from Issue 2 of the SVID.

61608 Issue 5

61609 Moved from SHARED MEMORY to BASE.

61610 The note about use of POSIX Realtime Extension IPC routines has been moved from FUTURE
61611 DIRECTIONS to a new APPLICATION USAGE section.

61612 Issue 6

61613 The Open Group Corrigendum U021/13 is applied.

61614 Issue 7

61615 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0572 [345] is applied.

+

61616 **NAME**

61617 shmctl — XSI shared memory control operations

61618 **SYNOPSIS**

```
61619 XSI      #include <sys/shm.h>
61620      int shmctl(int shmid, int cmd, struct shm_id *buf);
```

61621 **DESCRIPTION**

61622 The *shmctl()* function operates on XSI shared memory (see XBD [Section 3.342](#), on page 88). It is
 61623 unspecified whether this function interoperates with the realtime interprocess communication
 61624 facilities defined in [Section 2.8](#) (on page 502).

61625 The *shmctl()* function provides a variety of shared memory control operations as specified by
 61626 *cmd*. The following values for *cmd* are available:

61627 **IPC_STAT** Place the current value of each member of the **shm_id** data structure
 61628 associated with *shmid* into the structure pointed to by *buf*. The contents of the
 61629 structure are defined in **<sys/shm.h>**.

61630 **IPC_SET** Set the value of the following members of the **shm_id** data structure
 61631 associated with *shmid* to the corresponding value found in the structure
 61632 pointed to by *buf*:

```
61633     shm_perm.uid
61634     shm_perm.gid
61635     shm_perm.mode      Low-order nine bits.
```

61636 Also, the *shm_ctime* timestamp shall be set to the current time, as described in +
 61637 [Section 2.7.1](#) (on page 501). +

61638 IPC_SET can only be executed by a process that has an effective user ID equal
 61639 to either that of a process with appropriate privileges or to the value of
 61640 *shm_perm.cuid* or *shm_perm.uid* in the **shm_id** data structure associated with
 61641 *shmid*.

61642 **IPC_RMID** Remove the shared memory identifier specified by *shmid* from the system and
 61643 destroy the shared memory segment and **shm_id** data structure associated
 61644 with it. IPC_RMID can only be executed by a process that has an effective user
 61645 ID equal to either that of a process with appropriate privileges or to the value
 61646 of *shm_perm.cuid* or *shm_perm.uid* in the **shm_id** data structure associated
 61647 with *shmid*.

61648 **RETURN VALUE**

61649 Upon successful completion, *shmctl()* shall return 0; otherwise, it shall return -1 and set *errno* to
 61650 indicate the error.

61651 **ERRORS**

61652 The *shmctl()* function shall fail if:

61653 [EACCES] The argument *cmd* is equal to IPC_STAT and the calling process does not have
 61654 read permission; see [Section 2.7](#) (on page 501).

61655 [EINVAL] The value of *shmid* is not a valid shared memory identifier, or the value of *cmd*
 61656 is not a valid command.

61657 [EPERM] The argument *cmd* is equal to IPC_RMID or IPC_SET and the effective user ID
 61658 of the calling process is not equal to that of a process with appropriate
 61659 privileges and it is not equal to the value of *shm_perm.cuid* or *shm_perm.uid* in

61660 the data structure associated with *shmid*.

61661 The *shmctl()* function may fail if:

61662 [EOVERFLOW] The *cmd* argument is IPC_STAT and the *gid* or *uid* value is too large to be
61663 stored in the structure pointed to by the *buf* argument.

61664 EXAMPLES

61665 None.

61666 APPLICATION USAGE

61667 The POSIX Realtime Extension defines alternative interfaces for interprocess communication.
61668 Application developers who need to use IPC should design their applications so that modules
61669 using the IPC routines described in [Section 2.7](#) (on page 501) can be easily modified to use the
61670 alternative interfaces.

61671 RATIONALE

61672 None.

61673 FUTURE DIRECTIONS

61674 None.

61675 SEE ALSO

61676 [Section 2.7](#) (on page 501), [Section 2.8](#) (on page 502), [shmat\(\)](#), [shmdt\(\)](#), [shmget\(\)](#), [shm_open\(\)](#),
61677 [shm_unlink\(\)](#)

61678 XBD [Section 3.342](#) (on page 88), [<sys/shm.h>](#)

61679 CHANGE HISTORY

61680 First released in Issue 2. Derived from Issue 2 of the SVID.

61681 Issue 5

61682 Moved from SHARED MEMORY to BASE.

61683 The note about use of POSIX Realtime Extension IPC routines has been moved from FUTURE
61684 DIRECTIONS to a new APPLICATION USAGE section.

61685 Issue 7

61686 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0573 [345] is applied.

+

NAME

shmdt — XSI shared memory detach operation

SYNOPSIS

```
XSI    #include <sys/shm.h>
      int shmdt(const void *shmaddr);
```

DESCRIPTION

The *shmdt()* function operates on XSI shared memory (see XBD [Section 3.342](#), on page 88). It is unspecified whether this function interoperates with the realtime interprocess communication facilities defined in [Section 2.8](#) (on page 502).

The *shmdt()* function detaches the shared memory segment located at the address specified by *shmaddr* from the address space of the calling process.

RETURN VALUE

Upon successful completion, *shmdt()* shall decrement the value of *shm_nattch* in the data structure associated with the shared memory ID of the attached shared memory segment and return 0. Also, the *shm_dtime* timestamp shall be set to the current time, as described in [Section 2.7.1](#) (on page 501).

Otherwise, the shared memory segment shall not be detached, *shmdt()* shall return -1, and *errno* shall be set to indicate the error.

ERRORS

The *shmdt()* function shall fail if:

[EINVAL]	The value of <i>shmaddr</i> is not the data segment start address of a shared memory segment.

EXAMPLES

None.

APPLICATION USAGE

The POSIX Realtime Extension defines alternative interfaces for interprocess communication. Application developers who need to use IPC should design their applications so that modules using the IPC routines described in [Section 2.7](#) (on page 501) can be easily modified to use the alternative interfaces.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

[Section 2.7](#) (on page 501), [Section 2.8](#) (on page 502), *exec*, *exit()*, *fork()*, *shmat()*, *shmctl()*, *shmget()*, *shm_open()*, *shm_unlink()*

XBD [Section 3.342](#) (on page 88), [<sys/shm.h>](#)

CHANGE HISTORY

First released in Issue 2. Derived from Issue 2 of the SVID.

Issue 5

61726
61727 Moved from SHARED MEMORY to BASE.

61728 The note about use of POSIX Realtime Extension IPC routines has been moved from FUTURE
61729 DIRECTIONS to a new APPLICATION USAGE section.

Issue 7

61730
61731 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0574 [345] is applied.

+

61732 **NAME**

61733 shmget — get an XSI shared memory segment

61734 **SYNOPSIS**

```
61735 XSI      #include <sys/shm.h>
61736      int shmget(key_t key, size_t size, int shmflg);
```

61737 **DESCRIPTION**

61738 The *shmget()* function operates on XSI shared memory (see XBD [Section 3.342](#), on page 88). It is
 61739 unspecified whether this function interoperates with the realtime interprocess communication
 61740 facilities defined in [Section 2.8](#) (on page 502).

61741 The *shmget()* function shall return the shared memory identifier associated with *key*.

61742 A shared memory identifier, associated data structure, and shared memory segment of at least
 61743 *size* bytes (see **<sys/shm.h>**) are created for *key* if one of the following is true:

- 61744 • The argument *key* is equal to `IPC_PRIVATE`.
- 61745 • The argument *key* does not already have a shared memory identifier associated with it and
 61746 (*shmflg* & `IPC_CREAT`) is non-zero.

61747 Upon creation, the data structure associated with the new shared memory identifier shall be
 61748 initialized as follows:

- 61749 • The values of *shm_perm.cuid*, *shm_perm.uid*, *shm_perm.cgid*, and *shm_perm.gid* are set to the |
 61750 effective user ID and effective group ID, respectively, of the calling process.
- 61751 • The low-order nine bits of *shm_perm.mode* are set to the low-order nine bits of *shmflg*. |
- 61752 • The value of *shm_segsz* is set to the value of *size*. |
- 61753 • The values of *shm_lpid*, *shm_nattch*, *shm_atime*, and *shm_dtime* are set to 0. |
- 61754 • The value of *shm_ctime* is set to the current time, as described in [Section 2.7.1](#) (on page 501). |

61755 When the shared memory segment is created, it shall be initialized with all zero values.

61756 **RETURN VALUE**

61757 Upon successful completion, *shmget()* shall return a non-negative integer, namely a shared
 61758 memory identifier; otherwise, it shall return `-1` and set *errno* to indicate the error.

61759 **ERRORS**

61760 The *shmget()* function shall fail if:

61761 61762 61763	[EACCES]	A shared memory identifier exists for <i>key</i> but operation permission as specified by the low-order nine bits of <i>shmflg</i> would not be granted; see Section 2.7 (on page 501).
61764 61765	[EEXIST]	A shared memory identifier exists for the argument <i>key</i> but (<i>shmflg</i> & <code>IPC_CREAT</code>) && (<i>shmflg</i> & <code>IPC_EXCL</code>) is non-zero.
61766 61767	[EINVAL]	A shared memory segment is to be created and the value of <i>size</i> is less than the system-imposed minimum or greater than the system-imposed maximum.
61768 61769	[EINVAL]	No shared memory segment is to be created and a shared memory segment exists for <i>key</i> but the size of the segment associated with it is less than <i>size</i> .
61770 61771	[ENOENT]	A shared memory identifier does not exist for the argument <i>key</i> and (<i>shmflg</i> & <code>IPC_CREAT</code>) is 0.

61772 [ENOMEM] A shared memory identifier and associated shared memory segment shall be
 61773 created, but the amount of available physical memory is not sufficient to fill
 61774 the request.

61775 [ENOSPC] A shared memory identifier is to be created, but the system-imposed limit on
 61776 the maximum number of allowed shared memory identifiers system-wide
 61777 would be exceeded.

61778 EXAMPLES

61779 None.

61780 APPLICATION USAGE

61781 The POSIX Realtime Extension defines alternative interfaces for interprocess communication.
 61782 Application developers who need to use IPC should design their applications so that modules
 61783 using the IPC routines described in [Section 2.7](#) (on page 501) can be easily modified to use the
 61784 alternative interfaces.

61785 RATIONALE

61786 None.

61787 FUTURE DIRECTIONS

61788 None.

61789 SEE ALSO

61790 [Section 2.7](#) (on page 501), [Section 2.8](#) (on page 502), [ftok\(\)](#), [shmat\(\)](#), [shmctl\(\)](#), [shmdt\(\)](#), [shm_open\(\)](#),
 61791 [shm_unlink\(\)](#)

61792 XBD [Section 3.342](#) (on page 88), [<sys/shm.h>](#)

61793 CHANGE HISTORY

61794 First released in Issue 2. Derived from Issue 2 of the SVID.

61795 Issue 5

61796 Moved from SHARED MEMORY to BASE.

61797 The note about use of POSIX Realtime Extension IPC routines has been moved from FUTURE
 61798 DIRECTIONS to a new APPLICATION USAGE section.

61799 Issue 7

61800 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0575 [345], XSH/TC1-2008/0576 [363], +
 61801 and XSH/TC1-2008/0577 [344] are applied.

61802 **NAME**

61803 shutdown — shut down socket send and receive operations

61804 **SYNOPSIS**

61805 #include <sys/socket.h>

61806 int shutdown(int *socket*, int *how*);61807 **DESCRIPTION**61808 The *shutdown()* function shall cause all or part of a full-duplex connection on the socket
61809 associated with the file descriptor *socket* to be shut down.61810 The *shutdown()* function takes the following arguments:61811 *socket* Specifies the file descriptor of the socket.61812 *how* Specifies the type of shutdown. The values are as follows:

61813 SHUT_RD Disables further receive operations.

61814 SHUT_WR Disables further send operations.

61815 SHUT_RDWR Disables further send and receive operations.

61816 The *shutdown()* function disables subsequent send and/or receive operations on a socket,
61817 depending on the value of the *how* argument.61818 **RETURN VALUE**61819 Upon successful completion, *shutdown()* shall return 0; otherwise, -1 shall be returned and *errno*
61820 set to indicate the error.61821 **ERRORS**61822 The *shutdown()* function shall fail if:61823 [EBADF] The *socket* argument is not a valid file descriptor.61824 [EINVAL] The *how* argument is invalid.

61825 [ENOTCONN] The socket is not connected.

61826 [ENOTSOCK] The *socket* argument does not refer to a socket.61827 The *shutdown()* function may fail if:

61828 [ENOBUFS] Insufficient resources were available in the system to perform the operation.

61829 **EXAMPLES**

61830 None.

61831 **APPLICATION USAGE**

61832 None.

61833 **RATIONALE**

61834 None.

61835 **FUTURE DIRECTIONS**

61836 None.

61837 **SEE ALSO**61838 *getsockopt()*, *pselect()*, *read()*, *recv()*, *recvfrom()*, *recvmsg()*, *send()*, *sendto()*, *setsockopt()*, *socket()*,
61839 *write()*

61840 XBD <sys/socket.h>

61841	CHANGE HISTORY
61842	First released in Issue 6. Derived from the XNS, Issue 5.2 specification.

NAME

sigaction — examine and change a signal action

SYNOPSIS

```
CX      #include <signal.h>

      int sigaction(int sig, const struct sigaction *restrict act,
                    struct sigaction *restrict oact);
```

DESCRIPTION

The *sigaction()* function allows the calling process to examine and/or specify the action to be associated with a specific signal. The argument *sig* specifies the signal; acceptable values are defined in *<signal.h>*.

The structure **sigaction**, used to describe an action to be taken, is defined in the *<signal.h>* header to include at least the following members:

Member Type	Member Name	Description
void(*) (int)	<i>sa_handler</i>	Pointer to a signal-catching function or one of the macros SIG_IGN or SIG_DFL.
sigset_t	<i>sa_mask</i>	Additional set of signals to be blocked during execution of signal-catching function.
int	<i>sa_flags</i>	Special flags to affect behavior of signal.
void(*) (int, siginfo_t *, void *)	<i>sa_sigaction</i>	Pointer to a signal-catching function.

The storage occupied by *sa_handler* and *sa_sigaction* may overlap, and a conforming application shall not use both simultaneously.

If the argument *act* is not a null pointer, it points to a structure specifying the action to be associated with the specified signal. If the argument *oact* is not a null pointer, the action previously associated with the signal is stored in the location pointed to by the argument *oact*. If the argument *act* is a null pointer, signal handling is unchanged; thus, the call can be used to enquire about the current handling of a given signal. The SIGKILL and SIGSTOP signals shall not be added to the signal mask using this mechanism; this restriction shall be enforced by the system without causing an error to be indicated.

If the SA_SIGINFO flag (see below) is cleared in the *sa_flags* field of the **sigaction** structure, the *sa_handler* field identifies the action to be associated with the specified signal. If the SA_SIGINFO flag is set in the *sa_flags* field, the *sa_sigaction* field specifies a signal-catching function.

The *sa_flags* field can be used to modify the behavior of the specified signal.

The following flags, defined in the *<signal.h>* header, can be set in *sa_flags*:

XSI	SA_NOCLDSTOP	Do not generate SIGCHLD when children stop or stopped children continue.
		If <i>sig</i> is SIGCHLD and the SA_NOCLDSTOP flag is not set in <i>sa_flags</i> , and the implementation supports the SIGCHLD signal, then a SIGCHLD signal shall be generated for the calling process whenever any of its child processes stop and a SIGCHLD signal may be generated for the calling process whenever any of its stopped child processes are continued. If <i>sig</i> is SIGCHLD and the SA_NOCLDSTOP flag is set in <i>sa_flags</i> , then the implementation shall not generate a SIGCHLD signal in this way.

61888	XSI	SA_ONSTACK	If set and an alternate signal stack has been declared with <i>sigaltstack()</i> , the signal shall be delivered to the calling process on that stack. Otherwise, the signal shall be delivered on the current stack.
61889			
61890			
61891		SA_RESETHAND	If set, the disposition of the signal shall be reset to SIG_DFL and the SA_SIGINFO flag shall be cleared on entry to the signal handler.
61892			
61893			Note: SIGILL and SIGTRAP cannot be automatically reset when delivered; the system silently enforces this restriction.
61894			
61895			Otherwise, the disposition of the signal shall not be modified on entry to the signal handler.
61896			
61897			In addition, if this flag is set, <i>sigaction()</i> may behave as if the SA_NODEFER flag were also set.
61898			
61899		SA_RESTART	This flag affects the behavior of interruptible functions; that is, those specified to fail with <i>errno</i> set to [EINTR]. If set, and a function specified as interruptible is interrupted by this signal, the function shall restart and shall not fail with [EINTR] unless otherwise specified. If an interruptible function which uses a timeout is restarted, the duration of the timeout following the restart is set to an unspecified value that does not exceed the original timeout value. If the flag is not set, interruptible functions interrupted by this signal shall fail with <i>errno</i> set to [EINTR].
61900			
61901			
61902			
61903			
61904			
61905			
61906			
61907		SA_SIGINFO	If cleared and the signal is caught, the signal-catching function shall be entered as:
61908			
61909			<pre>void func(int signo);</pre>
61910			where <i>signo</i> is the only argument to the signal-catching function. In this case, the application shall use the <i>sa_handler</i> member to describe the signal-catching function and the application shall not modify the <i>sa_sigaction</i> member.
61911			
61912			
61913			
61914			If SA_SIGINFO is set and the signal is caught, the signal-catching function shall be entered as:
61915			
61916			<pre>void func(int signo, siginfo_t *info, void *context);</pre>
61917			where two additional arguments are passed to the signal-catching function. The second argument shall point to an object of type siginfo_t explaining the reason why the signal was generated; the third argument can be cast to a pointer to an object of type ucontext_t to refer to the receiving thread's context that was interrupted when the signal was delivered. In this case, the application shall use the <i>sa_sigaction</i> member to describe the signal-catching function and the application shall not modify the <i>sa_handler</i> member.
61918			
61919			
61920			
61921			
61922			
61923			
61924			
61925			The <i>si_signo</i> member contains the system-generated signal number.
61926	XSI		The <i>si_errno</i> member may contain implementation-defined additional error information; if non-zero, it contains an error number identifying the condition that caused the signal to be generated.
61927			
61928			
61929			The <i>si_code</i> member contains a code identifying the cause of the signal, as described in Section 2.4.3 (on page 490).
61930			

61931 SA_NOCLDWAIT If set, and *sig* equals SIGCHLD, child processes of the calling processes
 61932 shall not be transformed into zombie processes when they terminate. If
 61933 the calling process subsequently waits for its children, and the process has
 61934 no unwaited-for children that were transformed into zombie processes, it
 61935 shall block until all of its children terminate, and *wait()*, *waitid()*, and
 61936 *waitpid()* shall fail and set *errno* to [ECHILD]. Otherwise, terminating
 61937 child processes shall be transformed into zombie processes, unless
 61938 SIGCHLD is set to SIG_IGN.

61939 SA_NODEFER If set and *sig* is caught, *sig* shall not be added to the thread's signal mask
 61940 on entry to the signal handler unless it is included in *sa_mask*. Otherwise,
 61941 *sig* shall always be added to the thread's signal mask on entry to the
 61942 signal handler.

61943 When a signal is caught by a signal-catching function installed by *sigaction()*, a new signal mask
 61944 is calculated and installed for the duration of the signal-catching function (or until a call to either
 61945 *sigprocmask()* or *sigsuspend()* is made). This mask is formed by taking the union of the current
 61946 signal mask and the value of the *sa_mask* for the signal being delivered, and unless
 61947 SA_NODEFER or SA_RESETHAND is set, then including the signal being delivered. If and
 61948 when the user's signal handler returns normally, the original signal mask is restored.

61949 Once an action is installed for a specific signal, it shall remain installed until another action is
 61950 explicitly requested (by another call to *sigaction()*), until the SA_RESETHAND flag causes
 61951 resetting of the handler, or until one of the *exec* functions is called.

61952 If the previous action for *sig* had been established by *signal()*, the values of the fields returned in
 61953 the structure pointed to by *oact* are unspecified, and in particular *oact->sa_handler* is not
 61954 necessarily the same value passed to *signal()*. However, if a pointer to the same structure or a
 61955 copy thereof is passed to a subsequent call to *sigaction()* via the *act* argument, handling of the
 61956 signal shall be as if the original call to *signal()* were repeated.

61957 If *sigaction()* fails, no new signal handler is installed.

61958 It is unspecified whether an attempt to set the action for a signal that cannot be caught or
 61959 ignored to SIG_DFL is ignored or causes an error to be returned with *errno* set to [EINVAL].

61960 If SA_SIGINFO is not set in *sa_flags*, then the disposition of subsequent occurrences of *sig* when
 61961 it is already pending is implementation-defined; the signal-catching function shall be invoked
 61962 with a single argument. If SA_SIGINFO is set in *sa_flags*, then subsequent occurrences of *sig*
 61963 generated by *sigqueue()* or as a result of any signal-generating function that supports the
 61964 specification of an application-defined value (when *sig* is already pending) shall be queued in
 61965 FIFO order until delivered or accepted; the signal-catching function shall be invoked with three
 61966 arguments. The application specified value is passed to the signal-catching function as the
 61967 *si_value* member of the **siginfo_t** structure.

61968 The result of the use of *sigaction()* and a *sigwait()* function concurrently within a process on the
 61969 same signal is unspecified.

61970 RETURN VALUE

61971 Upon successful completion, *sigaction()* shall return 0; otherwise, -1 shall be returned, *errno* shall
 61972 be set to indicate the error, and no new signal-catching function shall be installed.

ERRORS

The *sigaction()* function shall fail if:

[EINVAL] The *sig* argument is not a valid signal number or an attempt is made to catch a signal that cannot be caught or ignore a signal that cannot be ignored.

[ENOTSUP] The SA_SIGINFO bit flag is set in the *sa_flags* field of the **sigaction** structure.

The *sigaction()* function may fail if:

[EINVAL] An attempt was made to set the action to SIG_DFL for a signal that cannot be caught or ignored (or both).

In addition, the *sigaction()* function may fail if the SA_SIGINFO flag is set in the *sa_flags* field of the **sigaction** structure for a signal not in the range SIGRTMIN to SIGRTMAX.

EXAMPLES**Establishing a Signal Handler**

The following example demonstrates the use of *sigaction()* to establish a handler for the SIGINT signal.

```
#include <signal.h>

static void handler(int signum)
{
    /* Take appropriate actions for signal delivery */
}

int main()
{
    struct sigaction sa;

    sa.sa_handler = handler;
    sigemptyset(&sa.sa_mask);
    sa.sa_flags = SA_RESTART; /* Restart functions if
                               interrupted by handler */
    if (sigaction(SIGINT, &sa, NULL) == -1)
        /* Handle error */;

    /* Further code */
}
```

APPLICATION USAGE

The *sigaction()* function supersedes the *signal()* function, and should be used in preference. In particular, *sigaction()* and *signal()* should not be used in the same process to control the same signal. The behavior of async-signal-safe functions, as defined in their respective DESCRIPTION sections, is as specified by this volume of POSIX.1-2008, regardless of invocation from a signal-catching function. This is the only intended meaning of the statement that async-signal-safe functions may be used in signal-catching functions without restrictions. Applications must still consider all effects of such functions on such things as data structures, files, and process state. In particular, application developers need to consider the restrictions on interactions when interrupting *sleep()* and interactions among multiple handles for a file description. The fact that any specific function is listed as async-signal-safe does not necessarily mean that invocation of that function from a signal-catching function is recommended.

In order to prevent errors arising from interrupting non-async-signal-safe function calls, applications should protect calls to these functions either by blocking the appropriate signals or

through the use of some programmatic semaphore (see [*semget\(\)*](#), [*sem_init\(\)*](#), [*sem_open\(\)*](#), and so on). Note in particular that even the “safe” functions may modify *errno*; the signal-catching function, if not executing as an independent thread, should save and restore its value in order to avoid the possibility that delivery of a signal in between an error return from a function that sets *errno* and the subsequent examination of *errno* could result in the signal-catching function changing the value of *errno*. Naturally, the same principles apply to the async-signal-safety of application routines and asynchronous data access. Note that [*longjmp\(\)*](#) and [*siglongjmp\(\)*](#) are not in the list of async-signal-safe functions. This is because the code executing after [*longjmp\(\)*](#) and [*siglongjmp\(\)*](#) can call any unsafe functions with the same danger as calling those unsafe functions directly from the signal handler. Applications that use [*longjmp\(\)*](#) and [*siglongjmp\(\)*](#) from within signal handlers require rigorous protection in order to be portable. Many of the other functions that are excluded from the list are traditionally implemented using either [*malloc\(\)*](#) or [*free\(\)*](#) functions or the standard I/O library, both of which traditionally use data structures in a non-async-signal-safe manner. Since any combination of different functions using a common data structure can cause async-signal-safety problems, this volume of POSIX.1-2008 does not define the behavior when any unsafe function is called in a signal handler that interrupts an unsafe function.

Usually, the signal is executed on the stack that was in effect before the signal was delivered. An alternate stack may be specified to receive a subset of the signals being caught.

When the signal handler returns, the receiving thread resumes execution at the point it was interrupted unless the signal handler makes other arrangements. If [*longjmp\(\)*](#) or [*_longjmp\(\)*](#) is used to leave the signal handler, then the signal mask must be explicitly restored.

This volume of POSIX.1-2008 defines the third argument of a signal handling function when [*SA_SIGINFO*](#) is set as a **void *** instead of a **ucontext_t ***, but without requiring type checking. New applications should explicitly cast the third argument of the signal handling function to **ucontext_t ***.

The BSD optional four argument signal handling function is not supported by this volume of POSIX.1-2008. The BSD declaration would be:

```
void handler(int sig, int code, struct sigcontext *scp,
             char *addr);
```

where *sig* is the signal number, *code* is additional information on certain signals, *scp* is a pointer to the **sigcontext** structure, and *addr* is additional address information. Much the same information is available in the objects pointed to by the second argument of the signal handler specified when [*SA_SIGINFO*](#) is set.

Since the [*sigaction\(\)*](#) function is allowed but not required to set [*SA_NODEFER*](#) when the application sets the [*SA_RESETHAND*](#) flag, applications which depend on the [*SA_RESETHAND*](#) functionality for the newly installed signal handler must always explicitly set [*SA_NODEFER*](#) when they set [*SA_RESETHAND*](#) in order to be portable.

See also the rationale for Realtime Signal Generation and Delivery in XRAT [Section B.2.4.2](#) (on page 3538).

RATIONALE

Although this volume of POSIX.1-2008 requires that signals that cannot be ignored shall not be added to the signal mask when a signal-catching function is entered, there is no explicit requirement that subsequent calls to [*sigaction\(\)*](#) reflect this in the information returned in the *oact* argument. In other words, if [*SIGKILL*](#) is included in the *sa_mask* field of *act*, it is unspecified whether or not a subsequent call to [*sigaction\(\)*](#) returns with [*SIGKILL*](#) included in the *sa_mask* field of *oact*.

The SA_NOCLDSTOP flag, when supplied in the *act->sa_flags* parameter, allows overloading SIGCHLD with the System V semantics that each SIGCHLD signal indicates a single terminated child. Most conforming applications that catch SIGCHLD are expected to install signal-catching functions that repeatedly call the *waitpid()* function with the WNOHANG flag set, acting on each child for which status is returned, until *waitpid()* returns zero. If stopped children are not of interest, the use of the SA_NOCLDSTOP flag can prevent the overhead from invoking the signal-catching routine when they stop.

Some historical implementations also define other mechanisms for stopping processes, such as the *ptrace()* function. These implementations usually do not generate a SIGCHLD signal when processes stop due to this mechanism; however, that is beyond the scope of this volume of POSIX.1-2008.

This volume of POSIX.1-2008 requires that calls to *sigaction()* that supply a NULL *act* argument succeed, even in the case of signals that cannot be caught or ignored (that is, SIGKILL or SIGSTOP). The System V *signal()* and BSD *sigvec()* functions return [EINVAL] in these cases and, in this respect, their behavior varies from *sigaction()*.

This volume of POSIX.1-2008 requires that *sigaction()* properly save and restore a signal action set up by the ISO C standard *signal()* function. However, there is no guarantee that the reverse is true, nor could there be given the greater amount of information conveyed by the **sigaction** structure. Because of this, applications should avoid using both functions for the same signal in the same process. Since this cannot always be avoided in case of general-purpose library routines, they should always be implemented with *sigaction()*.

It was intended that the *signal()* function should be implementable as a library routine using *sigaction()*.

The POSIX Realtime Extension extends the *sigaction()* function as specified by the POSIX.1-1990 standard to allow the application to request on a per-signal basis via an additional signal action flag that the extra parameters, including the application-defined signal value, if any, be passed to the signal-catching function.

FUTURE DIRECTIONS

None.

SEE ALSO

Section 2.4 (on page 488), *exec*, *kill()*, *_longjmp()*, *longjmp()*, *pthread_sigmask()*, *raise()*, *semget()*, *sem_init()*, *sem_open()*, *sigaddset()*, *sigaltstack()*, *sigdelset()*, *sigemptyset()*, *sigfillset()*, *sigismember()*, *signal()*, *sigsuspend()*, *wait()*, *waitid()*

XBD <signal.h>

CHANGE HISTORY

First released in Issue 3. Included for alignment with the POSIX.1-1988 standard.

Issue 5

The DESCRIPTION is updated for alignment with the POSIX Realtime Extension and POSIX Threads Extension.

In the DESCRIPTION, the second argument to *func* when SA_SIGINFO is set is no longer permitted to be NULL, and the description of permitted **siginfo_t** contents is expanded by reference to <signal.h>.

Since the X/OPEN UNIX Extension functionality is now folded into the BASE, the [ENOTSUP] error is deleted.

Issue 6

The Open Group Corrigendum U028/7 is applied. In the paragraph entitled “Signal Effects on Other Functions”, a reference to *sigpending()* is added.

In the DESCRIPTION, the text “Signal Generation and Delivery”, “Signal Actions”, and “Signal Effects on Other Functions” are moved to a separate section of this volume of POSIX.1-2008.

Text describing functionality from the Realtime Signals Extension option is marked.

The following changes are made for alignment with the ISO POSIX-1: 1996 standard:

- The [ENOTSUP] error condition is added.

The normative text is updated to avoid use of the term “must” for application requirements.

The **restrict** keyword is added to the *sigaction()* prototype for alignment with the ISO/IEC 9899: 1999 standard.

References to the *wait3()* function are removed.

The SYNOPSIS is marked CX since the presence of this function in the **<signal.h>** header is an extension over the ISO C standard.

IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/57 is applied, changing text in the table describing the **sigaction** structure.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/127 is applied, removing text from the DESCRIPTION duplicated later in the same section.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/128 is applied, updating the DESCRIPTION and APPLICATION USAGE sections. Changes are made to refer to the thread rather than the process.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/129 is applied, adding the example to the EXAMPLES section.

Issue 7

Austin Group Interpretation 1003.1-2001 #004 is applied.

Austin Group Interpretations 1003.1-2001 #065 and #084 are applied, clarifying the role of the SA_NODEFER flag with respect to the signal mask, and clarifying the SA_RESTART flag for interrupted functions which use timeouts.

Austin Group Interpretation 1003.1-2001 #156 is applied.

SD5-XSH-ERN-167 is applied, updating the APPLICATION USAGE section.

SD5-XSH-ERN-172 is applied, updating the DESCRIPTION to make optional the requirement that when the SA_RESETHAND flag is set, *sigaction()* shall behave as if the SA_NODEFER flag were also set.

Functionality relating to the Realtime Signals Extension option is moved to the Base.

The description of the *si_code* member is replaced with a reference to [Section 2.4.3](#) (on page 490).

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0578 [66] and XSH/TC1-2008/0579 + [140] are applied.

62145 **NAME**

62146 sigaddset — add a signal to a signal set

62147 **SYNOPSIS**

```
62148 CX    #include <signal.h>
62149    int sigaddset(sigset_t *set, int signo);
```

62150 **DESCRIPTION**

62151 The *sigaddset()* function adds the individual signal specified by the *signo* to the signal set pointed
 62152 to by *set*.

62153 Applications shall call either *sigemptyset()* or *sigfillset()* at least once for each object of type
 62154 **sigset_t** prior to any other use of that object. If such an object is not initialized in this way, but is
 62155 nonetheless supplied as an argument to any of *pthread_sigmask()*, *sigaction()*, *sigaddset()*,
 62156 *sigdelset()*, *sigismember()*, *sigpending()*, *sigprocmask()*, *sigsuspend()*, *sigtimedwait()*, *sigwait()*, or
 62157 *sigwaitinfo()*, the results are undefined.

62158 **RETURN VALUE**

62159 Upon successful completion, *sigaddset()* shall return 0; otherwise, it shall return -1 and set *errno*
 62160 to indicate the error.

62161 **ERRORS**

62162 The *sigaddset()* function may fail if:

62163 [EINVAL] The value of the *signo* argument is an invalid or unsupported signal number.

62164 **EXAMPLES**

62165 None.

62166 **APPLICATION USAGE**

62167 None.

62168 **RATIONALE**

62169 None.

62170 **FUTURE DIRECTIONS**

62171 None.

62172 **SEE ALSO**

62173 [Section 2.4](#) (on page 488), *pthread_sigmask()*, *sigaction()*, *sigdelset()*, *sigemptyset()*, *sigfillset()*,
 62174 *sigismember()*, *sigpending()*, *sigsuspend()*

62175 XBD [<signal.h>](#)

62176 **CHANGE HISTORY**

62177 First released in Issue 3. Included for alignment with the POSIX.1-1988 standard.

62178 **Issue 5**

62179 The last paragraph of the DESCRIPTION was included as an APPLICATION USAGE note in
 62180 previous issues.

62181 **Issue 6**

62182 The normative text is updated to avoid use of the term “must” for application requirements.

62183 The SYNOPSIS is marked CX since the presence of this function in the **<signal.h>** header is an
 62184 extension over the ISO C standard.

NAME

sigaltstack — set and get signal alternate stack context

SYNOPSIS

```
#include <signal.h>

int sigaltstack(const stack_t *restrict ss, stack_t *restrict oss);
```

DESCRIPTION

The *sigaltstack()* function allows a process to define and examine the state of an alternate stack for signal handlers for the current thread. Signals that have been explicitly declared to execute on the alternate stack shall be delivered on the alternate stack.

If *ss* is not a null pointer, it points to a **stack_t** structure that specifies the alternate signal stack that shall take effect upon return from *sigaltstack()*. The *ss_flags* member specifies the new stack state. If it is set to *SS_DISABLE*, the stack is disabled and *ss_sp* and *ss_size* are ignored. Otherwise, the stack shall be enabled, and the *ss_sp* and *ss_size* members specify the new address and size of the stack.

The range of addresses starting at *ss_sp* up to but not including *ss_sp+ss_size* is available to the implementation for use as the stack. This function makes no assumptions regarding which end is the stack base and in which direction the stack grows as items are pushed.

If *oss* is not a null pointer, upon successful completion it shall point to a **stack_t** structure that specifies the alternate signal stack that was in effect prior to the call to *sigaltstack()*. The *ss_sp* and *ss_size* members specify the address and size of that stack. The *ss_flags* member specifies the stack's state, and may contain one of the following values:

SS_ONSTACK The process is currently executing on the alternate signal stack. Attempts to modify the alternate signal stack while the process is executing on it fail. This flag shall not be modified by processes.

SS_DISABLE The alternate signal stack is currently disabled.

The value *SIGSTKSZ* is a system default specifying the number of bytes that would be used to cover the usual case when manually allocating an alternate stack area. The value *MINSIGSTKSZ* is defined to be the minimum stack size for a signal handler. In computing an alternate stack size, a program should add that amount to its stack requirements to allow for the system implementation overhead. The constants *SS_ONSTACK*, *SS_DISABLE*, *SIGSTKSZ*, and *MINSIGSTKSZ* are defined in **<signal.h>**.

After a successful call to one of the *exec* functions, there are no alternate signal stacks in the new process image.

In some implementations, a signal (whether or not indicated to execute on the alternate stack) shall always execute on the alternate stack if it is delivered while another signal is being caught using the alternate stack.

Use of this function by library threads that are not bound to kernel-scheduled entities results in undefined behavior.

RETURN VALUE

Upon successful completion, *sigaltstack()* shall return 0; otherwise, it shall return -1 and set *errno* to indicate the error.

ERRORS

The *sigaltstack()* function shall fail if:

- [EINVAL] The *ss* argument is not a null pointer, and the *ss_flags* member pointed to by *ss* contains flags other than *SS_DISABLE*.
- [ENOMEM] The size of the alternate stack area is less than *MINSIGSTKSZ*.
- [EPERM] An attempt was made to modify an active stack.

EXAMPLES**Allocating Memory for an Alternate Stack**

The following example illustrates a method for allocating memory for an alternate stack.

```
#include <signal.h>
...
if ((sigstk.ss_sp = malloc(SIGSTKSZ)) == NULL)
    /* Error return. */
sigstk.ss_size = SIGSTKSZ;
sigstk.ss_flags = 0;
if (sigaltstack(&sigstk, (stack_t *)0) < 0)
    perror("sigaltstack");
```

APPLICATION USAGE

On some implementations, stack space is automatically extended as needed. On those implementations, automatic extension is typically not available for an alternate stack. If the stack overflows, the behavior is undefined.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

[Section 2.4](#) (on page 488), *exec*, *sigaction()*, *sigsetjmp()*

XBD [<signal.h>](#)

CHANGE HISTORY

First released in Issue 4, Version 2.

Issue 5

Moved from X/OPEN UNIX extension to BASE.

The last sentence of the DESCRIPTION was included as an APPLICATION USAGE note in previous issues.

Issue 6

The normative text is updated to avoid use of the term “must” for application requirements.

The **restrict** keyword is added to the *sigaltstack()* prototype for alignment with the ISO/IEC 9899:1999 standard.

IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/58 is applied, updating the first sentence to include “for the current thread”.

NAME

sigdelset — delete a signal from a signal set

SYNOPSIS

```
CX #include <signal.h>

int sigdelset(sigset_t *set, int signo);
```

DESCRIPTION

The *sigdelset()* function deletes the individual signal specified by *signo* from the signal set pointed to by *set*.

Applications should call either *sigemptyset()* or *sigfillset()* at least once for each object of type **sigset_t** prior to any other use of that object. If such an object is not initialized in this way, but is nonetheless supplied as an argument to any of *pthread_sigmask()*, *sigaction()*, *sigaddset()*, *sigdelset()*, *sigismember()*, *sigpending()*, *sigprocmask()*, *sigsuspend()*, *sigtimedwait()*, *sigwait()*, or *sigwaitinfo()*, the results are undefined.

RETURN VALUE

Upon successful completion, *sigdelset()* shall return 0; otherwise, it shall return -1 and set *errno* to indicate the error.

ERRORS

The *sigdelset()* function may fail if:

[EINVAL]	The <i>signo</i> argument is not a valid signal number, or is an unsupported signal number.

EXAMPLES

None.

APPLICATION USAGE

None.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

Section 2.4 (on page 488), *pthread_sigmask()*, *sigaction()*, *sigaddset()*, *sigemptyset()*, *sigfillset()*, *sigismember()*, *sigpending()*, *sigsuspend()*

XBD **<signal.h>**

CHANGE HISTORY

First released in Issue 3. Included for alignment with the POSIX.1-1988 standard.

Issue 5

The last paragraph of the DESCRIPTION was included as an APPLICATION USAGE note in previous issues.

Issue 6

The SYNOPSIS is marked CX since the presence of this function in the **<signal.h>** header is an extension over the ISO C standard.

NAME

sigemptyset — initialize and empty a signal set

SYNOPSIS

```
CX      #include <signal.h>
int sigemptyset(sigset_t *set);
```

DESCRIPTION

The *sigemptyset()* function initializes the signal set pointed to by *set*, such that all signals defined in POSIX.1-2008 are excluded.

RETURN VALUE

Upon successful completion, *sigemptyset()* shall return 0; otherwise, it shall return -1 and set *errno* to indicate the error.

ERRORS

No errors are defined.

EXAMPLES

None.

APPLICATION USAGE

None.

RATIONALE

The implementation of the *sigemptyset()* (or *sigfillset()*) function could quite trivially clear (or set) all the bits in the signal set. Alternatively, it would be reasonable to initialize part of the structure, such as a version field, to permit binary-compatibility between releases where the size of the set varies. For such reasons, either *sigemptyset()* or *sigfillset()* must be called prior to any other use of the signal set, even if such use is read-only (for example, as an argument to *sigpending()*). This function is not intended for dynamic allocation.

The *sigfillset()* and *sigemptyset()* functions require that the resulting signal set include (or exclude) all the signals defined in this volume of POSIX.1-2008. Although it is outside the scope of this volume of POSIX.1-2008 to place this requirement on signals that are implemented as extensions, it is recommended that implementation-defined signals also be affected by these functions. However, there may be a good reason for a particular signal not to be affected. For example, blocking or ignoring an implementation-defined signal may have undesirable side-effects, whereas the default action for that signal is harmless. In such a case, it would be preferable for such a signal to be excluded from the signal set returned by *sigfillset()*.

In early proposals there was no distinction between invalid and unsupported signals (the names of optional signals that were not supported by an implementation were not defined by that implementation). The [EINVAL] error was thus specified as a required error for invalid signals. With that distinction, it is not necessary to require implementations of these functions to determine whether an optional signal is actually supported, as that could have a significant performance impact for little value. The error could have been required for invalid signals and optional for unsupported signals, but this seemed unnecessarily complex. Thus, the error is optional in both cases.

FUTURE DIRECTIONS

None.

SEE ALSO

Section 2.4 (on page 488), *pthread_sigmask()*, *sigaction()*, *sigaddset()*, *sigdelset()*, *sigfillset()*, *sigismember()*, *sigpending()*, *sigsuspend()*

XBD **<signal.h>**

CHANGE HISTORY

First released in Issue 3. Included for alignment with the POSIX.1-1988 standard.

Issue 6

The SYNOPSIS is marked CX since the presence of this function in the **<signal.h>** header is an extension over the ISO C standard.

62357 NAME

62358 sigfillset — initialize and fill a signal set

62359 SYNOPSIS

```
62360 CX #include <signal.h>  
62361 int sigfillset(sigset_t *set);
```

62362 DESCRIPTION

62363 The *sigfillset()* function shall initialize the signal set pointed to by *set*, such that all signals
62364 defined in this volume of POSIX.1-2008 are included.

62365 RETURN VALUE

62366 Upon successful completion, *sigfillset()* shall return 0; otherwise, it shall return -1 and set *errno*
62367 to indicate the error.

62368 ERRORS

62369 No errors are defined.

62370 EXAMPLES

62371 None.

62372 APPLICATION USAGE

62373 None.

62374 RATIONALE

62375 Refer to *sigemptyset()* (on page 1941).

62376 FUTURE DIRECTIONS

62377 None.

62378 SEE ALSO

62379 Section 2.4 (on page 488), *pthread_sigmask()*, *sigaction()*, *sigaddset()*, *sigdelset()*, *sigemptyset()*,
62380 *sigismember()*, *sigpending()*, *sigsuspend()*

62381 XBD **<signal.h>**

62382 CHANGE HISTORY

62383 First released in Issue 3. Included for alignment with the POSIX.1-1988 standard.

62384 Issue 6

62385 The SYNOPSIS is marked CX since the presence of this function in the **<signal.h>** header is an
62386 extension over the ISO C standard.

NAME

sighold, sigignore, sigpause, sigrelse, sigset — signal management

SYNOPSIS

```
#include <signal.h>

int sighold(int sig);
int sigignore(int sig);
int sigpause(int sig);
int sigrelse(int sig);
void (*sigset(int sig, void (*disp)(int)))(int);
```

DESCRIPTION

Use of any of these functions is unspecified in a multi-threaded process.

The *sighold()*, *sigignore()*, *sigpause()*, *sigrelse()*, and *sigset()* functions provide simplified signal management.

The *sigset()* function shall modify signal dispositions. The *sig* argument specifies the signal, which may be any signal except SIGKILL and SIGSTOP. The *disp* argument specifies the signal's disposition, which may be SIG_DFL, SIG_IGN, or the address of a signal handler. If *sigset()* is used, and *disp* is the address of a signal handler, the system shall add *sig* to the signal mask of the calling process before executing the signal handler; when the signal handler returns, the system shall restore the signal mask of the calling process to its state prior to the delivery of the signal. In addition, if *sigset()* is used, and *disp* is equal to SIG_HOLD, *sig* shall be added to the signal mask of the calling process and *sig*'s disposition shall remain unchanged. If *sigset()* is used, and *disp* is not equal to SIG_HOLD, *sig* shall be removed from the signal mask of the calling process.

The *sighold()* function shall add *sig* to the signal mask of the calling process.

The *sigrelse()* function shall remove *sig* from the signal mask of the calling process.

The *sigignore()* function shall set the disposition of *sig* to SIG_IGN.

The *sigpause()* function shall remove *sig* from the signal mask of the calling process and suspend the calling process until a signal is received. The *sigpause()* function shall restore the signal mask of the process to its original state before returning.

If the action for the SIGCHLD signal is set to SIG_IGN, child processes of the calling processes shall not be transformed into zombie processes when they terminate. If the calling process subsequently waits for its children, and the process has no unwaited-for children that were transformed into zombie processes, it shall block until all of its children terminate, and *wait()*, *waitid()*, and *waitpid()* shall fail and set *errno* to [ECHILD].

RETURN VALUE

Upon successful completion, *sigset()* shall return SIG_HOLD if the signal had been blocked and the signal's previous disposition if it had not been blocked. Otherwise, SIG_ERR shall be returned and *errno* set to indicate the error.

The *sigpause()* function shall suspend execution of the thread until a signal is received, whereupon it shall return -1 and set *errno* to [EINTR].

For all other functions, upon successful completion, 0 shall be returned. Otherwise, -1 shall be returned and *errno* set to indicate the error.

ERRORS

These functions shall fail if:

[EINVAL] The *sig* argument is an illegal signal number.

The *sigset*() and *sigignore*() functions shall fail if:

[EINVAL] An attempt is made to catch a signal that cannot be caught, or to ignore a signal that cannot be ignored.

EXAMPLES

None.

APPLICATION USAGE

The *sigaction*() function provides a more comprehensive and reliable mechanism for controlling signals; new applications should use the *sigaction*() function instead of the obsolescent *sigset*() function.

The *sighold*() function, in conjunction with *sigrelse*() or *sigpause*(), may be used to establish critical regions of code that require the delivery of a signal to be temporarily deferred. For broader portability, the *pthread_sigmask*() or *sigprocmask*() functions should be used instead of the obsolescent *sighold*() and *sigrelse*() functions.

For broader portability, the *sigsuspend*() function should be used instead of the obsolescent *sigpause*() function.

RATIONALE

Each of these historic functions has a direct analog in the other functions which are required to be per-thread and thread-safe (aside from *sigprocmask*(), which is replaced by *pthread_sigmask*()). The *sigset*() function can be implemented as a simple wrapper for *sigaction*(). The *sighold*() function is equivalent to *sigprocmask*() or *pthread_sigmask*() with SIG_BLOCK set. The *sigignore*() function is equivalent to *sigaction*() with SIG_IGN set. The *sigpause*() function is equivalent to *sigsuspend*(). The *sigrelse*() function is equivalent to *sigprocmask*() or *pthread_sigmask*() with SIG_UNBLOCK set.

FUTURE DIRECTIONS

These functions may be removed in a future version.

SEE ALSO

Section 2.4 (on page 488), *exec*, *pause*(), *pthread_sigmask*(), *sigaction*(), *signal*(), *sigsuspend*(), *wait*(), *waitid*()

XBD <signal.h>

CHANGE HISTORY

First released in Issue 4, Version 2.

Issue 5

Moved from X/OPEN UNIX extension to BASE.

The DESCRIPTION is updated to indicate that the *sigpause*() function restores the signal mask of the process to its original state before returning.

The RETURN VALUE section is updated to indicate that the *sigpause*() function suspends execution of the process until a signal is received, whereupon it returns -1 and sets *errno* to [EINTR].

62470 Issue 6

62471 The normative text is updated to avoid use of the term “must” for application requirements.

62472 References to the *wait3()* function are removed.

62473 The XSI functions are split out into their own reference page.

62474 Issue 7

62475 SD5-XSH-ERN-113 and SD5-XSH-ERN-42 are applied, marking these functions obsolescent and
62476 updating the APPLICATION USAGE and RATIONALE sections.

NAME

siginterrupt — allow signals to interrupt functions

SYNOPSIS

```
#include <signal.h>
```

```
int siginterrupt(int sig, int flag);
```

DESCRIPTION

The *siginterrupt()* function shall change the restart behavior when a function is interrupted by the specified signal. The function *siginterrupt(sig, flag)* has an effect as if implemented as:

```
int siginterrupt(int sig, int flag) {
    int ret;
    struct sigaction act;

    (void) sigaction(sig, NULL, &act);
    if (flag)
        act.sa_flags &= ~SA_RESTART;
    else
        act.sa_flags |= SA_RESTART;
    ret = sigaction(sig, &act, NULL);
    return ret;
}
```

RETURN VALUE

Upon successful completion, *siginterrupt()* shall return 0; otherwise, -1 shall be returned and *errno* set to indicate the error.

ERRORS

The *siginterrupt()* function shall fail if:

[EINVAL] The *sig* argument is not a valid signal number.

EXAMPLES

None.

APPLICATION USAGE

The *siginterrupt()* function supports programs written to historical system interfaces. Applications should use the *sigaction()* with the SA_RESTART flag instead of the obsolescent *siginterrupt()* function.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

[Section 2.4](#) (on page 488), *sigaction()*

XBD [<signal.h>](#)

CHANGE HISTORY

First released in Issue 4, Version 2.

62517 **Issue 5**

62518 Moved from X/OPEN UNIX extension to BASE.

62519 **Issue 6**

62520 IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/59 is applied, correcting the declaration in
62521 the sample implementation given in the DESCRIPTION.

62522 **Issue 7**

62523 The *siginterrupt()* function is marked obsolescent.

62524 **NAME**

62525 sigismember — test for a signal in a signal set

62526 **SYNOPSIS**

```
62527 CX    #include <signal.h>
62528    int sigismember(const sigset_t *set, int signo);
```

62529 **DESCRIPTION**

62530 The *sigismember()* function shall test whether the signal specified by *signo* is a member of the set
 62531 pointed to by *set*.

62532 Applications should call either *sigemptyset()* or *sigfillset()* at least once for each object of type
 62533 **sigset_t** prior to any other use of that object. If such an object is not initialized in this way, but is
 62534 nonetheless supplied as an argument to any of *pthread_sigmask()*, *sigaction()*, *sigaddset()*,
 62535 *sigdelset()*, *sigismember()*, *sigpending()*, *sigprocmask()*, *sigsuspend()*, *sigtimedwait()*, *sigwait()*, or
 62536 *sigwaitinfo()*, the results are undefined.

62537 **RETURN VALUE**

62538 Upon successful completion, *sigismember()* shall return 1 if the specified signal is a member of
 62539 the specified set, or 0 if it is not. Otherwise, it shall return -1 and set *errno* to indicate the error.

62540 **ERRORS**

62541 The *sigismember()* function may fail if:

62542	[EINVAL]	The <i>signo</i> argument is not a valid signal number, or is an unsupported signal number.
62543		

62544 **EXAMPLES**

62545 None.

62546 **APPLICATION USAGE**

62547 None.

62548 **RATIONALE**

62549 None.

62550 **FUTURE DIRECTIONS**

62551 None.

62552 **SEE ALSO**

62553 [Section 2.4](#) (on page 488), *pthread_sigmask()*, *sigaction()*, *sigaddset()*, *sigdelset()*, *sigfillset()*,
 62554 *sigemptyset()*, *sigpending()*, *sigsuspend()*

62555 XBD [<signal.h>](#)

62556 **CHANGE HISTORY**

62557 First released in Issue 3. Included for alignment with the POSIX.1-1988 standard.

62558 **Issue 5**

62559 The last paragraph of the DESCRIPTION was included as an APPLICATION USAGE note in
 62560 previous issues.

62561 **Issue 6**

62562 The SYNOPSIS is marked CX since the presence of this function in the **<signal.h>** header is an
 62563 extension over the ISO C standard.

62564 **NAME**

62565 siglongjmp — non-local goto with signal handling

62566 **SYNOPSIS**

```
62567 CX    #include <setjmp.h>
62568      void siglongjmp(sigjmp_buf env, int val);
```

62569 **DESCRIPTION**62570 The *siglongjmp()* function shall be equivalent to the *longjmp()* function, except as follows:

- 62571 • References to *setjmp()* shall be equivalent to *sigsetjmp()*.
- 62572 • The *siglongjmp()* function shall restore the saved signal mask if and only if the *env*
- 62573 argument was initialized by a call to *sigsetjmp()* with a non-zero *savemask* argument.

62574 **RETURN VALUE**

62575 After *siglongjmp()* is completed, program execution shall continue as if the corresponding

62576 invocation of *sigsetjmp()* had just returned the value specified by *val*. The *siglongjmp()* function

62577 shall not cause *sigsetjmp()* to return 0; if *val* is 0, *sigsetjmp()* shall return the value 1.

62578 **ERRORS**

62579 No errors are defined.

62580 **EXAMPLES**

62581 None.

62582 **APPLICATION USAGE**

62583 The distinction between *setjmp()* or *longjmp()* and *sigsetjmp()* or *siglongjmp()* is only significant

62584 for programs which use *sigaction()*, *sigprocmask()*, or *sigsuspend()*.

62585 **RATIONALE**

62586 None.

62587 **FUTURE DIRECTIONS**

62588 None.

62589 **SEE ALSO**62590 *longjmp()*, *pthread_sigmask()*, *setjmp()*, *sigsetjmp()*, *sigsuspend()*

62591 XBD <setjmp.h>

62592 **CHANGE HISTORY**

62593 First released in Issue 3. Included for alignment with the ISO POSIX-1 standard.

62594 **Issue 5**

62595 The DESCRIPTION is updated for alignment with the POSIX Threads Extension.

62596 **Issue 6**62597 The DESCRIPTION is rewritten in terms of *longjmp()*.

62598 The SYNOPSIS is marked CX since the presence of this function in the <setjmp.h> header is an

62599 extension over the ISO C standard.

62600 **NAME**

62601 signal — signal management

62602 **SYNOPSIS**

62603 #include <signal.h>

62604 void (*signal(int sig, void (*func)(int)))(int);

62605 **DESCRIPTION**

62606 CX The functionality described on this reference page is aligned with the ISO C standard. Any
 62607 conflict between the requirements described here and the ISO C standard is unintentional. This
 62608 volume of POSIX.1-2008 defers to the ISO C standard.

62609 Use of this function is unspecified in a multi-threaded process.

62610 The *signal()* function chooses one of three ways in which receipt of the signal number *sig* is to be
 62611 subsequently handled. If the value of *func* is *SIG_DFL*, default handling for that signal shall
 62612 occur. If the value of *func* is *SIG_IGN*, the signal shall be ignored. Otherwise, the application
 62613 shall ensure that *func* points to a function to be called when that signal occurs. An invocation of
 62614 such a function because of a signal, or (recursively) of any further functions called by that
 62615 invocation (other than functions in the standard library), is called a “signal handler”.

62616 When a signal occurs, and *func* points to a function, it is implementation-defined whether the
 62617 equivalent of a:

62618 signal(*sig*, *SIG_DFL*);

62619 is executed or the implementation prevents some implementation-defined set of signals (at least
 62620 including *sig*) from occurring until the current signal handling has completed. (If the value of *sig*
 62621 is *SIGILL*, the implementation may alternatively define that no action is taken.) Next the
 62622 equivalent of:

62623 (*func)(*sig*);

62624 is executed. If and when the function returns, if the value of *sig* was *SIGFPE*, *SIGILL*, or
 62625 *SIGSEGV* or any other implementation-defined value corresponding to a computational
 62626 exception, the behavior is undefined. Otherwise, the program shall resume execution at the
 62627 point it was interrupted. The ISO C standard places a restriction on applications relating to the
 62628 CX use of *raise()* from signal handlers. This restriction does not apply to POSIX applications, as
 62629 POSIX.1-2008 requires *raise()* to be async-signal-safe (see [Section 2.4.3](#), on page 490).

62630 CX If the process is multi-threaded, or if the process is single-threaded and a signal handler is
 62631 executed other than as the result of:

- 62632 CX • The process calling *abort()*, *raise()*, *kill()*, *pthread_kill()*, or *sigqueue()* to generate a signal
 62633 that is not blocked
- 62634 CX • A pending signal being unblocked and being delivered before the call that unblocked it
 62635 returns

62636 CX the behavior is undefined if the signal handler refers to any object other than *errno* with static
 62637 storage duration other than by assigning a value to an object declared as *volatile sig_atomic_t*,
 62638 CX or if the signal handler calls any function defined in this standard other than one of the
 62639 functions listed in [Section 2.4](#) (on page 488).

62640 At program start-up, the equivalent of:

62641 signal(*sig*, *SIG_IGN*);

62642 is executed for some signals, and the equivalent of:

```
62643      signal(sig, SIG_DFL);
```

62644 CX is executed for all other signals (see *exec*).

62645 The *signal()* function shall not change the setting of *errno* if successful.

62646 RETURN VALUE

62647 If the request can be honored, *signal()* shall return the value of *func* for the most recent call to
62648 *signal()* for the specified signal *sig*. Otherwise, SIG_ERR shall be returned and a positive value
62649 shall be stored in *errno*.

62650 **ERRORS**

62651 The *signal()* function shall fail if:

62652	CX	[EINVAL]	The <i>sig</i> argument is not a valid signal number or an attempt is made to catch a
62653			signal that cannot be caught or ignore a signal that cannot be ignored.

62654 The *signal()* function may fail if:

62655	CX	[EINVAL]	An attempt was made to set the action to SIG_DFL for a signal that cannot be
62656			caught or ignored (or both).

62657 **EXAMPLES**

62658 None.

62659 APPLICATION USAGE

62660 The *sigaction()* function provides a more comprehensive and reliable mechanism for controlling
62661 signals; new applications should use *sigaction()* rather than *signal()*.

62662 RATIONALE

62663 None.

62664 FUTURE DIRECTIONS

62665 None.

62666 **SEE ALSO**

62667 Section 2.4 (on page 488), *exec*, *pause()*, *raise()*, *sigaction()*, *sigsuspend()*, *waitid()*

```
62668 XBD <signal.h>
```

62669 CHANGE HISTORY

62670 First released in Issue 1. Derived from Issue 1 of the SVID.

62671 **Issue 5**

62672 Moved from X/OPEN UNIX extension to BASE.

62673 The DESCRIPTION is updated to indicate that the *sigpause()* function restores the signal mask of
62674 the process to its original state before returning.

62675 The RETURN VALUE section is updated to indicate that the *sigpause()* function suspends
62676 execution of the process until a signal is received, whereupon it returns `-1` and sets *errno* to
62677 `[EINTR]`.

62678 **Issue 6**

62679 Extensions beyond the ISO C standard are marked.

62680 The normative text is updated to avoid use of the term “must” for application requirements.

62681 The DESCRIPTION is updated for alignment with the ISO/IEC 9899:1999 standard.

62682 References to the *wait3()* function are removed.

62683 The *sighold()*, *sigignore()*, *sigrelse()*, and *sigset()* functions are split out onto their own reference

62684	page.
62685	Issue 7
62686	POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0580 [275], XSH/TC1-2008/0581 [66], +
62687	and XSH/TC1-2008/0582 [105] are applied.

62688 **NAME**

62689 signbit — test sign

62690 **SYNOPSIS**

62691 #include <math.h>

62692 int signbit(real-floating x);

62693 **DESCRIPTION**

62694 CX The functionality described on this reference page is aligned with the ISO C standard. Any
62695 conflict between the requirements described here and the ISO C standard is unintentional. This
62696 volume of POSIX.1-2008 defers to the ISO C standard.

62697 The *signbit()* macro shall determine whether the sign of its argument value is negative. NaNs,
62698 zeros, and infinities have a sign bit.

62699 **RETURN VALUE**

62700 The *signbit()* macro shall return a non-zero value if and only if the sign of its argument value is
62701 negative.

62702 **ERRORS**

62703 No errors are defined.

62704 **EXAMPLES**

62705 None.

62706 **APPLICATION USAGE**

62707 None.

62708 **RATIONALE**

62709 None.

62710 **FUTURE DIRECTIONS**

62711 None.

62712 **SEE ALSO**62713 *fpclassify(), isfinite(), isinf(), isnan(), isnormal()*

62714 XBD <math.h>

62715 **CHANGE HISTORY**

62716 First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

62717 **NAME**

62718 signgam — log gamma function

62719 **SYNOPSIS**

```
62720 XSI      #include <math.h>
62721          extern int signgam;
```

62722 **DESCRIPTION**62723 Refer to *lgamma()*.

NAME

sigpause — remove a signal from the signal mask and suspend the thread

SYNOPSIS

```
OB XSI #include <signal.h>
int sigpause(int sig);
```

DESCRIPTION

Refer to *sighold()*.

62731 **NAME**

62732 sigpending — examine pending signals

62733 **SYNOPSIS**

```
62734 CX    #include <signal.h>
62735    int sigpending(sigset_t *set);
```

62736 **DESCRIPTION**

62737 The *sigpending()* function shall store, in the location referenced by the *set* argument, the set of
 62738 signals that are blocked from delivery to the calling thread and that are pending on the process
 62739 or the calling thread.

62740 **RETURN VALUE**

62741 Upon successful completion, *sigpending()* shall return 0; otherwise, -1 shall be returned and
 62742 *errno* set to indicate the error.

62743 **ERRORS**

62744 No errors are defined.

62745 **EXAMPLES**

62746 None.

62747 **APPLICATION USAGE**

62748 None.

62749 **RATIONALE**

62750 None.

62751 **FUTURE DIRECTIONS**

62752 None.

62753 **SEE ALSO**62754 *exec*, *pthread_sigmask()*, *sigaddset()*, *sigdelset()*, *sigemptyset()*, *sigfillset()*, *sigismember()*62755 XBD [<signal.h>](#)62756 **CHANGE HISTORY**

62757 First released in Issue 3.

62758 **Issue 5**

62759 The DESCRIPTION is updated for alignment with the POSIX Threads Extension.

62760 **Issue 6**

62761 The SYNOPSIS is marked CX since the presence of this function in the **<signal.h>** header is an
 62762 extension over the ISO C standard.

62763 NAME

62764 sigprocmask — examine and change blocked signals

62765 SYNOPSIS

```
62766 CX      #include <signal.h>
62767          int sigprocmask(int how, const sigset_t *restrict set,
62768                          sigset_t *restrict oset);
```

62769 DESCRIPTION

62770 Refer to *pthread_sigmask()*.

62771 **NAME**

62772 sigqueue — queue a signal to a process

62773 **SYNOPSIS**

```
62774 CX #include <signal.h>
62775 int sigqueue(pid_t pid, int signo, const union sigval value);
```

62776 **DESCRIPTION**

62777 The *sigqueue()* function shall cause the signal specified by *signo* to be sent with the value
 62778 specified by *value* to the process specified by *pid*. If *signo* is zero (the null signal), error checking
 62779 is performed but no signal is actually sent. The null signal can be used to check the validity of
 62780 *pid*.

62781 The conditions required for a process to have permission to queue a signal to another process are
 62782 the same as for the *kill()* function.

62783 The *sigqueue()* function shall return immediately. If SA_SIGINFO is set for *signo* and if the
 62784 resources were available to queue the signal, the signal shall be queued and sent to the receiving
 62785 process. If SA_SIGINFO is not set for *signo*, then *signo* shall be sent at least once to the receiving
 62786 process; it is unspecified whether *value* shall be sent to the receiving process as a result of this
 62787 call.

62788 If the value of *pid* causes *signo* to be generated for the sending process, and if *signo* is not blocked
 62789 for the calling thread and if no other thread has *signo* unblocked or is waiting in a *sigwait()*
 62790 function for *signo*, either *signo* or at least the pending, unblocked signal shall be delivered to the
 62791 calling thread before the *sigqueue()* function returns. Should any multiple pending signals in the
 62792 range SIGRTMIN to SIGRTMAX be selected for delivery, it shall be the lowest numbered one.
 62793 The selection order between realtime and non-realtime signals, or between multiple pending
 62794 non-realtime signals, is unspecified.

62795 **RETURN VALUE**

62796 Upon successful completion, the specified signal shall have been queued, and the *sigqueue()*
 62797 function shall return a value of zero. Otherwise, the function shall return a value of -1 and set
 62798 *errno* to indicate the error.

62799 **ERRORS**

62800 The *sigqueue()* function shall fail if:

62801	[EAGAIN]	No resources are available to queue the signal. The process has already queued {SIGQUEUE_MAX} signals that are still pending at the receiver(s), or a system-wide resource limit has been exceeded.
62802		
62803		
62804	[EINVAL]	The value of the <i>signo</i> argument is an invalid or unsupported signal number.
62805	[EPERM]	The process does not have appropriate privileges to send the signal to the receiving process.
62806		
62807	[ESRCH]	The process <i>pid</i> does not exist.

EXAMPLES

None.

APPLICATION USAGE

None.

RATIONALE

The *sigqueue()* function allows an application to queue a realtime signal to itself or to another process, specifying the application-defined value. This is common practice in realtime applications on existing realtime systems. It was felt that specifying another function in the *sig...* name space already carved out for signals was preferable to extending the interface to *kill()*.

Such a function became necessary when the put/get event function of the message queues was removed. It should be noted that the *sigqueue()* function implies reduced performance in a security-conscious implementation as the access permissions between the sender and receiver have to be checked on each send when the *pid* is resolved into a target process. Such access checks were necessary only at message queue open in the previous interface.

The standard developers required that *sigqueue()* have the same semantics with respect to the null signal as *kill()*, and that the same permission checking be used. But because of the difficulty of implementing the “broadcast” semantic of *kill()* (for example, to process groups) and the interaction with resource allocation, this semantic was not adopted. The *sigqueue()* function queues a signal to a single process specified by the *pid* argument.

The *sigqueue()* function can fail if the system has insufficient resources to queue the signal. An explicit limit on the number of queued signals that a process could send was introduced. While the limit is “per-sender”, this volume of POSIX.1-2008 does not specify that the resources be part of the state of the sender. This would require either that the sender be maintained after exit until all signals that it had sent to other processes were handled or that all such signals that had not yet been acted upon be removed from the queue(s) of the receivers. This volume of POSIX.1-2008 does not preclude this behavior, but an implementation that allocated queuing resources from a system-wide pool (with per-sender limits) and that leaves queued signals pending after the sender exits is also permitted.

FUTURE DIRECTIONS

None.

SEE ALSO

[Section 2.8.1](#) (on page 502)

XBD [<signal.h>](#)

CHANGE HISTORY

First released in Issue 5. Included for alignment with the POSIX Realtime Extension and the POSIX Threads Extension.

Issue 6

The *sigqueue()* function is marked as part of the Realtime Signals Extension option.

The [ENOSYS] error condition has been removed as stubs need not be provided if an implementation does not support the Realtime Signals Extension option.

Issue 7

The *sigqueue()* function is moved from the Realtime Signals Extension option to the Base.

62851 **NAME**

62852 sigrelse, sigset — signal management

62853 **SYNOPSIS**

```
62854 OB XSI #include <signal.h>
62855 int sigrelse(int sig);
62856 void (*sigset(int sig, void (*disp)(int)))(int);
```

62857 **DESCRIPTION**62858 Refer to *sighold()*.

NAME

sigsetjmp — set jump point for a non-local goto

SYNOPSIS

```
CX      #include <setjmp.h>
int sigsetjmp(sigjmp_buf env, int savemask);
```

DESCRIPTION

The *sigsetjmp()* function shall be equivalent to the *setjmp()* function, except as follows:

- References to *setjmp()* are equivalent to *sigsetjmp()*.
- References to *longjmp()* are equivalent to *siglongjmp()*.
- If the value of the *savemask* argument is not 0, *sigsetjmp()* shall also save the current signal mask of the calling thread as part of the calling environment.

RETURN VALUE

If the return is from a successful direct invocation, *sigsetjmp()* shall return 0. If the return is from a call to *siglongjmp()*, *sigsetjmp()* shall return a non-zero value.

ERRORS

No errors are defined.

EXAMPLES

None.

APPLICATION USAGE

The distinction between *setjmp()/longjmp()* and *sigsetjmp()/siglongjmp()* is only significant for programs which use *sigaction()*, *sigprocmask()*, or *sigsuspend()*.

Note that since this function is defined in terms of *setjmp()*, if *savemask* is zero, it is unspecified whether the signal mask is saved.

RATIONALE

The ISO C standard specifies various restrictions on the usage of the *setjmp()* macro in order to permit implementors to recognize the name in the compiler and not implement an actual function. These same restrictions apply to the *sigsetjmp()* macro.

There are processors that cannot easily support these calls, but this was not considered a sufficient reason to exclude them.

4.2 BSD, 4.3 BSD, and XSI-conformant systems provide functions named *_setjmp()* and *_longjmp()* that, together with *setjmp()* and *longjmp()*, provide the same functionality as *sigsetjmp()* and *siglongjmp()*. On those systems, *setjmp()* and *longjmp()* save and restore signal masks, while *_setjmp()* and *_longjmp()* do not. On System V Release 3 and in corresponding issues of the SVID, *setjmp()* and *longjmp()* are explicitly defined not to save and restore signal masks. In order to permit existing practice in both cases, the relation of *setjmp()* and *longjmp()* to signal masks is not specified, and a new set of functions is defined instead.

The *longjmp()* and *siglongjmp()* functions operate as in the previous issue provided the matching *setjmp()* or *sigsetjmp()* has been performed in the same thread. Non-local jumps into contexts saved by other threads would be at best a questionable practice and were not considered worthy of standardization.

62899 FUTURE DIRECTIONS

62900 None.

62901 SEE ALSO

62902 *pthread_sigmask()*, *siglongjmp()*, *signal()*, *sigsuspend()*

62903 XBD **<setjmp.h>**

62904 CHANGE HISTORY

62905 First released in Issue 3. Included for alignment with the POSIX.1-1988 standard.

62906 Issue 5

62907 The DESCRIPTION is updated for alignment with the POSIX Threads Extension.

62908 Issue 6

62909 The DESCRIPTION is reworded in terms of *setjmp()*.

62910 The SYNOPSIS is marked CX since the presence of this function in the **<setjmp.h>** header is an
62911 extension over the ISO C standard.

62912 **NAME**

62913 sigsuspend — wait for a signal

62914 **SYNOPSIS**

```
62915 CX      #include <signal.h>
62916      int sigsuspend(const sigset_t *sigmask);
```

62917 **DESCRIPTION**

62918 The *sigsuspend()* function shall replace the current signal mask of the calling thread with the set
 62919 of signals pointed to by *sigmask* and then suspend the thread until delivery of a signal whose
 62920 action is either to execute a signal-catching function or to terminate the process. This shall not
 62921 cause any other signals that may have been pending on the process to become pending on the
 62922 thread.

62923 If the action is to terminate the process then *sigsuspend()* shall never return. If the action is to
 62924 execute a signal-catching function, then *sigsuspend()* shall return after the signal-catching
 62925 function returns, with the signal mask restored to the set that existed prior to the *sigsuspend()*
 62926 call.

62927 It is not possible to block signals that cannot be ignored. This is enforced by the system without
 62928 causing an error to be indicated.

62929 **RETURN VALUE**

62930 Since *sigsuspend()* suspends thread execution indefinitely, there is no successful completion
 62931 return value. If a return occurs, *-1* shall be returned and *errno* set to indicate the error.

62932 **ERRORS**

62933 The *sigsuspend()* function shall fail if:

62934 [EINTR] A signal is caught by the calling process and control is returned from the
 62935 signal-catching function.

62936 **EXAMPLES**

62937 None.

62938 **APPLICATION USAGE**

62939 Normally, at the beginning of a critical code section, a specified set of signals is blocked using
 62940 the *sigprocmask()* function. When the thread has completed the critical section and needs to wait
 62941 for the previously blocked signal(s), it pauses by calling *sigsuspend()* with the mask that was
 62942 returned by the *sigprocmask()* call.

62943 **RATIONALE**

62944 Code which wants to avoid the ambiguity of the signal mask for thread cancellation handlers
 62945 can install an additional cancellation handler which resets the signal mask to the expected value.

```
62946 void cleanup(void *arg)
62947 {
62948     sigset_t *ss = (sigset_t *) arg;
62949     pthread_sigmask(SIG_SETMASK, ss, NULL);
62950 }
62951 int call_sigsuspend(const sigset_t *mask)
62952 {
62953     sigset_t oldmask;
62954     int result;
62955     pthread_sigmask(SIG_SETMASK, NULL, &oldmask);
62956     pthread_cleanup_push(cleanup, &oldmask);
```

```
62957         result = sigsuspend(sigmask);
62958         pthread_cleanup_pop(0);
62959         return result;
62960     }
```

62961 FUTURE DIRECTIONS

62962 None.

62963 SEE ALSO

62964 [Section 2.4](#) (on page 488), [pause\(\)](#), [sigaction\(\)](#), [sigaddset\(\)](#), [sigdelset\(\)](#), [sigemptyset\(\)](#), [sigfillset\(\)](#)

62965 XBD [<signal.h>](#)

62966 CHANGE HISTORY

62967 First released in Issue 3. Included for alignment with the POSIX.1-1988 standard.

62968 Issue 5

62969 The DESCRIPTION is updated for alignment with the POSIX Threads Extension.

62970 Issue 6

62971 The text in the RETURN VALUE section has been changed from “suspends process execution”
62972 to “suspends thread execution”. This reflects IEEE PASC Interpretation 1003.1c #40.

62973 Text in the APPLICATION USAGE section has been replaced.

62974 The SYNOPSIS is marked CX since the presence of this function in the [<signal.h>](#) header is an
62975 extension over the ISO C standard.

62976 Issue 7

62977 SD5-XSH-ERN-122 is applied, adding the example code in the RATIONALE.

NAME

sigtimedwait, sigwaitinfo — wait for queued signals

SYNOPSIS

```

#include <signal.h>

int sigtimedwait(const sigset_t *restrict set,
                 siginfo_t *restrict info,
                 const struct timespec *restrict timeout);
int sigwaitinfo(const sigset_t *restrict set,
                 siginfo_t *restrict info);

```

DESCRIPTION

The *sigtimedwait()* function shall be equivalent to *sigwaitinfo()* except that if none of the signals specified by *set* are pending, *sigtimedwait()* shall wait for the time interval specified in the **timespec** structure referenced by *timeout*. If the **timespec** structure pointed to by *timeout* is zero-valued and if none of the signals specified by *set* are pending, then *sigtimedwait()* shall return immediately with an error. If *timeout* is the null pointer, the behavior is unspecified. If the Monotonic Clock option is supported, the **CLOCK_MONOTONIC** clock shall be used to measure the time interval specified by the *timeout* argument.

The *sigwaitinfo()* function selects the pending signal from the set specified by *set*. Should any of multiple pending signals in the range SIGRTMIN to SIGRTMAX be selected, it shall be the lowest numbered one. The selection order between realtime and non-realtime signals, or between multiple pending non-realtime signals, is unspecified. If no signal in *set* is pending at the time of the call, the calling thread shall be suspended until one or more signals in *set* become pending or until it is interrupted by an unblocked, caught signal.

The *sigwaitinfo()* function shall be equivalent to the *sigwait()* function if the *info* argument is NULL. If the *info* argument is non-NULL, the *sigwaitinfo()* function shall be equivalent to *sigwait()*, except that the selected signal number shall be stored in the *si_signo* member, and the cause of the signal shall be stored in the *si_code* member. If any value is queued to the selected signal, the first such queued value shall be dequeued and, if the *info* argument is non-NULL, the value shall be stored in the *si_value* member of *info*. The system resource used to queue the signal shall be released and returned to the system for other use. If no value is queued, the content of the *si_value* member is undefined. If no further signals are queued for the selected signal, the pending indication for that signal shall be reset.

RETURN VALUE

Upon successful completion (that is, one of the signals specified by *set* is pending or is generated) *sigwaitinfo()* and *sigtimedwait()* shall return the selected signal number. Otherwise, the function shall return a value of -1 and set *errno* to indicate the error.

ERRORS

The *sigtimedwait()* function shall fail if:

[EAGAIN] No signal specified by *set* was generated within the specified timeout period.

The *sigtimedwait()* and *sigwaitinfo()* functions may fail if:

[EINTR] The wait was interrupted by an unblocked, caught signal. It shall be documented in system documentation whether this error causes these functions to fail.

63021 The *sigtimedwait()* function may also fail if:
 63022 [EINVAL] The *timeout* argument specified a *tv_nsec* value less than zero or greater than
 63023 or equal to 1 000 million.

63024 An implementation should only check for this error if no signal is pending in *set* and it is
 63025 necessary to wait.

63026 EXAMPLES

63027 None.

63028 APPLICATION USAGE

63029 The *sigtimedwait()* function times out and returns an [EAGAIN] error. Application developers
 63030 should note that this is inconsistent with other functions such as *pthread_cond_timedwait()* that
 63031 return [ETIMEDOUT].

63032 Note that in order to ensure that generated signals are queued and signal values passed to +
 63033 *sigqueue()* are available in *si_value*, applications which use *sigwaitinfo()* or *sigtimedwait()* need to +
 63034 set the SA_SIGINFO flag for each signal in the set (see [Section 2.4](#), on page 488). This means +
 63035 setting each signal to be handled by a three-argument signal-catching function, even if the +
 63036 handler will never be called. It is not possible (portably) to set a signal handler to SIG_DFL +
 63037 while setting the SA_SIGINFO flag, because assigning to the *sa_handler* member of **struct** +
 63038 **sigaction** instead of the *sa_sigaction* member would result in undefined behavior, and SIG_DFL +
 63039 need not be assignment-compatible with *sa_sigaction*. Even if an assignment of SIG_DFL to +
 63040 *sa_sigaction* is accepted by the compiler, the implementation need not treat this value as special— +
 63041 it could just be taken as the address of a signal-catching function.

63042 RATIONALE

63043 Existing programming practice on realtime systems uses the ability to pause waiting for a
 63044 selected set of events and handle the first event that occurs in-line instead of in a signal-handling
 63045 function. This allows applications to be written in an event-directed style similar to a state
 63046 machine. This style of programming is useful for largescale transaction processing in which the
 63047 overall throughput of an application and the ability to clearly track states are more important
 63048 than the ability to minimize the response time of individual event handling.

63049 It is possible to construct a signal-waiting macro function out of the realtime signal function
 63050 mechanism defined in this volume of POSIX.1-2008. However, such a macro has to include the
 63051 definition of a generalized handler for all signals to be waited on. A significant portion of the
 63052 overhead of handler processing can be avoided if the signal-waiting function is provided by the
 63053 kernel. This volume of POSIX.1-2008 therefore provides two signal-waiting functions—one that
 63054 waits indefinitely and one with a timeout—as part of the overall realtime signal function
 63055 specification.

63056 The specification of a function with a timeout allows an application to be written that can be
 63057 broken out of a wait after a set period of time if no event has occurred. It was argued that setting
 63058 a timer event before the wait and recognizing the timer event in the wait would also implement
 63059 the same functionality, but at a lower performance level. Because of the performance
 63060 degradation associated with the user-level specification of a timer event and the subsequent
 63061 cancellation of that timer event after the wait completes for a valid event, and the complexity
 63062 associated with handling potential race conditions associated with the user-level method, the
 63063 separate function has been included.

63064 Note that the semantics of the *sigwaitinfo()* function are nearly identical to that of the *sigwait()*
 63065 function defined by this volume of POSIX.1-2008. The only difference is that *sigwaitinfo()* returns
 63066 the queued signal value in the *value* argument. The return of the queued value is required so that
 63067 applications can differentiate between multiple events queued to the same signal number.

The two distinct functions are being maintained because some implementations may choose to implement the POSIX Threads Extension functions and not implement the queued signals extensions. Note, though, that *sigwaitinfo()* does not return the queued value if the *value* argument is NULL, so the POSIX Threads Extension *sigwait()* function can be implemented as a macro on *sigwaitinfo()*.

The *sigtimedwait()* function was separated from the *sigwaitinfo()* function to address concerns regarding the overloading of the *timeout* pointer to indicate indefinite wait (no timeout), timed wait, and immediate return, and concerns regarding consistency with other functions where the conditional and timed waits were separate functions from the pure blocking function. The semantics of *sigtimedwait()* are specified such that *sigwaitinfo()* could be implemented as a macro with a null pointer for *timeout*.

The *sigwait* functions provide a synchronous mechanism for threads to wait for asynchronously-generated signals. One important question was how many threads that are suspended in a call to a *sigwait()* function for a signal should return from the call when the signal is sent. Four choices were considered:

1. Return an error for multiple simultaneous calls to *sigwait* functions for the same signal.
2. One or more threads return.
3. All waiting threads return.
4. Exactly one thread returns.

Prohibiting multiple calls to *sigwait()* for the same signal was felt to be overly restrictive. The “one or more” behavior made implementation of conforming packages easy at the expense of forcing POSIX threads clients to protect against multiple simultaneous calls to *sigwait()* in application code in order to achieve predictable behavior. There was concern that the “all waiting threads” behavior would result in “signal broadcast storms”, consuming excessive CPU resources by replicating the signals in the general case. Furthermore, no convincing examples could be presented that delivery to all was either simpler or more powerful than delivery to one.

Thus, the consensus was that exactly one thread that was suspended in a call to a *sigwait* function for a signal should return when that signal occurs. This is not an onerous restriction as:

- A multi-way signal wait can be built from the single-way wait.
- Signals should only be handled by application-level code, as library routines cannot guess what the application wants to do with signals generated for the entire process.
- Applications can thus arrange for a single thread to wait for any given signal and call any needed routines upon its arrival.

In an application that is using signals for interprocess communication, signal processing is typically done in one place. Alternatively, if the signal is being caught so that process cleanup can be done, the signal handler thread can call separate process cleanup routines for each portion of the application. Since the application main line started each portion of the application, it is at the right abstraction level to tell each portion of the application to clean up.

Certainly, there exist programming styles where it is logical to consider waiting for a single signal in multiple threads. A simple *sigwait_multiple()* routine can be constructed to achieve this goal. A possible implementation would be to have each *sigwait_multiple()* caller registered as having expressed interest in a set of signals. The caller then waits on a thread-specific condition variable. A single server thread calls a *sigwait()* function on the union of all registered signals. When the *sigwait()* function returns, the appropriate state is set and condition variables are broadcast. New *sigwait_multiple()* callers may cause the pending *sigwait()* call to be canceled

63113 and reissued in order to update the set of signals being waited for.

63114 FUTURE DIRECTIONS

63115 None.

63116 SEE ALSO

63117 [Section 2.4](#) (on page 488), [Section 2.8.1](#) (on page 502), [pause\(\)](#), [pthread_sigmask\(\)](#), [sigaction\(\)](#),
63118 [sigpending\(\)](#), [sigsuspend\(\)](#), [sigwait\(\)](#)

63119 XBD [<signal.h>](#), [<time.h>](#)

63120 CHANGE HISTORY

63121 First released in Issue 5. Included for alignment with the POSIX Realtime Extension and the
63122 POSIX Threads Extension.

63123 Issue 6

63124 These functions are marked as part of the Realtime Signals Extension option.

63125 The Open Group Corrigendum U035/3 is applied. The SYNOPSIS of the *sigwaitinfo()* function
63126 has been corrected so that the second argument is of type **siginfo_t** *.

63127 The [ENOSYS] error condition has been removed as stubs need not be provided if an
63128 implementation does not support the Realtime Signals Extension option.

63129 The DESCRIPTION is updated for alignment with IEEE Std 1003.1j-2000 by specifying that the
63130 CLOCK_MONOTONIC clock, if supported, is used to measure timeout intervals.

63131 The **restrict** keyword is added to the *sigtimedwait()* and *sigwaitinfo()* prototypes for alignment
63132 with the ISO/IEC 9899:1999 standard.

63133 IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/130 is applied, restoring wording in the
63134 RETURN VALUE section to that in the original base document (“An implementation should
63135 only check for this error if no signal is pending in *set* and it is necessary to wait”).

63136 Issue 7

63137 The *sigtimedwait()* and *sigwaitinfo()* functions are moved from the Realtime Signals Extension
63138 option to the Base.

63139 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0583 [392] is applied.

+

NAME

sigwait — wait for queued signals

SYNOPSIS

```
#include <signal.h>

int sigwait(const sigset_t *restrict set, int *restrict sig);
```

DESCRIPTION

The *sigwait()* function shall select a pending signal from *set*, atomically clear it from the system's set of pending signals, and return that signal number in the location referenced by *sig*. If prior to the call to *sigwait()* there are multiple pending instances of a single signal number, it is implementation-defined whether upon successful return there are any remaining pending signals for that signal number. If the implementation supports queued signals and there are multiple signals queued for the signal number selected, the first such queued signal shall cause a return from *sigwait()* and the remainder shall remain queued. If no signal in *set* is pending at the time of the call, the thread shall be suspended until one or more becomes pending. The signals defined by *set* shall have been blocked at the time of the call to *sigwait()*; otherwise, the behavior is undefined. The effect of *sigwait()* on the signal actions for the signals in *set* is unspecified.

If more than one thread is using *sigwait()* to wait for the same signal, no more than one of these threads shall return from *sigwait()* with the signal number. If more than a single thread is blocked in *sigwait()* for a signal when that signal is generated for the process, it is unspecified which of the waiting threads returns from *sigwait()*. If the signal is generated for a specific thread, as by *pthread_kill()*, only that thread shall return.

Should any of the multiple pending signals in the range SIGRTMIN to SIGRTMAX be selected, it shall be the lowest numbered one. The selection order between realtime and non-realtime signals, or between multiple pending non-realtime signals, is unspecified.

RETURN VALUE

Upon successful completion, *sigwait()* shall store the signal number of the received signal at the location referenced by *sig* and return zero. Otherwise, an error number shall be returned to indicate the error.

ERRORS

The *sigwait()* function may fail if:

[EINVAL] The *set* argument contains an invalid or unsupported signal number.

EXAMPLES

None.

APPLICATION USAGE

None.

RATIONALE

To provide a convenient way for a thread to wait for a signal, this volume of POSIX.1-2008 provides the *sigwait()* function. For most cases where a thread has to wait for a signal, the *sigwait()* function should be quite convenient, efficient, and adequate.

However, requests were made for a lower-level primitive than *sigwait()* and for semaphores that could be used by threads. After some consideration, threads were allowed to use semaphores and *sem_post()* was defined to be async-signal-safe.

In summary, when it is necessary for code run in response to an asynchronous signal to notify a thread, *sigwait()* should be used to handle the signal. Alternatively, if the implementation provides semaphores, they also can be used, either following *sigwait()* or from within a signal

63185 handling routine previously registered with *sigaction()*.

63186 FUTURE DIRECTIONS

63187 None.

63188 SEE ALSO

63189 [Section 2.4](#) (on page 488), [Section 2.8.1](#) (on page 502), [pause\(\)](#), [pthread_sigmask\(\)](#), [sigaction\(\)](#),
63190 [sigpending\(\)](#), [sigsuspend\(\)](#), [sigtimedwait\(\)](#)

63191 XBD [<signal.h>](#), [<time.h>](#)

63192 CHANGE HISTORY

63193 First released in Issue 5. Included for alignment with the POSIX Realtime Extension and the
63194 POSIX Threads Extension.

63195 Issue 6

63196 The **restrict** keyword is added to the *sigwait()* prototype for alignment with the
63197 ISO/IEC 9899:1999 standard.

63198 IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/131 is applied, updating the
63199 DESCRIPTION to state that if more than a single thread is blocked in *sigwait()*, it is unspecified
63200 which of the waiting threads returns, and that if a signal is generated for a specific thread only
63201 that thread shall return.

63202 Issue 7

63203 Functionality relating to the Realtime Signals Extension option is moved to the Base.

63204 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0584 [76] is applied.

+

63205 **NAME**

63206 sigwaitinfo — wait for queued signals

63207 **SYNOPSIS**

```
63208 #include <signal.h>  
63209 int sigwaitinfo(const sigset_t *restrict set, siginfo_t *restrict info);
```

63210 **DESCRIPTION**

63211 Refer to *sigtimedwait()*.

63212 **NAME**

63213 sin, sinf, sinl — sine function

63214 **SYNOPSIS**

```
63215 #include <math.h>
63216 double sin(double x);
63217 float sinf(float x);
63218 long double sinl(long double x);
```

63219 **DESCRIPTION**

63220 CX The functionality described on this reference page is aligned with the ISO C standard. Any
 63221 conflict between the requirements described here and the ISO C standard is unintentional. This
 63222 volume of POSIX.1-2008 defers to the ISO C standard.

63223 These functions shall compute the sine of their argument x , measured in radians.

63224 An application wishing to check for error situations should set *errno* to zero and call
 63225 *feclearexcept*(FE_ALL_EXCEPT) before calling these functions. On return, if *errno* is non-zero or
 63226 *fetestexcept*(FE_INVALID | FE_DIVBYZERO | FE_OVERFLOW | FE_UNDERFLOW) is non-
 63227 zero, an error has occurred.

63228 **RETURN VALUE**

63229 Upon successful completion, these functions shall return the sine of x .

63230 MX If x is NaN, a NaN shall be returned.

63231 If x is ± 0 , x shall be returned.

63232 If x is subnormal, a range error may occur

63233 MXX and x should be returned.

63234 MX If x is not returned, *sin()*, *sinf()*, and *sinl()* shall return an implementation-defined value no
 63235 greater in magnitude than DBL_MIN, FLT_MIN, and LDBL_MIN, respectively.

63236 If x is $\pm\text{Inf}$, a domain error shall occur, and a NaN shall be returned.

63237 **ERRORS**

63238 These functions shall fail if:

63239 MX **Domain Error** The x argument is $\pm\text{Inf}$.

63240 If the integer expression (*math_errhandling* & MATH_ERRNO) is non-zero,
 63241 then *errno* shall be set to [EDOM]. If the integer expression (*math_errhandling*
 63242 & MATH_ERREXCEPT) is non-zero, then the invalid floating-point exception
 63243 shall be raised.

63244 These functions may fail if:

63245 MX **Range Error** The value of x is subnormal

63246 If the integer expression (*math_errhandling* & MATH_ERRNO) is non-zero,
 63247 then *errno* shall be set to [ERANGE]. If the integer expression
 63248 (*math_errhandling* & MATH_ERREXCEPT) is non-zero, then the underflow
 63249 floating-point exception shall be raised.

63250 **EXAMPLES**63251 **Taking the Sine of a 45-Degree Angle**

```

63252 #include <math.h>
63253 ...
63254 double radians = 45.0 * M_PI / 180;
63255 double result;
63256 ...
63257 result = sin(radians);

```

63258 **APPLICATION USAGE**

63259 These functions may lose accuracy when their argument is near a multiple of π or is far from 0.0.

63260 On error, the expressions (*math_errhandling* & MATH_ERRNO) and (*math_errhandling* &
 63261 MATH_ERREXCEPT) are independent of each other, but at least one of them must be non-zero.

63262 **RATIONALE**

63263 None.

63264 **FUTURE DIRECTIONS**

63265 None.

63266 **SEE ALSO**

63267 *asin()*, *feclearexcept()*, *fetestexcept()*, *isnan()*

63268 XBD Section 4.19 (on page 116), **<math.h>**

63269 **CHANGE HISTORY**

63270 First released in Issue 1. Derived from Issue 1 of the SVID.

63271 **Issue 5**

63272 The last two paragraphs of the DESCRIPTION were included as APPLICATION USAGE notes
 63273 in previous issues.

63274 **Issue 6**

63275 The *sinf()* and *sinl()* functions are added for alignment with the ISO/IEC 9899:1999 standard.

63276 The DESCRIPTION, RETURN VALUE, ERRORS, and APPLICATION USAGE sections are
 63277 revised to align with the ISO/IEC 9899:1999 standard.

63278 IEC 60559:1989 standard floating-point extensions over the ISO/IEC 9899:1999 standard are
 63279 marked.

63280 **Issue 7**

63281 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0585 [68] and XSH/TC1-2008/0586 +
 63282 [320] are applied.

63283 **NAME**

63284 sinh, sinhf, sinh1 — hyperbolic sine functions

63285 **SYNOPSIS**

```
63286       #include <math.h>
63287       double sinh(double x);
63288       float  sinhf(float x);
63289       long double sinh1(long double x);
```

63290 **DESCRIPTION**

63291 CX The functionality described on this reference page is aligned with the ISO C standard. Any
 63292 conflict between the requirements described here and the ISO C standard is unintentional. This
 63293 volume of POSIX.1-2008 defers to the ISO C standard.

63294 These functions shall compute the hyperbolic sine of their argument x .

63295 An application wishing to check for error situations should set *errno* to zero and call
 63296 *feclearexcept*(FE_ALL_EXCEPT) before calling these functions. On return, if *errno* is non-zero or
 63297 *fetestexcept*(FE_INVALID | FE_DIVBYZERO | FE_OVERFLOW | FE_UNDERFLOW) is non-
 63298 zero, an error has occurred.

63299 **RETURN VALUE**

63300 Upon successful completion, these functions shall return the hyperbolic sine of x .

63301 If the result would cause an overflow, a range error shall occur and $\pm\text{HUGE\_VAL}$,
 63302 $\pm\text{HUGE\_VALF}$, and $\pm\text{HUGE\_VALL}$ (with the same sign as x) shall be returned as appropriate for
 63303 the type of the function.

63304 MX If x is NaN, a NaN shall be returned.

63305 If x is ± 0 or $\pm\text{Inf}$, x shall be returned.

63306 If x is subnormal, a range error may occur

63307 MXX and x should be returned.

63308 MX If x is not returned, *sinh*(), *sinhf*(), and *sinh1*() shall return an implementation-defined value no
 63309 greater in magnitude than DBL_MIN, FLT_MIN, and LDBL_MIN, respectively.

63310 **ERRORS**

63311 These functions shall fail if:

63312 Range Error The result would cause an overflow.

63313 If the integer expression (*math_errhandling* & MATH_ERRNO) is non-zero,
 63314 then *errno* shall be set to [ERANGE]. If the integer expression
 63315 (*math_errhandling* & MATH_ERREXCEPT) is non-zero, then the overflow
 63316 floating-point exception shall be raised.

63317 These functions may fail if:

63318 MX Range Error The value x is subnormal.

63319 If the integer expression (*math_errhandling* & MATH_ERRNO) is non-zero,
 63320 then *errno* shall be set to [ERANGE]. If the integer expression
 63321 (*math_errhandling* & MATH_ERREXCEPT) is non-zero, then the underflow
 63322 floating-point exception shall be raised.

63323 **EXAMPLES**

63324 None.

63325 **APPLICATION USAGE**

63326 On error, the expressions (*math_errhandling* & MATH_ERRNO) and (*math_errhandling* &
 63327 MATH_ERREXCEPT) are independent of each other, but at least one of them must be non-zero.

63328 **RATIONALE**

63329 None.

63330 **FUTURE DIRECTIONS**

63331 None.

63332 **SEE ALSO**63333 *asinh()*, *cosh()*, *feclearexcept()*, *fetestexcept()*, *isnan()*, *tanh()*

63334 XBD Section 4.19 (on page 116), <math.h>

63335 **CHANGE HISTORY**

63336 First released in Issue 1. Derived from Issue 1 of the SVID.

63337 **Issue 5**

63338 The DESCRIPTION is updated to indicate how an application should check for an error. This
 63339 text was previously published in the APPLICATION USAGE section.

63340 **Issue 6**63341 The *sinhf()* and *sinhl()* functions are added for alignment with the ISO/IEC 9899:1999 standard.

63342 The DESCRIPTION, RETURN VALUE, ERRORS, and APPLICATION USAGE sections are
 63343 revised to align with the ISO/IEC 9899:1999 standard.

63344 IEC 60559:1989 standard floating-point extensions over the ISO/IEC 9899:1999 standard are
 63345 marked.

63346 **Issue 7**

63347 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0587 [68] is applied.

+

63348 **NAME**

63349 sinl — sine function

63350 **SYNOPSIS**

63351 #include <math.h>

63352 long double sinl(long double *x*);63353 **DESCRIPTION**63354 Refer to *sin()*.

NAME

sleep — suspend execution for an interval of time

SYNOPSIS

```
#include <unistd.h>
```

```
unsigned sleep(unsigned seconds);
```

DESCRIPTION

The *sleep()* function shall cause the calling thread to be suspended from execution until either the number of realtime seconds specified by the argument *seconds* has elapsed or a signal is delivered to the calling thread and its action is to invoke a signal-catching function or to terminate the process. The suspension time may be longer than requested due to the scheduling of other activity by the system.

If a SIGALRM signal is generated for the calling process during execution of *sleep()* and if the SIGALRM signal is being ignored or blocked from delivery, it is unspecified whether *sleep()* returns when the SIGALRM signal is scheduled. If the signal is being blocked, it is also unspecified whether it remains pending after *sleep()* returns or it is discarded.

If a SIGALRM signal is generated for the calling process during execution of *sleep()*, except as a result of a prior call to *alarm()*, and if the SIGALRM signal is not being ignored or blocked from delivery, it is unspecified whether that signal has any effect other than causing *sleep()* to return.

If a signal-catching function interrupts *sleep()* and examines or changes either the time a SIGALRM is scheduled to be generated, the action associated with the SIGALRM signal, or whether the SIGALRM signal is blocked from delivery, the results are unspecified.

If a signal-catching function interrupts *sleep()* and calls *siglongjmp()* or *longjmp()* to restore an environment saved prior to the *sleep()* call, the action associated with the SIGALRM signal and the time at which a SIGALRM signal is scheduled to be generated are unspecified. It is also unspecified whether the SIGALRM signal is blocked, unless the signal mask of the process is restored as part of the environment.

XSI Interactions between *sleep()* and *setitimer()* are unspecified.

RETURN VALUE

If *sleep()* returns because the requested time has elapsed, the value returned shall be 0. If *sleep()* returns due to delivery of a signal, the return value shall be the “unslept” amount (the requested time minus the time actually slept) in seconds.

ERRORS

No errors are defined.

EXAMPLES

None.

APPLICATION USAGE

None.

RATIONALE

There are two general approaches to the implementation of the *sleep()* function. One is to use the *alarm()* function to schedule a SIGALRM signal and then suspend the calling thread waiting for that signal. The other is to implement an independent facility. This volume of POSIX.1-2008 permits either approach.

In order to comply with the requirement that no primitive shall change a process attribute unless explicitly described by this volume of POSIX.1-2008, an implementation using SIGALRM must carefully take into account any SIGALRM signal scheduled by previous *alarm()* calls, the action

previously established for SIGALRM, and whether SIGALRM was blocked. If a SIGALRM has been scheduled before the *sleep()* would ordinarily complete, the *sleep()* must be shortened to that time and a SIGALRM generated (possibly simulated by direct invocation of the signal-catching function) before *sleep()* returns. If a SIGALRM has been scheduled after the *sleep()* would ordinarily complete, it must be rescheduled for the same time before *sleep()* returns. The action and blocking for SIGALRM must be saved and restored.

Historical implementations often implement the SIGALRM-based version using *alarm()* and *pause()*. One such implementation is prone to infinite hangups, as described in *pause()*. Another such implementation uses the C-language *setjmp()* and *longjmp()* functions to avoid that window. That implementation introduces a different problem: when the SIGALRM signal interrupts a signal-catching function installed by the user to catch a different signal, the *longjmp()* aborts that signal-catching function. An implementation based on *sigprocmask()*, *alarm()*, and *sigsuspend()* can avoid these problems.

Despite all reasonable care, there are several very subtle, but detectable and unavoidable, differences between the two types of implementations. These are the cases mentioned in this volume of POSIX.1-2008 where some other activity relating to SIGALRM takes place, and the results are stated to be unspecified. All of these cases are sufficiently unusual as not to be of concern to most applications.

See also the discussion of the term *realtime* in *alarm()*.

Since *sleep()* can be implemented using *alarm()*, the discussion about alarms occurring early under *alarm()* applies to *sleep()* as well.

Application developers should note that the type of the argument *seconds* and the return value of *sleep()* is **unsigned**. That means that a Strictly Conforming POSIX System Interfaces Application cannot pass a value greater than the minimum guaranteed value for {UINT_MAX}, which the ISO C standard sets as 65535, and any application passing a larger value is restricting its portability. A different type was considered, but historical implementations, including those with a 16-bit **int** type, consistently use either **unsigned** or **int**.

Scheduling delays may cause the process to return from the *sleep()* function significantly after the requested time. In such cases, the return value should be set to zero, since the formula (requested time minus the time actually spent) yields a negative number and *sleep()* returns an **unsigned**.

FUTURE DIRECTIONS

None.

SEE ALSO

alarm(), *getitimer()*, *nanosleep()*, *pause()*, *sigaction()*, *sigsetjmp()*

XBD <**unistd.h**>

CHANGE HISTORY

First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 5

The DESCRIPTION is updated for alignment with the POSIX Threads Extension.

Issue 6

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/132 is applied, making a correction in the RATIONALE section.

63443 **NAME**

63444 snprintf — print formatted output

63445 **SYNOPSIS**

63446 #include <stdio.h>

63447 int snprintf(char *restrict *s*, size_t *n*,
63448 const char *restrict *format*, ...);

63449 **DESCRIPTION**

63450 Refer to *fprintf()*.

NAME

socketmark — determine whether a socket is at the out-of-band mark

SYNOPSIS

```
#include <sys/socket.h>
```

```
int socketmark(int s);
```

DESCRIPTION

The *socketmark()* function shall determine whether the socket specified by the descriptor *s* is at the out-of-band data mark (see [Section 2.10.12](#), on page 525). If the protocol for the socket supports out-of-band data by marking the stream with an out-of-band data mark, the *socketmark()* function shall return 1 when all data preceding the mark has been read and the out-of-band data mark is the first element in the receive queue. The *socketmark()* function shall not remove the mark from the stream.

RETURN VALUE

Upon successful completion, the *socketmark()* function shall return a value indicating whether the socket is at an out-of-band data mark. If the protocol has marked the data stream and all data preceding the mark has been read, the return value shall be 1; if there is no mark, or if data precedes the mark in the receive queue, the *socketmark()* function shall return 0. Otherwise, it shall return a value of -1 and set *errno* to indicate the error.

ERRORS

The *socketmark()* function shall fail if:

- [EBADF] The *s* argument is not a valid file descriptor.
- [ENOTTY] The file associated with the *s* argument is not a socket.

EXAMPLES

None.

APPLICATION USAGE

The use of this function between receive operations allows an application to determine which received data precedes the out-of-band data and which follows the out-of-band data.

There is an inherent race condition in the use of this function. On an empty receive queue, the current read of the location might well be at the “mark”, but the system has no way of knowing that the next data segment that will arrive from the network will carry the mark, and *socketmark()* will return false, and the next read operation will silently consume the mark.

Hence, this function can only be used reliably when the application already knows that the out-of-band data has been seen by the system or that it is known that there is data waiting to be read at the socket (via SIGURG or *select()*). See [Section 2.10.11](#) (on page 525), [Section 2.10.12](#) (on page 525), [Section 2.10.14](#) (on page 526), and *pselect()* for details.

RATIONALE

The *socketmark()* function replaces the historical SIOCATMARK command to *ioctl()* which implemented the same functionality on many implementations. Using a wrapper function follows the adopted conventions to avoid specifying commands to the *ioctl()* function, other than those now included to support XSI STREAMS. The *socketmark()* function could be implemented as follows:

```
#include <sys/ioctl.h>

int socketmark(int s)
{
    int val;
```

```

63496         if (ioctl(s, SIOCATMARK, &val) == -1)
63497             return (-1);
63498         return (val);
63499     }

```

63500 The use of [ENOTTY] to indicate an incorrect descriptor type matches the historical behavior of
63501 SIOCATMARK.

63502 FUTURE DIRECTIONS

63503 None.

63504 SEE ALSO

63505 [Section 2.10.12](#) (on page 525), [pselect\(\)](#), [recv\(\)](#), [recvmsg\(\)](#)

63506 XBD [<sys/socket.h>](#)

63507 CHANGE HISTORY

63508 First released in Issue 6. Derived from IEEE Std 1003.1g-2000.

63509 Issue 7

63510 SD5-XSH-ERN-100 is applied, correcting the definition of the [ENOTTY] error condition.

NAME

socket — create an endpoint for communication

SYNOPSIS

```
#include <sys/socket.h>

int socket(int domain, int type, int protocol);
```

DESCRIPTION

The *socket()* function shall create an unbound socket in a communications domain, and return a file descriptor that can be used in later function calls that operate on sockets.

The *socket()* function takes the following arguments:

<i>domain</i>	Specifies the communications domain in which a socket is to be created.
<i>type</i>	Specifies the type of socket to be created.
<i>protocol</i>	Specifies a particular protocol to be used with the socket. Specifying a <i>protocol</i> of 0 causes <i>socket()</i> to use an unspecified default protocol appropriate for the requested socket type.

The *domain* argument specifies the address family used in the communications domain. The address families supported by the system are implementation-defined.

Symbolic constants that can be used for the domain argument are defined in the **<sys/socket.h>** header.

The *type* argument specifies the socket type, which determines the semantics of communication over the socket. The following socket types are defined; implementations may specify additional socket types:

SOCK_STREAM	Provides sequenced, reliable, bidirectional, connection-mode byte streams, and may provide a transmission mechanism for out-of-band data.
SOCK_DGRAM	Provides datagrams, which are connectionless-mode, unreliable messages of fixed maximum length.
SOCK_SEQPACKET	Provides sequenced, reliable, bidirectional, connection-mode transmission paths for records. A record can be sent using one or more output operations and received using one or more input operations, but a single operation never transfers part of more than one record. Record boundaries are visible to the receiver via the MSG_EOR flag.

If the *protocol* argument is non-zero, it shall specify a protocol that is supported by the address family. If the *protocol* argument is zero, the default protocol for this address family and type shall be used. The protocols supported by the system are implementation-defined.

The process may need to have appropriate privileges to use the *socket()* function or to create some sockets.

RETURN VALUE

Upon successful completion, *socket()* shall return a non-negative integer, the socket file descriptor. Otherwise, a value of -1 shall be returned and *errno* set to indicate the error.

ERRORS

The *socket()* function shall fail if:

[EAFNOSUPPORT]

The implementation does not support the specified address family.

[EMFILE]

All file descriptors available to the process are currently open.

[ENFILE]

No more file descriptors are available for the system.

[EPROTONOSUPPORT]

The protocol is not supported by the address family, or the protocol is not supported by the implementation.

[EPROTOTYPE] The socket type is not supported by the protocol.

The *socket()* function may fail if:

[EACCES]

The process does not have appropriate privileges.

[ENOBUFS]

Insufficient resources were available in the system to perform the operation.

[ENOMEM]

Insufficient memory was available to fulfill the request.

EXAMPLES

None.

APPLICATION USAGE

The documentation for specific address families specifies which protocols each address family supports. The documentation for specific protocols specifies which socket types each protocol supports.

The application can determine whether an address family is supported by trying to create a socket with *domain* set to the protocol in question.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

[*accept\(\)*](#), [*bind\(\)*](#), [*connect\(\)*](#), [*getsockname\(\)*](#), [*getsockopt\(\)*](#), [*listen\(\)*](#), [*recv\(\)*](#), [*recvfrom\(\)*](#), [*recvmsg\(\)*](#), [*send\(\)*](#), [*sendmsg\(\)*](#), [*setsockopt\(\)*](#), [*shutdown\(\)*](#), [*socketpair\(\)*](#)

XBD [*<netinet/in.h>*](#), [*<sys/socket.h>*](#)

CHANGE HISTORY

First released in Issue 6. Derived from the XNS, Issue 5.2 specification.

NAME

socketpair — create a pair of connected sockets

SYNOPSIS

```
#include <sys/socket.h>

int socketpair(int domain, int type, int protocol,
               int socket_vector[2]);
```

DESCRIPTION

The *socketpair()* function shall create an unbound pair of connected sockets in a specified *domain*, of a specified *type*, under the protocol optionally specified by the *protocol* argument. The two sockets shall be identical. The file descriptors used in referencing the created sockets shall be returned in *socket_vector*[0] and *socket_vector*[1].

The *socketpair()* function takes the following arguments:

<i>domain</i>	Specifies the communications domain in which the sockets are to be created.
<i>type</i>	Specifies the type of sockets to be created.
<i>protocol</i>	Specifies a particular protocol to be used with the sockets. Specifying a <i>protocol</i> of 0 causes <i>socketpair()</i> to use an unspecified default protocol appropriate for the requested socket type.
<i>socket_vector</i>	Specifies a 2-integer array to hold the file descriptors of the created socket pair.

The *type* argument specifies the socket type, which determines the semantics of communications over the socket. The following socket types are defined; implementations may specify additional socket types:

SOCK_STREAM	Provides sequenced, reliable, bidirectional, connection-mode byte streams, and may provide a transmission mechanism for out-of-band data.
SOCK_DGRAM	Provides datagrams, which are connectionless-mode, unreliable messages of fixed maximum length.
SOCK_SEQPACKET	Provides sequenced, reliable, bidirectional, connection-mode transmission paths for records. A record can be sent using one or more output operations and received using one or more input operations, but a single operation never transfers part of more than one record. Record boundaries are visible to the receiver via the MSG_EOR flag.

If the *protocol* argument is non-zero, it shall specify a protocol that is supported by the address family. If the *protocol* argument is zero, the default protocol for this address family and type shall be used. The protocols supported by the system are implementation-defined.

The process may need to have appropriate privileges to use the *socketpair()* function or to create some sockets.

RETURN VALUE

Upon successful completion, this function shall return 0; otherwise, *-1* shall be returned and *errno* set to indicate the error.

ERRORS

The *socketpair()* function shall fail if:

[EAFNOSUPPORT]

The implementation does not support the specified address family.

63625 [EMFILE] All, or all but one, of the file descriptors available to the process are currently
 63626 open.

63627 [ENFILE] No more file descriptors are available for the system.

63628 [EOPNOTSUPP] The specified protocol does not permit creation of socket pairs.

63629 [EPROTONOSUPPORT]
 63630 The protocol is not supported by the address family, or the protocol is not
 63631 supported by the implementation.

63632 [EPROTOYPE] The socket type is not supported by the protocol.

63633 The *socketpair()* function may fail if:

63634 [EACCES] The process does not have appropriate privileges.

63635 [ENOBUFS] Insufficient resources were available in the system to perform the operation.

63636 [ENOMEM] Insufficient memory was available to fulfill the request.

63637 EXAMPLES

63638 None.

63639 APPLICATION USAGE

63640 The documentation for specific address families specifies which protocols each address family
 63641 supports. The documentation for specific protocols specifies which socket types each protocol
 63642 supports.

63643 The *socketpair()* function is used primarily with UNIX domain sockets and need not be
 63644 supported for other domains.

63645 RATIONALE

63646 None.

63647 FUTURE DIRECTIONS

63648 None.

63649 SEE ALSO

63650 *socket()*

63651 XBD <sys/socket.h>

63652 CHANGE HISTORY

63653 First released in Issue 6. Derived from the XNS, Issue 5.2 specification.

63654 Issue 7

63655 The description of the [EMFILE] error condition is aligned with the *pipe()* function.

63656 **NAME**

63657 sprintf — print formatted output

63658 **SYNOPSIS**

63659 #include <stdio.h>

63660 int sprintf(char *restrict *s*, const char *restrict *format*, ...);63661 **DESCRIPTION**63662 Refer to *fprintf()*.

63663 **NAME**

63664 sqrt, sqrtf, sqrtl — square root function

63665 **SYNOPSIS**

```
63666       #include <math.h>
63667       double sqrt(double x);
63668       float sqrtf(float x);
63669       long double sqrtl(long double x);
```

63670 **DESCRIPTION**

63671 CX The functionality described on this reference page is aligned with the ISO C standard. Any
 63672 conflict between the requirements described here and the ISO C standard is unintentional. This
 63673 volume of POSIX.1-2008 defers to the ISO C standard.

63674 These functions shall compute the square root of their argument x , $\sqrt{x}$.

63675 An application wishing to check for error situations should set *errno* to zero and call
 63676 *feclearexcept*(FE_ALL_EXCEPT) before calling these functions. On return, if *errno* is non-zero or
 63677 *fetestexcept*(FE_INVALID | FE_DIVBYZERO | FE_OVERFLOW | FE_UNDERFLOW) is non-
 63678 zero, an error has occurred.

63679 **RETURN VALUE**

63680 Upon successful completion, these functions shall return the square root of x .

63681 MX For finite values of $x < -0$, a domain error shall occur, and either a NaN (if supported), or an
 63682 implementation-defined value shall be returned.

63683 MX If x is NaN, a NaN shall be returned.

63684 If x is ± 0 or $+\text{Inf}$, x shall be returned.

63685 If x is $-\text{Inf}$, a domain error shall occur, and a NaN shall be returned.

63686 **ERRORS**

63687 These functions shall fail if:

63688 MX Domain Error The finite value of x is < -0 , or x is $-\text{Inf}$.

63689 If the integer expression *(math_errhandling & MATH_ERRNO)* is non-zero,
 63690 then *errno* shall be set to [EDOM]. If the integer expression *(math_errhandling*
 63691 *& MATH_ERREXCEPT)* is non-zero, then the invalid floating-point exception
 63692 shall be raised.

63693 **EXAMPLES**63694 **Taking the Square Root of 9.0**

```
63695       #include <math.h>
63696       ...
63697       double x = 9.0;
63698       double result;
63699       ...
63700       result = sqrt(x);
```

63701 **APPLICATION USAGE**

63702 On error, the expressions *(math_errhandling & MATH_ERRNO)* and *(math_errhandling &*
 63703 *MATH_ERREXCEPT)* are independent of each other, but at least one of them must be non-zero.

63704 **RATIONALE**

63705 None.

63706 **FUTURE DIRECTIONS**

63707 None.

63708 **SEE ALSO**63709 *feclearexcept()*, *fetestexcept()*, *isnan()*63710 XBD Section 4.19 (on page 116), *<math.h>*, *<stdio.h>*63711 **CHANGE HISTORY**

63712 First released in Issue 1. Derived from Issue 1 of the SVID.

63713 **Issue 5**63714 The DESCRIPTION is updated to indicate how an application should check for an error. This
63715 text was previously published in the APPLICATION USAGE section.63716 **Issue 6**63717 The *sqrtrf()* and *sqrtrl()* functions are added for alignment with the ISO/IEC 9899:1999 standard.63718 The DESCRIPTION, RETURN VALUE, ERRORS, and APPLICATION USAGE sections are
63719 revised to align with the ISO/IEC 9899:1999 standard.63720 IEC 60559:1989 standard floating-point extensions over the ISO/IEC 9899:1999 standard are
63721 marked.63722 **Issue 7**

63723 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0588 [320] is applied.

+

63724 **NAME**

63725 srand — pseudo-random number generator

63726 **SYNOPSIS**

63727 #include <stdlib.h>

63728 void srand(unsigned seed);

63729 **DESCRIPTION**

63730 Refer to *rand()*.

63731 **NAME**

63732 srand48 — seed the uniformly distributed double-precision pseudo-random number generator

63733 **SYNOPSIS**

63734 XSI #include <stdlib.h>

63735 void srand48(long seedval);

63736 **DESCRIPTION**63737 Refer to *drand48()*.

63738 **NAME**

63739 srandom — seed pseudo-random number generator

63740 **SYNOPSIS**

63741 XSI #include <stdlib.h>

63742 void srandom(unsigned seed);

63743 **DESCRIPTION**

63744 Refer to *initstate()*.

63745 **NAME**63746 `sscanf` — convert formatted input63747 **SYNOPSIS**63748 `#include <stdio.h>`63749 `int sscanf(const char *restrict s, const char *restrict format, ...);`63750 **DESCRIPTION**63751 Refer to [*fscanf\(\)*](#).

63752 **NAME**

63753 stat — get file status

63754 **SYNOPSIS**

63755 #include <sys/stat.h>

63756 int stat(const char *restrict *path*, struct stat *restrict *buf*);63757 **DESCRIPTION**63758 Refer to *fstatat()*.

63759 **NAME**

63760 statvfs — get file system information

63761 **SYNOPSIS**

63762 #include <sys/statvfs.h>

63763 int statvfs(const char *restrict *path*, struct statvfs *restrict *buf*);63764 **DESCRIPTION**63765 Refer to *fstatvfs()*.

NAME

stderr, stdin, stdout — standard I/O streams

SYNOPSIS

```
#include <stdio.h>
```

```
extern FILE *stderr, *stdin, *stdout;
```

DESCRIPTION

The functionality described on this reference page is aligned with the ISO C standard. Any conflict between the requirements described here and the ISO C standard is unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

A file with associated buffering is called a *stream* and is declared to be a pointer to a defined type **FILE**. The *fopen()* function shall create certain descriptive data for a stream and return a pointer to designate the stream in all further transactions. Normally, there are three open streams with constant pointers declared in the **<stdio.h>** header and associated with the standard open files.

At program start-up, three streams shall be predefined and need not be opened explicitly: *standard input* (for reading conventional input), *standard output* (for writing conventional output), and *standard error* (for writing diagnostic output). When opened, the standard error stream is not fully buffered; the standard input and standard output streams are fully buffered if and only if the stream can be determined not to refer to an interactive device.

The following symbolic values in **<unistd.h>** define the file descriptors that shall be associated with the C-language *stdin*, *stdout*, and *stderr* when the application is started:

STDIN_FILENO Standard input value, *stdin*. Its value is 0.

STDOUT_FILENO Standard output value, *stdout*. Its value is 1.

STDERR_FILENO Standard error value, *stderr*. Its value is 2.

The *stderr* stream is expected to be open for reading and writing.

RETURN VALUE

None.

ERRORS

No errors are defined.

EXAMPLES

None.

APPLICATION USAGE

None.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

fclose(), *feof()*, *ferror()*, *fileno()*, *fopen()*, *fprintf()*, *fread()*, *fscanf()*, *fseek()*, *getc()*, *gets()*, *popen()*, *putc()*, *puts()*, *read()*, *setbuf()*, *setvbuf()*, *tmpfile()*, *ungetc()*, *vfprintf()*

XBD **<stdio.h>**, **<unistd.h>**

CHANGE HISTORY

63806
63807 First released in Issue 1.

Issue 6

63808
63809 Extensions beyond the ISO C standard are marked.

63810 A note that *stderr* is expected to be open for reading and writing is added to the DESCRIPTION.

63811 **NAME**

63812 stpcpy — copy a string and return a pointer to the end of the result

63813 **SYNOPSIS**

63814 CX `#include <string.h>`

63815 `char *stpcpy(char *restrict s1, const char *restrict s2);`

63816 **DESCRIPTION**

63817 Refer to *strcpy()*.

63818 **NAME**

63819 stpncpy — copy fixed length string, returning a pointer to the array end

63820 **SYNOPSIS**63821 CX `#include <string.h>`63822 `char *stpncpy(char *restrict s1, const char *restrict s2, size_t size);`63823 **DESCRIPTION**63824 Refer to *strncpy()*.

NAME

strcasecmp, strcasecmp_l, strncasecmp, strncasecmp_l — case-insensitive string comparisons

SYNOPSIS

```
#include <strings.h>

int strcasecmp(const char *s1, const char *s2);
int strcasecmp_l(const char *s1, const char *s2,
    locale_t locale);
int strncasecmp(const char *s1, const char *s2, size_t n);
int strncasecmp_l(const char *s1, const char *s2,
    size_t n, locale_t locale);
```

DESCRIPTION

The *strcasecmp()* and *strcasecmp_l()* functions shall compare, while ignoring differences in case, the string pointed to by *s1* to the string pointed to by *s2*. The *strncasecmp()* and *strncasecmp_l()* functions shall compare, while ignoring differences in case, not more than *n* bytes from the string pointed to by *s1* to the string pointed to by *s2*.

The *strcasecmp()* and *strncasecmp()* functions use the current locale to determine the case of the characters.

The *strcasecmp_l()* and *strncasecmp_l()* functions use the locale represented by *locale* to determine the case of the characters.

When the *LC_CTYPE* category of the locale being used is from the POSIX locale, these functions shall behave as if the strings had been converted to lowercase and then a byte comparison performed. Otherwise, the results are unspecified.

The behavior is undefined if the *locale* argument to *strcasecmp_l()* or *strncasecmp_l()* is the special locale object *LC_GLOBAL_LOCALE* or is not a valid locale object handle.

RETURN VALUE

Upon completion, *strcasecmp()* and *strcasecmp_l()* shall return an integer greater than, equal to, or less than 0, if the string pointed to by *s1* is, ignoring case, greater than, equal to, or less than the string pointed to by *s2*, respectively.

Upon successful completion, *strncasecmp()* and *strncasecmp_l()* shall return an integer greater than, equal to, or less than 0, if the possibly null-terminated array pointed to by *s1* is, ignoring case, greater than, equal to, or less than the possibly null-terminated array pointed to by *s2*, respectively.

ERRORS

No errors are defined.

EXAMPLES

None.

APPLICATION USAGE

None.

RATIONALE

None.

FUTURE DIRECTIONS

None.

63867 **SEE ALSO**63868 [wcscasecmp\(\)](#)63869 XBD [<strings.h>](#)63870 **CHANGE HISTORY**

63871 First released in Issue 4, Version 2.

63872 **Issue 5**

63873 Moved from X/OPEN UNIX extension to BASE.

63874 **Issue 7**63875 The *strcasecmp()* and *strncasecmp()* functions are moved from the XSI option to the Base.63876 The *strcasecmp_l()* and *strncasecmp_l()* functions are added from The Open Group Technical
63877 Standard, 2006, Extended API Set Part 4.63878 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0589 [302], XSH/TC1-2008/0590 [294], +
63879 XSH/TC1-2008/0591 [283], and XSH/TC1-2008/0592 [283] are applied.

63880 **NAME**

63881 strcat — concatenate two strings

63882 **SYNOPSIS**

63883 #include <string.h>

63884 char *strcat(char *restrict *s1*, const char *restrict *s2*);63885 **DESCRIPTION**

63886 CX The functionality described on this reference page is aligned with the ISO C standard. Any
63887 conflict between the requirements described here and the ISO C standard is unintentional. This
63888 volume of POSIX.1-2008 defers to the ISO C standard.

63889 The *strcat()* function shall append a copy of the string pointed to by *s2* (including the
63890 terminating NUL character) to the end of the string pointed to by *s1*. The initial byte of *s2*
63891 overwrites the NUL character at the end of *s1*. If copying takes place between objects that
63892 overlap, the behavior is undefined.

63893 **RETURN VALUE**63894 The *strcat()* function shall return *s1*; no return value is reserved to indicate an error.63895 **ERRORS**

63896 No errors are defined.

63897 **EXAMPLES**

63898 None.

63899 **APPLICATION USAGE**

63900 This version is aligned with the ISO C standard; this does not affect compatibility with XPG3
63901 applications. Reliable error detection by this function was never guaranteed.

63902 **RATIONALE**

63903 None.

63904 **FUTURE DIRECTIONS**

63905 None.

63906 **SEE ALSO**63907 [*strncat\(\)*](#)63908 XBD [**<string.h>**](#)63909 **CHANGE HISTORY**

63910 First released in Issue 1. Derived from Issue 1 of the SVID.

63911 **Issue 6**63912 The *strcat()* prototype is updated for alignment with the ISO/IEC 9899:1999 standard.

63913 **NAME**

63914 strchr — string scanning operation

63915 **SYNOPSIS**

63916 #include <string.h>

63917 char *strchr(const char *s, int c);

63918 **DESCRIPTION**

63919 CX The functionality described on this reference page is aligned with the ISO C standard. Any
63920 conflict between the requirements described here and the ISO C standard is unintentional. This
63921 volume of POSIX.1-2008 defers to the ISO C standard.

63922 The *strchr()* function shall locate the first occurrence of *c* (converted to a **char**) in the string
63923 pointed to by *s*. The terminating NUL character is considered to be part of the string.

63924 **RETURN VALUE**

63925 Upon completion, *strchr()* shall return a pointer to the byte, or a null pointer if the byte was not
63926 found.

63927 **ERRORS**

63928 No errors are defined.

63929 **EXAMPLES**

63930 None.

63931 **APPLICATION USAGE**

63932 None.

63933 **RATIONALE**

63934 None.

63935 **FUTURE DIRECTIONS**

63936 None.

63937 **SEE ALSO**63938 [*strrchr\(\)*](#)63939 XBD [**<string.h>**](#)63940 **CHANGE HISTORY**

63941 First released in Issue 1. Derived from Issue 1 of the SVID.

63942 **Issue 6**

63943 Extensions beyond the ISO C standard are marked.

NAME

strcmp — compare two strings

SYNOPSIS

```
#include <string.h>
```

```
int strcmp(const char *s1, const char *s2);
```

DESCRIPTION

CX The functionality described on this reference page is aligned with the ISO C standard. Any conflict between the requirements described here and the ISO C standard is unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

The *strcmp()* function shall compare the string pointed to by *s1* to the string pointed to by *s2*.

The sign of a non-zero return value shall be determined by the sign of the difference between the values of the first pair of bytes (both interpreted as type **unsigned char**) that differ in the strings being compared.

RETURN VALUE

Upon completion, *strcmp()* shall return an integer greater than, equal to, or less than 0, if the string pointed to by *s1* is greater than, equal to, or less than the string pointed to by *s2*, respectively.

ERRORS

No errors are defined.

EXAMPLES**Checking a Password Entry**

The following example compares the information read from standard input to the value of the name of the user entry. If the *strcmp()* function returns 0 (indicating a match), a further check will be made to see if the user entered the proper old password. The *crypt()* function shall encrypt the old password entered by the user, using the value of the encrypted password in the **passwd** structure as the salt. If this value matches the value of the encrypted **passwd** in the structure, the entered password *oldpasswd* is the correct user's password. Finally, the program encrypts the new password so that it can store the information in the **passwd** structure.

```
#include <string.h>
#include <unistd.h>
#include <stdio.h>
...
int valid_change;
struct passwd *p;
char user[100];
char oldpasswd[100];
char newpasswd[100];
char savepasswd[100];
...
if (strcmp(p->pw_name, user) == 0) {
    if (strcmp(p->pw_passwd, crypt(oldpasswd, p->pw_passwd)) == 0) {
        strcpy(savepasswd, crypt(newpasswd, user));
        p->pw_passwd = savepasswd;
        valid_change = 1;
    }
    else {
```

```
63990         fprintf(stderr, "Old password is not valid\n");
63991     }
63992 }
63993 ...
```

63994 APPLICATION USAGE

63995 None.

63996 RATIONALE

63997 None.

63998 FUTURE DIRECTIONS

63999 None.

64000 SEE ALSO

64001 [strncmp\(\)](#)

64002 XBD [<string.h>](#)

64003 CHANGE HISTORY

64004 First released in Issue 1. Derived from Issue 1 of the SVID.

64005 Issue 6

64006 Extensions beyond the ISO C standard are marked.

64007 **NAME**

64008 strcoll, strcoll_l — string comparison using collating information

64009 **SYNOPSIS**

```
64010 #include <string.h>
64011 int strcoll(const char *s1, const char *s2);
64012 CX int strcoll_l(const char *s1, const char *s2,
64013 locale_t locale);
```

64014 **DESCRIPTION**

64015 CX For *strcoll()*: The functionality described on this reference page is aligned with the ISO C
 64016 standard. Any conflict between the requirements described here and the ISO C standard is
 64017 unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

64018 CX The *strcoll()* and *strcoll_l()* functions shall compare the string pointed to by *s1* to the string
 64019 pointed to by *s2*, both interpreted as appropriate to the *LC_COLLATE* category of the current
 64020 locale, or of the locale represented by *locale*, respectively.

64021 CX The *strcoll()* and *strcoll_l()* functions shall not change the setting of *errno* if successful.

64022 Since no return value is reserved to indicate an error, an application wishing to check for error
 64023 CX situations should set *errno* to 0, then call *strcoll()*, or *strcoll_l()* then check *errno*.

64024 CX The behavior is undefined if the *locale* argument to *strcoll_l()* is the special locale object +
 64025 LC_GLOBAL_LOCALE or is not a valid locale object handle.

64026 **RETURN VALUE**

64027 Upon successful completion, *strcoll()* shall return an integer greater than, equal to, or less than 0,
 64028 according to whether the string pointed to by *s1* is greater than, equal to, or less than the string
 64029 CX pointed to by *s2* when both are interpreted as appropriate to the current locale. On error,
 64030 *strcoll()* may set *errno*, but no return value is reserved to indicate an error.

64031 Upon successful completion, *strcoll_l()* shall return an integer greater than, equal to, or less than
 64032 0, according to whether the string pointed to by *s1* is greater than, equal to, or less than the
 64033 string pointed to by *s2* when both are interpreted as appropriate to the locale represented by
 64034 *locale*. On error, *strcoll_l()* may set *errno*, but no return value is reserved to indicate an error.

64035 **ERRORS**

64036 These functions may fail if:

64037 CX [EINVAL] The *s1* or *s2* arguments contain characters outside the domain of the collating
 64038 sequence.

64039 **EXAMPLES**64040 **Comparing Nodes**

64041 The following example uses an application-defined function, *node_compare()*, to compare two
 64042 nodes based on an alphabetical ordering of the *string* field.

```
64043 #include <string.h>
64044 ...
64045 struct node { /* These are stored in the table. */
64046     char *string;
64047     int length;
64048 };
64049 ...
```

```

64050     int node_compare(const void *node1, const void *node2)
64051     {
64052         return strcoll(((const struct node *)node1)->string,
64053                        ((const struct node *)node2)->string);
64054     }
64055     ...

```

APPLICATION USAGE

The *strxfrm()* and *strcmp()* functions should be used for sorting large lists.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

alphasort(), *strcmp()*, *strxfrm()*

XBD <string.h>

CHANGE HISTORY

First released in Issue 3.

Issue 5

The DESCRIPTION is updated to indicate that *errno* does not change if the function is successful.

Issue 6

Extensions beyond the ISO C standard are marked.

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The [EINVAL] optional error condition is added.

An example is added.

Issue 7

The *strcoll_l()* function is added from The Open Group Technical Standard, 2006, Extended API Set Part 4.

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0593 [283] and XSH/TC1-2008/0594 + [283] are applied.

64080 **NAME**

64081 stpcpy, strcpy — copy a string and return a pointer to the end of the result

64082 **SYNOPSIS**

64083 #include <string.h>

64084 CX char *stpcpy(char *restrict s1, const char *restrict s2);

64085 char *strcpy(char *restrict s1, const char *restrict s2);

64086 **DESCRIPTION**64087 CX For *stpcpy()*: The functionality described on this reference page is aligned with the ISO C
64088 standard. Any conflict between the requirements described here and the ISO C standard is
64089 unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.64090 CX The *stpcpy()* and *strcpy()* functions shall copy the string pointed to by s2 (including the
64091 terminating NUL character) into the array pointed to by s1.

64092 If copying takes place between objects that overlap, the behavior is undefined.

64093 **RETURN VALUE**64094 CX The *stpcpy()* function shall return a pointer to the terminating NUL character copied into the s1
64095 buffer.64096 The *strcpy()* function shall return s1.

64097 No return values are reserved to indicate an error.

64098 **ERRORS**

64099 No errors are defined.

64100 **EXAMPLES**64101 **Construction of a Multi-Part Message in a Single Buffer**

64102 #include <string.h>

64103 #include <stdio.h>

64104 int

64105 main (void)

64106 {

64107 char buffer [10];

64108 char *name = buffer;

64109 name = stpcpy (stpcpy (stpcpy (name, "ice"), "-"), "cream");

64110 puts (buffer);

64111 return 0;

64112 }

64113 **Initializing a String**64114 The following example copies the string "-----" into the *permstring* variable.

64115 #include <string.h>

64116 ...

64117 static char permstring[11];

64118 ...

64119 strcpy(permstring, "-----");

64120 ...

Storing a Key and Data

The following example allocates space for a key using *malloc()* then uses *strcpy()* to place the key there. Then it allocates space for data using *malloc()*, and uses *strcpy()* to place data there. (The user-defined function *dbfree()* frees memory previously allocated to an array of type **struct element** *.)

```
#include <string.h>
#include <stdlib.h>
#include <stdio.h>
...
/* Structure used to read data and store it. */
struct element {
    char *key;
    char *data;
};

struct element *tbl, *curtbl;
char *key, *data;
int count;
...
void dbfree(struct element *, int);
...
if ((curtbl->key = malloc(strlen(key) + 1)) == NULL) {
    perror("malloc"); dbfree(tbl, count); return NULL;
}
strcpy(curtbl->key, key);

if ((curtbl->data = malloc(strlen(data) + 1)) == NULL) {
    perror("malloc"); free(curtbl->key); dbfree(tbl, count); return NULL;
}
strcpy(curtbl->data, data);
...
```

APPLICATION USAGE

Character movement is performed differently in different implementations. Thus, overlapping moves may yield surprises.

This version is aligned with the ISO C standard; this does not affect compatibility with XPG3 applications. Reliable error detection by this function was never guaranteed.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

strncpy(), *wcscpy()*

XBD **<string.h>**

CHANGE HISTORY

First released in Issue 1. Derived from Issue 1 of the SVID.

64164 **Issue 6**

64165 The *strcpy()* prototype is updated for alignment with the ISO/IEC 9899:1999 standard.

64166 **Issue 7**

64167 The *strcpy()* function is added from The Open Group Technical Standard, 2006, Extended API
64168 Set Part 1.

64169 **NAME**

64170 strcspn — get the length of a complementary substring

64171 **SYNOPSIS**

64172 #include <string.h>

64173 size_t strcspn(const char *s1, const char *s2);

64174 **DESCRIPTION**

64175 CX The functionality described on this reference page is aligned with the ISO C standard. Any
 64176 conflict between the requirements described here and the ISO C standard is unintentional. This
 64177 volume of POSIX.1-2008 defers to the ISO C standard.

64178 The *strcspn()* function shall compute the length (in bytes) of the maximum initial segment of the
 64179 string pointed to by *s1* which consists entirely of bytes *not* from the string pointed to by *s2*.

64180 **RETURN VALUE**

64181 The *strcspn()* function shall return the length of the computed segment of the string pointed to
 64182 by *s1*; no return value is reserved to indicate an error.

64183 **ERRORS**

64184 No errors are defined.

64185 **EXAMPLES**

64186 None.

64187 **APPLICATION USAGE**

64188 None.

64189 **RATIONALE**

64190 None.

64191 **FUTURE DIRECTIONS**

64192 None.

64193 **SEE ALSO**64194 [strspn\(\)](#)64195 XBD [<string.h>](#)64196 **CHANGE HISTORY**

64197 First released in Issue 1. Derived from Issue 1 of the SVID.

64198 **Issue 5**

64199 The RETURN VALUE section is updated to indicate that *strcspn()* returns the length of *s1*, and
 64200 not *s1* itself as was previously stated.

64201 **Issue 6**

64202 The Open Group Corrigendum U030/1 is applied. The text of the RETURN VALUE section is
 64203 updated to indicate that the computed segment length is returned, not the *s1* length.

NAME

strdup, strndup — duplicate a specific number of bytes from a string

SYNOPSIS

```
CX      #include <string.h>
char *strdup(const char *s);
char *strndup(const char *s, size_t size);
```

DESCRIPTION

The *strdup()* function shall return a pointer to a new string, which is a duplicate of the string pointed to by *s*. The returned pointer can be passed to *free()*. A null pointer is returned if the new string cannot be created.

The *strndup()* function shall be equivalent to the *strdup()* function, duplicating the provided *s* in a new block of memory allocated as if by using *malloc()*, with the exception being that *strndup()* copies at most *size* plus one bytes into the newly allocated memory, terminating the new string with a NUL character. If the length of *s* is larger than *size*, only *size* bytes shall be duplicated. If *size* is larger than the length of *s*, all bytes in *s* shall be copied into the new memory buffer, including the terminating NUL character. The newly created string shall always be properly terminated.

RETURN VALUE

The *strdup()* function shall return a pointer to a new string on success. Otherwise, it shall return a null pointer and set *errno* to indicate the error.

Upon successful completion, the *strndup()* function shall return a pointer to the newly allocated memory containing the duplicated string. Otherwise, it shall return a null pointer and set *errno* to indicate the error.

ERRORS

These functions shall fail if:

[ENOMEM] Storage space available is insufficient.

EXAMPLES

None.

APPLICATION USAGE

For functions that allocate memory as if by *malloc()*, the application should release such memory when it is no longer required by a call to *free()*. For *strdup()* and *strndup()*, this is the return value.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

free(), *wcsdup()*

XBD *<string.h>*

CHANGE HISTORY

First released in Issue 4, Version 2.

64245 **Issue 5**

64246 Moved from X/OPEN UNIX extension to BASE.

64247 **Issue 7**64248 Austin Group Interpretation 1003.1-2001 #044 is applied, changing the “may fail” [ENOMEM]
64249 error to become a “shall fail” error.64250 The *strdup()* function is moved from the XSI option to the Base.64251 The *strndup()* function is added from The Open Group Technical Standard, 2006, Extended API
64252 Set Part 1.64253 The APPLICATION USAGE section is updated to clarify that memory is allocated as if by
64254 *malloc()*.

64255 NAME

64256 strerror, strerror_l, strerror_r — get error message string

64257 SYNOPSIS

64258 #include <string.h>

64259 char *strerror(int *errnum*);64260 CX char *strerror_l(int *errnum*, locale_t *locale*);64261 int strerror_r(int *errnum*, char **strerrbuf*, size_t *buflen*);

64262 DESCRIPTION

64263 CX For *strerror()*: The functionality described on this reference page is aligned with the ISO C
 64264 standard. Any conflict between the requirements described here and the ISO C standard is
 64265 unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

64266 The *strerror()* function shall map the error number in *errnum* to a locale-dependent error
 64267 message string and shall return a pointer to it. Typically, the values for *errnum* come from *errno*,
 64268 but *strerror()* shall map any value of type *int* to a message.

64269 CX The application shall not modify the string returned. The returned string pointer might be
 64270 CX invalidated or the string content might be overwritten by a subsequent call to *strerror()*, or by a
 64271 subsequent call to *strerror_l()* in the same thread.

64272 CX The string may be overwritten by a subsequent call to *strerror_l()* in the same thread.

64273 The contents of the error message strings returned by *strerror()* should be determined by the
 64274 setting of the *LC_MESSAGES* category in the current locale.

64275 The implementation shall behave as if no function defined in this volume of POSIX.1-2008 calls
 64276 *strerror()*.

64277 CX The *strerror()* and *strerror_l()* functions shall not change the setting of *errno* if successful.

64278 Since no return value is reserved to indicate an error of *strerror()*, an application wishing to
 64279 check for error situations should set *errno* to 0, then call *strerror()*, then check *errno*. Similarly,
 64280 since *strerror_l()* is required to return a string for some errors, an application wishing to check
 64281 for all error situations should set *errno* to 0, then call *strerror_l()*, then check *errno*.

64282 The *strerror()* function need not be thread-safe.

64283 The *strerror_l()* function shall map the error number in *errnum* to a locale-dependent error
 64284 message string in the locale represented by *locale* and shall return a pointer to it.

64285 The *strerror_r()* function shall map the error number in *errnum* to a locale-dependent error
 64286 message string and shall return the string in the buffer pointed to by *strerrbuf*, with length
 64287 *buflen*.

64288 CX If the value of *errnum* is a valid error number, the message string shall indicate what error
 64289 occurred; if the value of *errnum* is zero, the message string shall either be an empty string or
 64290 indicate that no error occurred; otherwise, if these functions complete successfully, the message
 64291 string shall indicate that an unknown error occurred.

64292 CX The behavior is undefined if the *locale* argument to *strerror_l()* is the special locale object
 64293 *LC_GLOBAL_LOCALE* or is not a valid locale object handle.

64294 RETURN VALUE

64295 Upon completion, whether successful or not, *strerror()* shall return a pointer to the generated
 64296 CX message string. On error *errno* may be set, but no return value is reserved to indicate an error.

64297 Upon successful completion, *strerror_l()* shall return a pointer to the generated message string. If

64298 *errnum* is not a valid error number, *errno* may be set to [EINVAL], but a pointer to a message
 64299 string shall still be returned. If any other error occurs, *errno* shall be set to indicate the error and
 64300 a null pointer shall be returned.

64301 Upon successful completion, *strerror_r()* shall return 0. Otherwise, an error number shall be
 64302 returned to indicate the error.

64303 ERRORS

64304 These functions may fail if:

64305 CX [EINVAL] The value of *errnum* is neither a valid error number nor zero.

64306 The *strerror_r()* function may fail if:

64307 CX [ERANGE] Insufficient storage was supplied via *strrbuf* and *buflen* to contain the
 64308 generated message string.

64309 EXAMPLES

64310 None.

64311 APPLICATION USAGE

64312 Historically in some implementations, calls to *perror()* would overwrite the string that the
 64313 pointer returned by *strerror()* points to. Such implementations did not conform to the ISO C
 64314 standard; however, application developers should be aware of this behavior if they wish their
 64315 applications to be portable to such implementations.

64316 RATIONALE

64317 The *strerror_l()* function is required to be thread-safe, thereby eliminating the need for an
 64318 equivalent to the *strerror_r()* function.

64319 Earlier versions of this standard did not explicitly require that the error message strings returned
 64320 by *strerror()* and *strerror_r()* provide any information about the error. This version of the
 64321 standard requires a meaningful message for any successful completion.

64322 Since no return value is reserved to indicate a *strerror()* error, but all calls (whether successful or
 64323 not) must return a pointer to a message string, on error *strerror()* can return a pointer to an
 64324 empty string or a pointer to a meaningful string that can be printed.

64325 Note that the [EINVAL] error condition is a may fail error. If an invalid error number is supplied
 64326 as the value of *errnum*, applications should be prepared to handle any of the following:

- 64327 1. Error (with no meaningful message): *errno* is set to [EINVAL], the return value is a pointer
 64328 to an empty string.
- 64329 2. Successful completion: *errno* is unchanged and the return value points to a string like
 64330 "unknown error" or "error number xxx" (where xxx is the value of *errnum*).
- 64331 3. Combination of #1 and #2: *errno* is set to [EINVAL] and the return value points to a string
 64332 like "unknown error" or "error number xxx" (where xxx is the value of *errnum*).
 64333 Since applications frequently use the return value of *strerror()* as an argument to
 64334 functions like *fprintf()* (without checking the return value) and since applications have no
 64335 way to parse an error message string to determine whether *errnum* represents a valid
 64336 error number, implementations are encouraged to implement #3. Similarly,
 64337 implementations are encouraged to have *strerror_r()* return [EINVAL] and put a string
 64338 like "unknown error" or "error number xxx" in the buffer pointed to by *strrbuf*
 64339 when the value of *errnum* is not a valid error number.

64340 Some applications rely on being able to set *errno* to 0 before calling a function with no reserved +
 64341 value to indicate an error, then call *strerror(errno)* afterwards to detect whether an error occurred +

64342 (because *errno* changed) or to indicate success (because *errno* remained zero). This usage pattern +
 64343 requires that *strerror*(0) succeed with useful results. Previous versions of the standard did not +
 64344 specify the behavior when *errnum* is zero.

64345 FUTURE DIRECTIONS

64346 None.

64347 SEE ALSO

64348 *perror*()

64349 XBD <string.h>

64350 CHANGE HISTORY

64351 First released in Issue 3.

64352 Issue 5

64353 The DESCRIPTION is updated to indicate that *errno* is not changed if the function is successful.

64354 A note indicating that the *strerror*() function need not be reentrant is added to the
 64355 DESCRIPTION.

64356 Issue 6

64357 Extensions beyond the ISO C standard are marked.

64358 The following new requirements on POSIX implementations derive from alignment with the
 64359 Single UNIX Specification:

- 64360 • In the RETURN VALUE section, the fact that *errno* may be set is added.
- 64361 • The [EINVAL] optional error condition is added.

64362 The normative text is updated to avoid use of the term “must” for application requirements.

64363 The *strerror_r*() function is added in response to IEEE PASC Interpretation 1003.1c #39.

64364 The *strerror_r*() function is marked as part of the Thread-Safe Functions option.

64365 Issue 7

64366 Austin Group Interpretation 1003.1-2001 #072 is applied, updating the ERRORS section.

64367 Austin Group Interpretation 1003.1-2001 #156 is applied.

64368 Austin Group Interpretation 1003.1-2001 #187 is applied, clarifying the behavior when the
 64369 generated error message is an empty string.

64370 SD5-XSH-ERN-191 is applied, updating the APPLICATION USAGE section.

64371 The *strerror_l*() function is added from The Open Group Technical Standard, 2006, Extended API
 64372 Set Part 4.

64373 The *strerror_r*() function is moved from the Thread-Safe Functions option to the Base.

64374 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0595 [75], XSH/TC1-2008/0596 [447], +
 64375 XSH/TC1-2008/0597 [382,428], XSH/TC1-2008/0598 [283], XSH/TC1-2008/0599 [382,428], +
 64376 XSH/TC1-2008/0600 [283], and XSH/TC1-2008/0601 [382,428] are applied.

NAME

strfmon, strfmon_l — convert monetary value to a string

SYNOPSIS

```
#include <monetary.h>

ssize_t strfmon(char *restrict s, size_t maxsize,
               const char *restrict format, ...);
ssize_t strfmon_l(char *restrict s, size_t maxsize,
                 locale_t locale, const char *restrict format, ...);
```

DESCRIPTION

The *strfmon()* function shall place characters into the array pointed to by *s* as controlled by the string pointed to by *format*. No more than *maxsize* bytes are placed into the array.

The format is a character string, beginning and ending in its initial state, if any, that contains two types of objects: *plain characters*, which are simply copied to the output stream, and *conversion specifications*, each of which shall result in the fetching of zero or more arguments which are converted and formatted. The results are undefined if there are insufficient arguments for the format. If the format is exhausted while arguments remain, the excess arguments are simply ignored.

The application shall ensure that a conversion specification consists of the following sequence:

- A '%' character
- Optional flags
- Optional field width
- Optional left precision
- Optional right precision
- A required conversion specifier character that determines the conversion to be performed

The *strfmon_l()* function shall be equivalent to the *strfmon()* function, except that the locale data used is from the locale represented by *locale*.

Flags

One or more of the following optional flags can be specified to control the conversion:

- =f* An '=' followed by a single character *f* which is used as the numeric fill character. In order to work with precision or width counts, the fill character shall be a single byte character; if not, the behavior is undefined. The default numeric fill character is the <space>. This flag does not affect field width filling which always uses the <space>. This flag is ignored unless a left precision (see below) is specified.
- ^* Do not format the currency amount with grouping characters. The default is to insert the grouping characters if defined for the current locale.
- + or (* Specify the style of representing positive and negative currency amounts. Only one of '+' or '(' may be specified. If '+' is specified, the locale's equivalent of '+' and '-' are used (for example, in many locales, the empty string if positive and '-' if negative). If '(' is specified, negative amounts are enclosed within parentheses. If neither flag is specified, the '+' style is used.
- !* Suppress the currency symbol from the output conversion.

64418 – Specify the alignment. If this flag is present the result of the conversion is left-justified
 64419 (padded to the right) rather than right-justified. This flag shall be ignored unless a field
 64420 width (see below) is specified.

64421 Field Width

64422 *w* A decimal digit string *w* specifying a minimum field width in bytes in which the result
 64423 of the conversion is right-justified (or left-justified if the flag '*-*' is specified). The
 64424 default is 0.

64425 Left Precision

64426 *#n* A '*#*' followed by a decimal digit string *n* specifying a maximum number of digits
 64427 expected to be formatted to the left of the radix character. This option can be used to
 64428 keep the formatted output from multiple calls to the *strfmon()* function aligned in the
 64429 same columns. It can also be used to fill unused positions with a special character as in
 64430 "\$***123.45". This option causes an amount to be formatted as if it has the number
 64431 of digits specified by *n*. If more than *n* digit positions are required, this conversion
 64432 specification is ignored. Digit positions in excess of those actually required are filled
 64433 with the numeric fill character (see the *=f* flag above).

64434 If grouping has not been suppressed with the '*^*' flag, and it is defined for the current
 64435 locale, grouping separators are inserted before the fill characters (if any) are added.
 64436 Grouping separators are not applied to fill characters even if the fill character is a digit.

64437 To ensure alignment, any characters appearing before or after the number in the
 64438 formatted output such as currency or sign symbols are padded as necessary with
 64439 <space> characters to make their positive and negative formats an equal length.

64440 Right Precision

64441 *.p* A <period> followed by a decimal digit string *p* specifying the number of digits after
 64442 the radix character. If the value of the right precision *p* is 0, no radix character appears.
 64443 If a right precision is not included, a default specified by the current locale is used. The
 64444 amount being formatted is rounded to the specified number of digits prior to
 64445 formatting.

64446 Conversion Specifier Characters

64447 The conversion specifier characters and their meanings are:

64448 *i* The **double** argument is formatted according to the locale's international currency
 64449 format (for example, in the US: USD 1,234.56). If the argument is $\pm\text{Inf}$ or NaN, the result
 64450 of the conversion is unspecified.

64451 *n* The **double** argument is formatted according to the locale's national currency format
 64452 (for example, in the US: \$1,234.56). If the argument is $\pm\text{Inf}$ or NaN, the result of the
 64453 conversion is unspecified.

64454 *%* Convert to a '*%*'; no argument is converted. The entire conversion specification shall
 64455 be *%%*.

Locale Information

The `LC_MONETARY` category of the current locale affects the behavior of this function including the monetary radix character (which may be different from the numeric radix character affected by the `LC_NUMERIC` category), the grouping separator, the currency symbols, and formats. The international currency symbol should be conformant with the ISO 4217:2001 standard.

If the value of `maxsize` is greater than `{SSIZE_MAX}`, the result is implementation-defined.

The behavior is undefined if the `locale` argument to `strfmon_l()` is the special locale object `LC_GLOBAL_LOCALE` or is not a valid locale object handle.

RETURN VALUE

If the total number of resulting bytes including the terminating null byte is not more than `maxsize`, these functions shall return the number of bytes placed into the array pointed to by `s`, not including the terminating NUL character. Otherwise, `-1` shall be returned, the contents of the array are unspecified, and `errno` shall be set to indicate the error.

ERRORS

These functions shall fail if:

[E2BIG] Conversion stopped due to lack of space in the buffer.

EXAMPLES

Given a locale for the US and the values 123.45, -123.45, and 3456.781, the following output might be produced. Square brackets ("`[ ]`") are used in this example to delimit the output.

<code>%n</code>	<code>[\$123.45]</code> <code>[-\$123.45]</code> <code>[\$3,456.78]</code>	Default formatting
<code>%11n</code>	<code>[\$123.45]</code> <code>[-\$123.45]</code> <code>[\$3,456.78]</code>	Right align within an 11-character field
<code>%#5n</code>	<code>[\$ 123.45]</code> <code>[-\$ 123.45]</code> <code>[\$ 3,456.78]</code>	Aligned columns for values up to 99 999
<code>%=*#5n</code>	<code>[\$***123.45]</code> <code>[-\$***123.45]</code> <code>[\$*3,456.78]</code>	Specify a fill character
<code>%=0#5n</code>	<code>[\$000123.45]</code> <code>[-\$000123.45]</code> <code>[\$03,456.78]</code>	Fill characters do not use grouping even if the fill character is a digit
<code>%^#5n</code>	<code>[\$ 123.45]</code> <code>[-\$ 123.45]</code> <code>[\$ 3456.78]</code>	Disable the grouping separator
<code>%^#5.0n</code>	<code>[\$ 123]</code> <code>[-\$ 123]</code> <code>[\$ 3457]</code>	Round off to whole units
<code>%^#5.4n</code>	<code>[\$ 123.4500]</code> <code>[-\$ 123.4500]</code> <code>[\$ 3456.7810]</code>	Increase the precision
<code>%(#5n</code>	<code>[\$ 123.45]</code>	Use an alternative pos/neg style

```

64500          [ ($ 123.45) ]
64501          [ $ 3,456.78 ]

64502      %! (#5n  [ 123.45 ]      Disable the currency symbol
64503          [ ( 123.45) ]
64504          [ 3,456.78 ]

64505      %-14#5.4n [ $ 123.4500 ]      Left-justify the output
64506          [-$ 123.4500 ]
64507          [ $ 3,456.7810 ]

64508      %14#5.4n  [ $ 123.4500]      Corresponding right-justified output
64509          [-$ 123.4500]
64510          [ $ 3,456.7810]

```

64511 See also the EXAMPLES section in *fprintf()*.

64512 APPLICATION USAGE

64513 None.

64514 RATIONALE

64515 None.

64516 FUTURE DIRECTIONS

64517 Lowercase conversion characters are reserved for future standards use and uppercase for
 64518 implementation-defined use.

64519 SEE ALSO

64520 *fprintf()*, *localeconv()*

64521 XBD <monetary.h>

64522 CHANGE HISTORY

64523 First released in Issue 4.

64524 Issue 5

64525 Moved from ENHANCED I18N to BASE.

64526 The [ENOSYS] error is removed.

64527 Text is added to the DESCRIPTION warning about values of *maxsize* that are greater than
 64528 {SSIZE_MAX}.

64529 Issue 6

64530 The normative text is updated to avoid use of the term “must” for application requirements.

64531 The **restrict** keyword is added to the *strfmon()* prototype for alignment with the
 64532 ISO/IEC 9899:1999 standard.

64533 The EXAMPLES section is reworked, clarifying the output format.

64534 Issue 7

64535 SD5-XSH-ERN-29 is applied, updating the examples for % (#5n and %! (#5n.

64536 SD5-XSH-ERN-233 is applied, changing the definition of the '+' or '(' flags to refer to
 64537 multiple locales.

64538 The *strfmon()* function is moved from the XSI option to the Base.

64539 The *strfmon_l()* function is added from The Open Group Technical Standard, 2006, Extended API
64540 Set Part 4.

64541 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0602 [302], XSH/TC1-2008/0603 [283], +
64542 and XSH/TC1-2008/0604 [283] are applied.

64543 **NAME**64544 `strftime, strftime_l` — convert date and time to a string64545 **SYNOPSIS**

```
64546 #include <time.h>
64547 size_t strftime(char *restrict s, size_t maxsize,
64548               const char *restrict format, const struct tm *restrict timeptr);
64549 CX size_t strftime_l(char *restrict s, size_t maxsize,
64550                  const char *restrict format, const struct tm *restrict timeptr,
64551                  locale_t locale);
```

64552 **DESCRIPTION**

64553 CX For `strftime()`: The functionality described on this reference page is aligned with the ISO C
 64554 standard. Any conflict between the requirements described here and the ISO C standard is
 64555 unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

64556 The `strftime()` function shall place bytes into the array pointed to by *s* as controlled by the string
 64557 pointed to by *format*. The format is a character string, beginning and ending in its initial shift
 64558 state, if any. The format string consists of zero or more conversion specifications and ordinary
 64559 characters.

64560 Each conversion specification is introduced by the `'%'` character after which the following
 64561 appear in sequence:

- 64562 CX
- An optional flag:
 - 64563 0 The zero character (`'0'`), which specifies that the character used as the padding
 64564 character is `'0'`,
 - 64565 + The `<plus-sign>` character (`'+'`), which specifies that the character used as the
 64566 padding character is `'0'`, and that if and only if the field being produced consumes
 64567 more than four bytes to represent a year (for `%F`, `%G`, or `%Y`) or more than two bytes to
 64568 represent the year divided by 100 (for `%C`) then a leading `<plus-sign>` character shall
 64569 be included if the year being processed is greater than or equal to zero or a leading
 64570 minus-sign character (`'-'`) shall be included if the year is less than zero.
 - 64571 The default padding character is unspecified.
 - An optional minimum field width. If the converted value, including any leading `'+'` or
 64572 `'-'` sign, has fewer bytes than the minimum field width and the padding character is not
 64573 the NUL character, the output shall be padded on the left (after any leading `'+'` or `'-'`
 64574 sign) with the padding character.
 - 64575
 - An optional `E` or `O` modifier.
 - 64576
 - A terminating conversion specifier character that indicates the type of conversion to be
 64577 applied.
 - 64578

64579 CX The results are unspecified if more than one flag character is specified, a flag character is
 64580 specified without a minimum field width; a minimum field width is specified without a flag
 64581 character; a modifier is specified with a flag or with a minimum field width; or if a minimum
 64582 field width is specified for any conversion specifier other than `C`, `F`, `G`, or `Y`.

64583 All ordinary characters (including the terminating NUL character) are copied unchanged into
 64584 the array. If copying takes place between objects that overlap, the behavior is undefined. No
 64585 more than *maxsize* bytes are placed into the array. Each conversion specifier is replaced by
 64586 appropriate characters as described in the following list. The appropriate characters are

determined using the *LC_TIME* category of the current locale and by the values of zero or more members of the broken-down time structure pointed to by *timeptr*, as specified in brackets in the description. If any of the specified values are outside the normal range, the characters stored are unspecified.

64591 CX The *strptime_l()* function shall be equivalent to the *strptime()* function, except that the locale data used is from the locale represented by *locale*.

64592

64593 Local timezone information is used as though *strptime()* called *tzset()*.

64594 The following conversion specifiers shall be supported:

64595 a Replaced by the locale's abbreviated weekday name. [*tm_wday*]

64596 A Replaced by the locale's full weekday name. [*tm_wday*]

64597 b Replaced by the locale's abbreviated month name. [*tm_mon*]

64598 B Replaced by the locale's full month name. [*tm_mon*]

64599 c Replaced by the locale's appropriate date and time representation. (See the Base Definitions volume of POSIX.1-2008, <**time.h**>.)

64600

64601 C Replaced by the year divided by 100 and truncated to an integer, as a decimal number.

64602 [*tm_year*]

64603 If a minimum field width is not specified, the number of characters placed into the

64604 array pointed to by *s* will be the number of digits in the year divided by 100 or two,

64605 CX whichever is greater. If a minimum field width is specified, the number of characters

64606 placed into the array pointed to by *s* will be the number of digits in the year divided by

64607 100 or the minimum field width, whichever is greater.

64608 d Replaced by the day of the month as a decimal number [01,31]. [*tm_mday*]

64609 D Equivalent to %m/%d/%Y. [*tm_mon, tm_mday, tm_year*]

64610 e Replaced by the day of the month as a decimal number [1,31]; a single digit is preceded

64611 by a space. [*tm_mday*]

64612 CX F Equivalent to %⁺⁴Y-%m-%d if no flag and no minimum field width are specified.

64613 [*tm_year, tm_mon, tm_mday*]

64614 CX If a minimum field width of *x* is specified, the year shall be output as if by the Y

64615 specifier (described below) with whatever flag was given and a minimum field width

64616 of *x*−6. If *x* is less than 6, the behavior shall be as if *x* equalled 6.

64617 If the minimum field width is specified to be 10, and the year is four digits long, then

64618 the output string produced will match the ISO 8601:2004 standard subclause 4.1.2.2

64619 complete representation, extended format date representation of a specific day. If a +

64620 flag is specified, a minimum field width of *x* is specified, and *x*−7 bytes are sufficient to

64621 hold the digits of the year (not including any needed sign character), then the output

64622 will match the ISO 8601:2004 standard subclause 4.1.2.4 complete representation,

64623 expanded format date representation of a specific day.

64624 g Replaced by the last 2 digits of the week-based year (see below) as a decimal number

64625 [00,99]. [*tm_year, tm_wday, tm_yday*]

64626 G Replaced by the week-based year (see below) as a decimal number (for example, 1977).

64627 [*tm_year, tm_wday, tm_yday*]

64628	CX	If a minimum field width is specified, the number of characters placed into the array pointed to by <i>s</i> will be the number of digits and leading sign characters (if any) in the year, or the minimum field width, whichever is greater.
64629		
64630		
64631	h	Equivalent to %b. [<i>tm_mon</i>]
64632	H	Replaced by the hour (24-hour clock) as a decimal number [00,23]. [<i>tm_hour</i>]
64633	I	Replaced by the hour (12-hour clock) as a decimal number [01,12]. [<i>tm_hour</i>]
64634	j	Replaced by the day of the year as a decimal number [001,366]. [<i>tm_yday</i>]
64635	m	Replaced by the month as a decimal number [01,12]. [<i>tm_mon</i>]
64636	M	Replaced by the minute as a decimal number [00,59]. [<i>tm_min</i>]
64637	n	Replaced by a <newline>.
64638	p	Replaced by the locale's equivalent of either a.m. or p.m. [<i>tm_hour</i>]
64639	CX	Replaced by the time in a.m. and p.m. notation; in the POSIX locale this shall be equivalent to %I:%M:%S %p. [<i>tm_hour</i> , <i>tm_min</i> , <i>tm_sec</i>]
64640		
64641	R	Replaced by the time in 24-hour notation (%H:%M). [<i>tm_hour</i> , <i>tm_min</i>]
64642	S	Replaced by the second as a decimal number [00,60]. [<i>tm_sec</i>]
64643	t	Replaced by a <tab>.
64644	T	Replaced by the time (%H:%M:%S). [<i>tm_hour</i> , <i>tm_min</i> , <i>tm_sec</i>]
64645	u	Replaced by the weekday as a decimal number [1,7], with 1 representing Monday. [<i>tm_wday</i>]
64646		
64647	U	Replaced by the week number of the year as a decimal number [00,53]. The first Sunday of January is the first day of week 1; days in the new year before this are in week 0. [<i>tm_year</i> , <i>tm_wday</i> , <i>tm_yday</i>]
64648		
64649		
64650	V	Replaced by the week number of the year (Monday as the first day of the week) as a decimal number [01,53]. If the week containing 1 January has four or more days in the new year, then it is considered week 1. Otherwise, it is the last week of the previous year, and the next week is week 1. Both January 4th and the first Thursday of January are always in week 1. [<i>tm_year</i> , <i>tm_wday</i> , <i>tm_yday</i>]
64651		
64652		
64653		
64654		
64655	w	Replaced by the weekday as a decimal number [0,6], with 0 representing Sunday. [<i>tm_wday</i>]
64656		
64657	W	Replaced by the week number of the year as a decimal number [00,53]. The first Monday of January is the first day of week 1; days in the new year before this are in week 0. [<i>tm_year</i> , <i>tm_wday</i> , <i>tm_yday</i>]
64658		
64659		
64660	x	Replaced by the locale's appropriate date representation. (See the Base Definitions volume of POSIX.1-2008, <time.h>.)
64661		
64662	X	Replaced by the locale's appropriate time representation. (See the Base Definitions volume of POSIX.1-2008, <time.h>.)
64663		
64664	y	Replaced by the last two digits of the year as a decimal number [00,99]. [<i>tm_year</i>]
64665	Y	Replaced by the year as a decimal number (for example, 1997). [<i>tm_year</i>]
64666	CX	If a minimum field width is specified, the number of characters placed into the array pointed to by <i>s</i> will be the number of digits and leading sign characters (if any) in the
64667		

64668		year, or the minimum field width, whichever is greater.
64669	z	Replaced by the offset from UTC in the ISO 8601:2004 standard format (+hhmm or -hhmm), or by no characters if no timezone is determinable. For example, "-0430"
64670		means 4 hours 30 minutes behind UTC (west of Greenwich). If <i>tm_isdst</i> is zero, the
64671	CX	standard time offset is used. If <i>tm_isdst</i> is greater than zero, the daylight savings time
64672		offset is used. If <i>tm_isdst</i> is negative, no characters are returned. [<i>tm_isdst</i>]
64673		
64674	Z	Replaced by the timezone name or abbreviation, or by no bytes if no timezone
64675		information exists. [<i>tm_isdst</i>]
64676	%	Replaced by %.
64677		If a conversion specification does not correspond to any of the above, the behavior is undefined.
64678	CX	If a struct tm broken-down time structure is created by <i>localtime()</i> or <i>localtime_r()</i> , or modified
64679		by <i>mktime()</i> , and the value of <i>TZ</i> is subsequently modified, the results of the %Z and %z
64680		<i>strptime()</i> conversion specifiers are undefined, when <i>strptime()</i> is called with such a broken-down
64681		time structure.
64682		If a struct tm broken-down time structure is created or modified by <i>gmtime()</i> or <i>gmtime_r()</i> , it is
64683		unspecified whether the result of the %Z and %z conversion specifiers shall refer to UTC or the
64684		current local timezone, when <i>strptime()</i> is called with such a broken-down time structure.
64685		Modified Conversion Specifiers
64686		Some conversion specifiers can be modified by the E or O modifier characters to indicate that an
64687		alternative format or specification should be used rather than the one normally used by the
64688		unmodified conversion specifier. If the alternative format or specification does not exist for the
64689		current locale (see ERA in XBD Section 7.3.5 , on page 159), the behavior shall be as if the
64690		unmodified conversion specification were used.
64691	%Ec	Replaced by the locale's alternative appropriate date and time representation.
64692	%EC	Replaced by the name of the base year (period) in the locale's alternative
64693		representation.
64694	%Ex	Replaced by the locale's alternative date representation.
64695	%EX	Replaced by the locale's alternative time representation.
64696	%Ey	Replaced by the offset from %EC (year only) in the locale's alternative representation.
64697	%EY	Replaced by the full alternative year representation.
64698	%Od	Replaced by the day of the month, using the locale's alternative numeric symbols, filled
64699		as needed with leading zeros if there is any alternative symbol for zero; otherwise, with
64700		leading <space> characters.
64701	%Oe	Replaced by the day of the month, using the locale's alternative numeric symbols, filled
64702		as needed with leading <space> characters.
64703	%OH	Replaced by the hour (24-hour clock) using the locale's alternative numeric symbols.
64704	%OI	Replaced by the hour (12-hour clock) using the locale's alternative numeric symbols.
64705	%Om	Replaced by the month using the locale's alternative numeric symbols.
64706	%OM	Replaced by the minutes using the locale's alternative numeric symbols.

64707 %OS Replaced by the seconds using the locale's alternative numeric symbols.

64708 %Ou Replaced by the weekday as a number in the locale's alternative representation
64709 (Monday=1).

64710 %OU Replaced by the week number of the year (Sunday as the first day of the week, rules
64711 corresponding to %U) using the locale's alternative numeric symbols.

64712 %OV Replaced by the week number of the year (Monday as the first day of the week, rules
64713 corresponding to %V) using the locale's alternative numeric symbols.

64714 %Ow Replaced by the number of the weekday (Sunday=0) using the locale's alternative
64715 numeric symbols.

64716 %OW Replaced by the week number of the year (Monday as the first day of the week) using
64717 the locale's alternative numeric symbols.

64718 %Oy Replaced by the year (offset from %C) using the locale's alternative numeric symbols.

64719 %g, %G, and %V give values according to the ISO 8601:2004 standard week-based year. In this
64720 system, weeks begin on a Monday and week 1 of the year is the week that includes January 4th,
64721 which is also the week that includes the first Thursday of the year, and is also the first week that
64722 contains at least four days in the year. If the first Monday of January is the 2nd, 3rd, or 4th, the
64723 preceding days are part of the last week of the preceding year; thus, for Saturday 2nd January
64724 1999, %G is replaced by 1998 and %V is replaced by 53. If December 29th, 30th, or 31st is a
64725 Monday, it and any following days are part of week 1 of the following year. Thus, for Tuesday
64726 30th December 1997, %G is replaced by 1998 and %V is replaced by 01.

64727 If a conversion specifier is not one of the above, the behavior is undefined.

64728 CX The behavior is undefined if the *locale* argument to *strftime_l()* is the special locale object +
64729 LC_GLOBAL_LOCALE or is not a valid locale object handle.

64730 RETURN VALUE

64731 If the total number of resulting bytes including the terminating null byte is not more than
64732 *maxsize*, these functions shall return the number of bytes placed into the array pointed to by *s*,
64733 not including the terminating NUL character. Otherwise, 0 shall be returned and the contents of
64734 the array are unspecified.

64735 ERRORS

64736 No errors are defined.

64737 EXAMPLES

64738 Getting a Localized Date String

64739 The following example first sets the locale to the user's default. The locale information will be
64740 used in the *nl_langinfo()* and *strftime()* functions. The *nl_langinfo()* function returns the localized
64741 date string which specifies how the date is laid out. The *strftime()* function takes this
64742 information and, using the **tm** structure for values, places the date and time information into
64743 *datestring*.

```
64744 #include <time.h>
64745 #include <locale.h>
64746 #include <langinfo.h>
64747 ...
64748 struct tm *tm;
64749 char datestring[256];
64750 ...
```



```

64751     setlocale (LC_ALL, "");
64752     ...
64753     strptime (datestring, sizeof(datestring), nl_langinfo (D_T_FMT), tm);
64754     ...

```

APPLICATION USAGE

The range of values for %S is [00,60] rather than [00,59] to allow for the occasional leap second.

Some of the conversion specifications are duplicates of others. They are included for compatibility with *nl_cxtime()* and *nl_ascxtime()*, which were published in Issue 2.

The %C, %F, %G, and %Y format specifiers in *strptime()* always print full values, but the *strptime()* %C, %F, and %Y format specifiers only scan two digits (assumed to be the first two digits of a four-digit year) for %C and four digits (assumed to be the entire (four-digit) year) for %F and %Y. This mimics the behavior of *printf()* and *scanf()*; that is:

```
printf("%2d", x = 1000);
```

prints "1000", but:

```
scanf("%2d", &x);
```

when given "1000" as input will only store 10 in x). Applications using extended ranges of years must be sure that the number of digits specified for scanning years with *strptime()* matches the number of digits that will actually be present in the input stream. Historic implementations of the %Y conversion specification (with no flags and no minimum field width) produced different output formats. Some always produced at least four digits (with 0 fill for years from 0 through 999) while others only produced the number of digits present in the year (with no fill and no padding). These two forms can be produced with the '0' flag and a minimum field width options using the conversions specifications %04Y and %01Y, respectively.

In the past, the C and POSIX standards specified that %F produced an ISO 8601:2004 standard date format, but didn't specify which one. For years in the range [0001,9999], POSIX.1-2008 requires that the output produced match the ISO 8601:2004 standard complete representation extended format (YYYY-MM-DD) and for years outside of this range produce output that matches the ISO 8601:2004 standard expanded representation extended format (<+/-><Underline>Y</Underline>YYYY-MM-DD). To fully meet ISO 8601:2004 standard requirements, the producer and consumer must agree on a date format that has a specific number of bytes reserved to hold the characters used to represent the years that is sufficiently large to hold all values that will be shared. For example, the %+13F conversion specification will produce output matching the format "<+/->YYYYYY-MM-DD" (a leading '+' or '-' sign; a six-digit, 0-filled year; a '-' ; a two-digit, leading 0-filled month; another '-' ; and the two-digit, leading 0-filled day within the month).

Note that if the year being printed is greater than 9999, the resulting string from the unadorned %F conversion specifications will not conform to the ISO 8601:2004 standard extended format, complete representation for a date and will instead be an extended format, expanded representation (presumably without the required agreement between the date's producer and consumer).

In the C locale, the E and O modifiers are ignored and the replacement strings for the following specifiers are:

- %a The first three characters of %A.
- %A One of Sunday, Monday, ..., Saturday.

64795	%b	The first three characters of %B.
64796	%B	One of January, February, ..., December.
64797	%C	Equivalent to %a %b %e %T %Y.
64798	%P	One of AM or PM.
64799	%r	Equivalent to %I:%M:%S %p.
64800	%x	Equivalent to %m/%d/%y.
64801	%X	Equivalent to %T.
64802	%Z	Implementation-defined.

64803 RATIONALE

64804 The %Y conversion specification to *strptime()* was frequently assumed to be a four-digit year, but
 64805 the ISO C standard does not specify that %Y is restricted to any subset of allowed values from the
 64806 *tm_year* field. Similarly, the %C conversion specification was assumed to be a two-digit field and
 64807 the first part of the output from the %F conversion specification was assumed to be a four-digit
 64808 field. With *tm_year* being a signed 32 or more-bit **int** and with many current implementations
 64809 supporting 64-bit **time_t** types in one or more programming environments, these assumptions
 64810 are clearly wrong.

64811 POSIX.1-2008 now allows the format specifications %0xC, %0xF, %0xG, and %0xY (where 'x' is
 64812 a string of decimal digits used to specify printing and scanning of a string of *x* decimal digits)
 64813 with leading zero fill characters. Allowing applications to set the field width enables them to
 64814 agree on the number of digits to be printed and scanned in the ISO 8601:2004 standard
 64815 expanded representation of a year (for %F, %G, and %Y) or all but the last two digits of the year
 64816 (for %C). This is based on a feature in some versions of GNU **libc**'s *strptime()*. The GNU version
 64817 allows specifying space, zero, or no-fill characters in *strptime()* format strings, but does not allow
 64818 any flags to be specified in *strptime()* format strings. These implementations also allow these
 64819 flags to be specified for any numeric field. POSIX.1-2008 only requires the zero fill flag ('0') and
 64820 only requires that it be recognized when processing %C, %F, %G, and %Y specifications when a
 64821 minimum field width is also specified. The '0' flag is the only flag needed to produce and scan
 64822 the ISO 8601:2004 standard year fields using the extended format forms. POSIX.1-2008 also
 64823 allows applications to specify the same flag and field width specifiers to be used in both
 64824 *strptime()* and *strptime()* format strings for symmetry. Systems may provide other flag characters
 64825 and may accept flags in conjunction with conversion specifiers other than %C, %F, %G, and %Y;
 64826 but portable applications cannot depend on such extensions.

64827 POSIX.1-2008 now also allows the format specifications %+xC, %+xF, %+xG, and %+xY (where
 64828 'x' is a string of decimal digits used to specify printing and scanning of a string of 'x' decimal
 64829 digits) with leading zero fill characters and a leading '+' sign character if the year being
 64830 converted is more than four digits or a minimum field width is specified that allows room for
 64831 more than four digits for the year. This allows date providers and consumers to agree on a
 64832 specific number of digits to represent a year as required by the ISO 8601:2004 standard
 64833 expanded representation formats. The expanded representation formats all require the year to
 64834 begin with a leading '+' or '-' sign. (All of these specifiers can also provide a leading '-'
 64835 sign for negative years. Since negative years and the year 0 don't fit well with the Gregorian or
 64836 Julian calendars, the normal ranges of dates start with year 1. The ISO C standard allows *tm_year*
 64837 to assume values corresponding to years before year 1, but the use of such years provided
 64838 unspecified results.)

64839 Some earlier version of this standard specified that applications wanting to use *strptime()* to scan
 64840 dates and times printed by *strptime()* should provide non-digit characters between fields to

separate years from months and days. It also supported %F to print and scan the ISO 8601:2004 standard extended format, complete representation date for years 1 through 9999 (i.e., YYYY-MM-DD). However, many applications were written to print (using *strptime()*) and scan (using *strptime()*) dates written using the basic format complete representation (four-digit years) and truncated representation (two-digit years) specified by the ISO 8601:2004 standard representation of dates and times which do not have any separation characters between fields. The ISO 8601:2004 standard also specifies basic format expanded representation where the creator and consumer of these fields agree beforehand to represent years as leading zero-filled strings of an agreed length of more than four digits to represent a year (again with no separation characters when year, month, and day are all displayed). Applications producing and consuming expanded representations are encouraged to use the '+' flag and an appropriate maximum field width to scan the year including the leading sign. Note that even without the '+' flag, years less than zero may be represented with a leading minus-sign for %F, %G, and %Y conversion specifications. Using negative years results in unspecified behavior.

If a format specification %x F with the field width *x* greater than 11 is specified and the width is large enough to display the full year, the output string produced will match the ISO 8601:2004 standard subclause 4.1.2.4 expanded representation, extended format date representation for a specific day. (For years in the range [1,99 999], %+12F is sufficient for an agreed five-digit year with a leading sign using the ISO 8601:2004 standard expanded representation, extended format for a specific day "<+/->YYYY-MM-DD".) Note also that years less than 0 may produce a leading minus-sign ('-') when using %Y or %C whether or not the '0' or '+' flags are used.

The difference between the '0' flag and the '+' flag is whether the leading '+' character will be provided for years >9999 as required for the ISO 8601:2004 standard extended representation format containing a year. For example:

Year	Conversion Specification	<i>strptime()</i> Output	<i>strptime()</i> Scan Back
1970	%Y	1970	1970
1970	%+4Y	1970	1970
27	%Y	27 or 0027	27
270	%Y	270 or 0270	270
270	%+4Y	0270	270
17	%C%y	0017	17
270	%C%y	0270	270
12345	%Y	12345	1234*
12345	%+4Y	+12345	123*
12345	%05Y	12345	12345
270	%+5Y or %+3C%y	+0270	270
12345	%+5Y or %+3C%y	+12345	1234*
12345	%06Y or %04C%y	012345	12345
12345	%+6Y or %+4C%y	+12345	12345
123456	%08Y or %06C%y	00123456	123456
123456	%+8Y or %+6C%y	+0123456	123456

In the cases above marked with a * in the *strptime()* scan back field, the implied or specified number of characters scanned by *strptime()* was less than the number of characters output by *strptime()* using the same format; so the remaining digits of the year were dropped when the output date produced by *strptime()* was scanned back in by *strptime()*.

FUTURE DIRECTIONS

None.

SEE ALSO

asctime(), *clock()*, *ctime()*, *difftime()*, *getdate()*, *gmtime()*, *localtime()*, *mktime()*, *strptime()*, *time()*, *tzset()*, *uselocale()*, *utime()*

XBD Section 7.3.5 (on page 159), **<time.h>**

CHANGE HISTORY

First released in Issue 3.

Issue 5

The description of %OV is changed to be consistent with %V and defines Monday as the first day of the week.

The description of %Oy is clarified.

Issue 6

Extensions beyond the ISO C standard are marked.

The Open Group Corrigendum U033/8 is applied. The %V conversion specifier is changed from “Otherwise, it is week 53 of the previous year, and the next week is week 1” to “Otherwise, it is the last week of the previous year, and the next week is week 1”.

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The %C, %D, %e, %h, %n, %r, %R, %t, and %T conversion specifiers are added.
- The modified conversion specifiers are added for consistency with the ISO POSIX-2 standard *date* utility.

The following changes are made for alignment with the ISO/IEC 9899:1999 standard:

- The *strptime()* prototype is updated.
- The DESCRIPTION is extensively revised.
- The %z conversion specifier is added.

An example is added.

IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/60 is applied.

Issue 7

Austin Group Interpretation 1003.1-2001 #163 is applied.

The *strptime_l()* function is added from The Open Group Technical Standard, 2006, Extended API Set Part 4.

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0605 [283], XSH/TC1-2008/0606 [283], + XSH/TC1-2008/0607 [193], and XSH/TC1-2008/0608 [193] are applied.

64921 **NAME**

64922 strlen, strlen — get length of fixed size string

64923 **SYNOPSIS**

64924 #include <string.h>

64925 size_t strlen(const char *s);

64926 CX size_t strnlen(const char *s, size_t maxlen);

64927 **DESCRIPTION**

64928 CX For *strlen()*: The functionality described on this reference page is aligned with the ISO C
 64929 standard. Any conflict between the requirements described here and the ISO C standard is
 64930 unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

64931 The *strlen()* function shall compute the number of bytes in the string to which *s* points, not
 64932 including the terminating NUL character.

64933 CX The *strnlen()* function shall compute the smaller of the number of bytes in the array to which *s*
 64934 points, not including the terminating NUL character, or the value of the *maxlen* argument. The
 64935 *strnlen()* function shall never examine more than *maxlen* bytes of the array pointed to by *s*.

64936 **RETURN VALUE**

64937 The *strlen()* function shall return the length of *s*; no return value shall be reserved to indicate an
 64938 error.

64939 CX The *strnlen()* function shall return an integer containing the smaller of either the length of the
 64940 string pointed to by *s* or *maxlen*.

64941 **ERRORS**

64942 No errors are defined.

64943 **EXAMPLES**64944 **Getting String Lengths**

64945 The following example sets the maximum length of *key* and *data* by using *strlen()* to get the
 64946 lengths of those strings.

```

64947     #include <string.h>
64948     ...
64949     struct element {
64950         char *key;
64951         char *data;
64952     };
64953     ...
64954     char *key, *data;
64955     int len;
64956
64957     *keylength = *datalength = 0;
64958     ...
64959     if ((len = strlen(key)) > *keylength)
64960         *keylength = len;
64961     if ((len = strlen(data)) > *datalength)
64962         *datalength = len;
64963     ...

```

64963 APPLICATION USAGE

64964 None.

64965 RATIONALE

64966 None.

64967 FUTURE DIRECTIONS

64968 None.

64969 SEE ALSO

64970 [*wcslen\(\)*](#)

64971 XBD [**<string.h>**](#)

64972 CHANGE HISTORY

64973 First released in Issue 1. Derived from Issue 1 of the SVID.

64974 Issue 5

64975 The RETURN VALUE section is updated to indicate that *strlen()* returns the length of *s*, and not
64976 *s* itself as was previously stated.

64977 Issue 7

64978 The *strnlen()* function is added from The Open Group Technical Standard, 2006, Extended API
64979 Set Part 1.

64980 **NAME**

64981 strncasecmp, strncasecmp_l — case-insensitive string comparisons

64982 **SYNOPSIS**

64983 #include <strings.h>

64984 int strncasecmp(const char *s1, const char *s2, size_t n);

64985 int strncasecmp_l(const char *s1, const char *s2,

64986 size_t n, locale_t locale);

64987 **DESCRIPTION**64988 Refer to *strcasecmp()*.

NAME

strncat — concatenate a string with part of another

SYNOPSIS

```
#include <string.h>
```

```
char *strncat(char *restrict s1, const char *restrict s2, size_t n);
```

DESCRIPTION

CX The functionality described on this reference page is aligned with the ISO C standard. Any conflict between the requirements described here and the ISO C standard is unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

The *strncat()* function shall append not more than *n* bytes (a NUL character and bytes that follow it are not appended) from the array pointed to by *s2* to the end of the string pointed to by *s1*. The initial byte of *s2* overwrites the NUL character at the end of *s1*. A terminating NUL character is always appended to the result. If copying takes place between objects that overlap, the behavior is undefined.

RETURN VALUE

The *strncat()* function shall return *s1*; no return value shall be reserved to indicate an error.

ERRORS

No errors are defined.

EXAMPLES

None.

APPLICATION USAGE

None.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

[strcat\(\)](#)

XBD [<string.h>](#)

CHANGE HISTORY

First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6

The *strncat()* prototype is updated for alignment with the ISO/IEC 9899:1999 standard.

65022 **NAME**

65023 strncmp — compare part of two strings

65024 **SYNOPSIS**

65025 #include <string.h>

65026 int strncmp(const char *s1, const char *s2, size_t n);

65027 **DESCRIPTION**

65028 CX The functionality described on this reference page is aligned with the ISO C standard. Any
65029 conflict between the requirements described here and the ISO C standard is unintentional. This
65030 volume of POSIX.1-2008 defers to the ISO C standard.

65031 The *strncmp()* function shall compare not more than *n* bytes (bytes that follow a NUL character
65032 are not compared) from the array pointed to by *s1* to the array pointed to by *s2*.

65033 The sign of a non-zero return value is determined by the sign of the difference between the
65034 values of the first pair of bytes (both interpreted as type **unsigned char**) that differ in the strings
65035 being compared.

65036 **RETURN VALUE**

65037 Upon successful completion, *strncmp()* shall return an integer greater than, equal to, or less than
65038 0, if the possibly null-terminated array pointed to by *s1* is greater than, equal to, or less than the
65039 possibly null-terminated array pointed to by *s2* respectively.

65040 **ERRORS**

65041 No errors are defined.

65042 **EXAMPLES**

65043 None.

65044 **APPLICATION USAGE**

65045 None.

65046 **RATIONALE**

65047 None.

65048 **FUTURE DIRECTIONS**

65049 None.

65050 **SEE ALSO**65051 *strcmp()*

65052 XBD <string.h>

65053 **CHANGE HISTORY**

65054 First released in Issue 1. Derived from Issue 1 of the SVID.

65055 **Issue 6**

65056 Extensions beyond the ISO C standard are marked.

NAME

65057
65058 `stpncpy`, `strncpy` — copy fixed length string, returning a pointer to the array end

SYNOPSIS

65059
65060 `#include <string.h>`

65061 CX `char *stpncpy(char *restrict s1, const char *restrict s2, size_t n);`
65062 `char *strncpy(char *restrict s1, const char *restrict s2, size_t n);`

DESCRIPTION

65063
65064 CX For `strncpy()`: The functionality described on this reference page is aligned with the ISO C
65065 standard. Any conflict between the requirements described here and the ISO C standard is
65066 unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

65067 CX The `stpncpy()` and `strncpy()` functions shall copy not more than *n* bytes (bytes that follow a NUL
65068 character are not copied) from the array pointed to by *s2* to the array pointed to by *s1*.

65069 If the array pointed to by *s2* is a string that is shorter than *n* bytes, NUL characters shall be
65070 appended to the copy in the array pointed to by *s1*, until *n* bytes in all are written.

65071 If copying takes place between objects that overlap, the behavior is undefined.

RETURN VALUE

65072
65073 CX If a NUL character is written to the destination, the `stpncpy()` function shall return the address of
65074 the first such NUL character. Otherwise, it shall return `&s1[n]`.

65075 The `strncpy()` function shall return *s1*.

65076 No return values are reserved to indicate an error.

ERRORS

65077
65078 No errors are defined.

EXAMPLES

65079
65080 None.

APPLICATION USAGE

65081
65082 Applications must provide the space in *s1* for the *n* bytes to be transferred, as well as ensure that
65083 the *s2* and *s1* arrays do not overlap.

65084 Character movement is performed differently in different implementations. Thus, overlapping
65085 moves may yield surprises.

65086 If there is no NUL character byte in the first *n* bytes of the array pointed to by *s2*, the result is not
65087 null-terminated.

RATIONALE

65088
65089 None.

FUTURE DIRECTIONS

65090
65091 None.

SEE ALSO

65092
65093 [*strcpy\(\)*](#), [*wcsncpy\(\)*](#)

65094 XBD [*<string.h>*](#)

CHANGE HISTORY

65095
65096 First released in Issue 1. Derived from Issue 1 of the SVID.

65097 **Issue 6**

65098 The *strncpy()* prototype is updated for alignment with the ISO/IEC 9899:1999 standard.

65099 **Issue 7**

65100 The *stpncpy()* function is added from The Open Group Technical Standard, 2006, Extended API
65101 Set Part 1.

65102 NAME

65103 strndup — duplicate a specific number of bytes from a string

65104 SYNOPSIS

```
65105 CX #include <string.h>  
65106 char *strndup(const char *s, size_t size);
```

65107 DESCRIPTION

65108 Refer to *strdup()*.

65109 **NAME**

65110 strnlen — get length of fixed size string

65111 **SYNOPSIS**65112 CX `#include <string.h>`65113 `size_t strnlen(const char *s, size_t maxlen);`65114 **DESCRIPTION**65115 Refer to *strlen()*.

65116 **NAME**

65117 strpbrk — scan a string for a byte

65118 **SYNOPSIS**

65119 #include <string.h>

65120 char *strpbrk(const char *s1, const char *s2);

65121 **DESCRIPTION**

65122 CX The functionality described on this reference page is aligned with the ISO C standard. Any
65123 conflict between the requirements described here and the ISO C standard is unintentional. This
65124 volume of POSIX.1-2008 defers to the ISO C standard.

65125 The *strpbrk()* function shall locate the first occurrence in the string pointed to by *s1* of any byte
65126 from the string pointed to by *s2*.

65127 **RETURN VALUE**

65128 Upon successful completion, *strpbrk()* shall return a pointer to the byte or a null pointer if no
65129 byte from *s2* occurs in *s1*.

65130 **ERRORS**

65131 No errors are defined.

65132 **EXAMPLES**

65133 None.

65134 **APPLICATION USAGE**

65135 None.

65136 **RATIONALE**

65137 None.

65138 **FUTURE DIRECTIONS**

65139 None.

65140 **SEE ALSO**65141 *strchr()*, *strrchr()*

65142 XBD <string.h>

65143 **CHANGE HISTORY**

65144 First released in Issue 1. Derived from Issue 1 of the SVID.

NAME

strptime — date and time conversion

SYNOPSIS

```
XSI    #include <time.h>

char *strptime(const char *restrict buf, const char *restrict format,
               struct tm *restrict tm);
```

DESCRIPTION

The *strptime()* function shall convert the character string pointed to by *buf* to values which are stored in the **tm** structure pointed to by *tm*, using the format specified by *format*.

The format is composed of zero or more directives. Each directive is composed of one of the following: one or more white-space characters (as specified by *isspace()*); an ordinary character (neither `' % '` nor a white-space character); or a conversion specification.

Each conversion specification is introduced by the `' % '` character after which the following appear in sequence:

- An optional flag, the zero character (`' 0 '`) or the `<plus-sign>` character (`' + '`), which is ignored.
- An optional field width. If a field width is specified, it shall be interpreted as a string of decimal digits that will determine the maximum number of bytes converted for the conversion rather than the number of bytes specified below in the description of the conversion specifiers.
- An optional `E` or `O` modifier.
- A terminating conversion specifier character that indicates the type of conversion to be applied.

The conversions are determined using the *LC_TIME* category of the current locale. The application shall ensure that there is white-space or other non-alphanumeric characters between any two conversion specifications unless all of the adjacent conversion specifications convert a known, fixed number of characters. In the following list, the maximum number of characters scanned (excluding the one matching the next directive) is as follows:

- If a maximum field width is specified, then that number
- Otherwise, the pattern `" { x } "` indicates that the maximum is *x*
- Otherwise, the pattern `" [ x, y ] "` indicates that the value shall fall within the range given (both bounds being inclusive), and the maximum number of characters scanned shall be the maximum required to represent any value in the range without leading zeros and without a leading `<plus-sign>`

The following conversion specifiers are supported.

The results are unspecified if a modifier is specified with a flag or with a minimum field width, or if a field width is specified for any conversion specifier other than `C`, `F`, or `Y`.

a	The day of the week, using the locale's weekday names; either the abbreviated or full name may be specified.
A	Equivalent to <code>%a</code> .

65185	b	The month, using the locale's month names; either the abbreviated or full name may be specified.
65186		
65187	B	Equivalent to %b.
65188	c	Replaced by the locale's appropriate date and time representation.
65189	C	All but the last two digits of the year {2}; leading zeros shall be permitted but shall not be required. A leading '+' or '-' character shall be permitted before any leading zeros but shall not be required.
65190		
65191		
65192	d	The day of the month [01,31]; leading zeros shall be permitted but shall not be required.
65193	D	The date as %m/%d/%y.
65194	e	Equivalent to %d.
65195	h	Equivalent to %b.
65196	H	The hour (24-hour clock) [00,23]; leading zeros shall be permitted but shall not be required.
65197		
65198	I	The hour (12-hour clock) [01,12]; leading zeros shall be permitted but shall not be required.
65199		
65200	j	The day number of the year [001,366]; leading zeros shall be permitted but shall not be required.
65201		
65202	m	The month number [01,12]; leading zeros shall be permitted but shall not be required.
65203	M	The minute [00,59]; leading zeros shall be permitted but shall not be required.
65204	n	Any white space.
65205	p	The locale's equivalent of a.m. or p.m.
65206	r	12-hour clock time using the AM/PM notation if t_fmt_ampm is not an empty string in the <i>LC_TIME</i> portion of the current locale; in the POSIX locale, this shall be equivalent to %I:%M:%S %p.
65207		
65208		
65209	R	The time as %H:%M.
65210	S	The seconds [00,60]; leading zeros shall be permitted but shall not be required.
65211	t	Any white space.
65212	T	The time as %H:%M:%S.
65213	U	The week number of the year (Sunday as the first day of the week) as a decimal number [00,53]; leading zeros shall be permitted but shall not be required.
65214		
65215	w	The weekday as a decimal number [0,6], with 0 representing Sunday.
65216	W	The week number of the year (Monday as the first day of the week) as a decimal number [00,53]; leading zeros shall be permitted but shall not be required.
65217		
65218	x	The date, using the locale's date format.
65219	X	The time, using the locale's time format.
65220	y	The last two digits of the year. When <i>format</i> contains neither a C conversion specifier nor a Y conversion specifier, values in the range [69,99] shall refer to years 1969 to 1999 inclusive and values in the range [00,68] shall refer to years 2000 to 2068 inclusive; leading zeros shall be permitted but shall not be required. A leading '+' or '-'
65221		
65222		
65223		

65224		character shall be permitted before any leading zeros but shall not be required.
65225	Note:	It is expected that in a future version of this standard the default century inferred
65226		from a 2-digit year will change. (This would apply to all commands accepting a
65227		2-digit year as input.)
65228	Y	The full year {4}; leading zeros shall be permitted but shall not be required. A leading
65229		'+' or '-' character shall be permitted before any leading zeros but shall not be
65230		required.
65231	%	Replaced by %.

65232 Modified Conversion Specifiers

65233 Some conversion specifiers can be modified by the E and O modifier characters to indicate that
 65234 an alternative format or specification should be used rather than the one normally used by the
 65235 unmodified conversion specifier. If the alternative format or specification does not exist in the
 65236 current locale, the behavior shall be as if the unmodified conversion specification were used.

65237	%Ec	The locale's alternative appropriate date and time representation.
65238	%EC	The name of the base year (period) in the locale's alternative representation.
65239	%Ex	The locale's alternative date representation.
65240	%EX	The locale's alternative time representation.
65241	%Ey	The offset from %EC (year only) in the locale's alternative representation.
65242	%EY	The full alternative year representation.
65243	%Od	The day of the month using the locale's alternative numeric symbols; leading zeros
65244		shall be permitted but shall not be required.
65245	%Oe	Equivalent to %Od.
65246	%OH	The hour (24-hour clock) using the locale's alternative numeric symbols.
65247	%OI	The hour (12-hour clock) using the locale's alternative numeric symbols.
65248	%Om	The month using the locale's alternative numeric symbols.
65249	%OM	The minutes using the locale's alternative numeric symbols.
65250	%OS	The seconds using the locale's alternative numeric symbols.
65251	%OU	The week number of the year (Sunday as the first day of the week) using the locale's
65252		alternative numeric symbols.
65253	%Ow	The number of the weekday (Sunday=0) using the locale's alternative numeric
65254		symbols.
65255	%OW	The week number of the year (Monday as the first day of the week) using the locale's
65256		alternative numeric symbols.
65257	%Oy	The year (offset from %C) using the locale's alternative numeric symbols.
65258		A conversion specification composed of white-space characters is executed by scanning input up
65259		to the first character that is not white-space (which remains unscanned), or until no more
65260		characters can be scanned.
65261		A conversion specification that is an ordinary character is executed by scanning the next
65262		character from the buffer. If the character scanned from the buffer differs from the one
65263		comprising the directive, the directive fails, and the differing and subsequent characters remain

65264 unscanned.

65265 A series of conversion specifications composed of %n, %t, white-space characters, or any
65266 combination is executed by scanning up to the first character that is not white space (which
65267 remains unscanned), or until no more characters can be scanned.

65268 Any other conversion specification is executed by scanning characters until a character matching
65269 the next directive is scanned, or until no more characters can be scanned. These characters,
65270 except the one matching the next directive, are then compared to the locale values associated
65271 with the conversion specifier. If a match is found, values for the appropriate **tm** structure
65272 members are set to values corresponding to the locale information. Case is ignored when
65273 matching items in *buf* such as month or weekday names. If no match is found, *strptime()* fails
65274 and no more characters are scanned.

65275 RETURN VALUE

65276 Upon successful completion, *strptime()* shall return a pointer to the character following the last
65277 character parsed. Otherwise, a null pointer shall be returned.

65278 ERRORS

65279 No errors are defined.

65280 EXAMPLES

65281 Convert a Data-Plus-Time String to Broken-Down Time and Then into Seconds

65282 The following example demonstrates the use of *strptime()* to convert a string into broken-down
65283 time. The broken-down time is then converted into seconds since the Epoch using *mktime()*.

```
65284 #include <time.h>
65285 ...
65286 struct tm tm;
65287 time_t t;
65288 if (strptime("6 Dec 2001 12:33:45", "%d %b %Y %H:%M:%S", &tm) == NULL)
65289     /* Handle error */;
65290 printf("year: %d; month: %d; day: %d;\n",
65291        tm.tm_year, tm.tm_mon, tm.tm_mday);
65292 printf("hour: %d; minute: %d; second: %d\n",
65293        tm.tm_hour, tm.tm_min, tm.tm_sec);
65294 printf("week day: %d; year day: %d\n", tm.tm_wday, tm.tm_yday);
65295 tm.tm_isdst = -1;          /* Not set by strptime(); tells mktime()
65296                            to determine whether daylight saving time
65297                            is in effect */
65298 t = mktime(&tm);
65299 if (t == -1)
65300     /* Handle error */;
65301 printf("seconds since the Epoch: %ld\n", (long) t);"
```

65302 APPLICATION USAGE

65303 Several “equivalent to” formats and the special processing of white-space characters are
65304 provided in order to ease the use of identical *format* strings for *strptime()* and *strptime()*.

65305 It should be noted that dates constructed by the *strptime()* function with the %Y or %C%y
65306 conversion specifiers may have values larger than 9999. If the *strptime()* function is used to read
65307 such values using %C%y or %Y, the year values will be truncated to four digits. Applications

should use `%+w%y` or `%+xY` with *w* and *x* set large enough to contain the full value of any years that will be printed or scanned.

See also the APPLICATION USAGE section in *strptime()*.

It is unspecified whether multiple calls to *strptime()* using the same **tm** structure will update the current contents of the structure or overwrite all contents of the structure. Conforming applications should make a single call to *strptime()* with a format and all data needed to completely specify the date and time being converted.

RATIONALE

See the RATIONALE section for *strptime()*.

FUTURE DIRECTIONS

None.

SEE ALSO

fprintf(), *fscanf()*, *strptime()*, *time()*

XBD **<time.h>**

CHANGE HISTORY

First released in Issue 4.

Issue 5

Moved from ENHANCED I18N to BASE.

The [ENOSYS] error is removed.

The exact meaning of the `%y` and `%0y` specifiers is clarified in the DESCRIPTION.

Issue 6

The Open Group Corrigendum U033/5 is applied. The `%r` specifier description is reworded.

The normative text is updated to avoid use of the term “must” for application requirements.

The **restrict** keyword is added to the *strptime()* prototype for alignment with the ISO/IEC 9899:1999 standard.

The Open Group Corrigendum U047/2 is applied.

The DESCRIPTION is updated to use the terms “conversion specifier” and “conversion specification” for consistency with *strptime()*.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/133 is applied, adding the example to the EXAMPLES section.

Issue 7

Austin Group Interpretation 1003.1-2001 #041 is applied, updating the DESCRIPTION and APPLICATION USAGE sections.

Austin Group Interpretation 1003.1-2001 #163 is applied.

SD5-XSH-ERN-67 is applied, correcting the APPLICATION USAGE to remove the impression that `%Y` is 4-digit years.

NAME

strrchr — string scanning operation

SYNOPSIS

```
#include <string.h>
```

```
char *strrchr(const char *s, int c);
```

DESCRIPTION

CX The functionality described on this reference page is aligned with the ISO C standard. Any conflict between the requirements described here and the ISO C standard is unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

The *strrchr()* function shall locate the last occurrence of *c* (converted to a **char**) in the string pointed to by *s*. The terminating NUL character is considered to be part of the string.

RETURN VALUE

Upon successful completion, *strrchr()* shall return a pointer to the byte or a null pointer if *c* does not occur in the string.

ERRORS

No errors are defined.

EXAMPLES**Finding the Base Name of a File**

The following example uses *strrchr()* to get a pointer to the base name of a file. The *strrchr()* function searches backwards through the name of the file to find the last '/' character in *name*. This pointer (plus one) will point to the base name of the file.

```
#include <string.h>
...
const char *name;
char *basename;
...
basename = strrchr(name, '/') + 1;
...
```

APPLICATION USAGE

None.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

strchr()

XBD *<string.h>*

CHANGE HISTORY

First released in Issue 1. Derived from Issue 1 of the SVID.

NAME

strsignal — get name of signal

SYNOPSIS

```
CX      #include <string.h>
char *strsignal(int signum);
```

DESCRIPTION

The *strsignal()* function shall map the signal number in *signum* to an implementation-defined string and shall return a pointer to it. It shall use the same set of messages as the *psignal()* function.

The application shall not modify the string returned. The returned pointer might be invalidated or the string content might be overwritten by a subsequent call to *strsignal()* or *setlocale()*.

The contents of the message strings returned by *strsignal()* should be determined by the setting of the *LC_MESSAGES* category in the current locale.

The implementation shall behave as if no function defined in this standard calls *strsignal()*.

Since no return value is reserved to indicate an error, an application wishing to check for error situations should set *errno* to 0, then call *strsignal()*, then check *errno*.

The *strsignal()* function need not be thread-safe.

RETURN VALUE

Upon successful completion, *strsignal()* shall return a pointer to a string. Otherwise, if *signum* is not a valid signal number, the return value is unspecified.

ERRORS

No errors are defined.

EXAMPLES

None.

APPLICATION USAGE

None.

RATIONALE

If *signum* is not a valid signal number, some implementations return NULL, while for others the *strsignal()* function returns a pointer to a string containing an unspecified message denoting an unknown signal. POSIX.1-2008 leaves this return value unspecified.

FUTURE DIRECTIONS

None.

SEE ALSO

psiginfo(), *setlocale()*

XBD *<string.h>*

CHANGE HISTORY

First released in Issue 7.

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0609 [75] is applied.

+

65421 **NAME**

65422 strspn — get length of a substring

65423 **SYNOPSIS**

65424 #include <string.h>

65425 size_t strspn(const char *s1, const char *s2);

65426 **DESCRIPTION**

65427 CX The functionality described on this reference page is aligned with the ISO C standard. Any
65428 conflict between the requirements described here and the ISO C standard is unintentional. This
65429 volume of POSIX.1-2008 defers to the ISO C standard.

65430 The *strspn()* function shall compute the length (in bytes) of the maximum initial segment of the
65431 string pointed to by *s1* which consists entirely of bytes from the string pointed to by *s2*.

65432 **RETURN VALUE**

65433 The *strspn()* function shall return the computed length; no return value is reserved to indicate an
65434 error.

65435 **ERRORS**

65436 No errors are defined.

65437 **EXAMPLES**

65438 None.

65439 **APPLICATION USAGE**

65440 None.

65441 **RATIONALE**

65442 None.

65443 **FUTURE DIRECTIONS**

65444 None.

65445 **SEE ALSO**65446 [strcspn\(\)](#)65447 XBD [<string.h>](#)65448 **CHANGE HISTORY**

65449 First released in Issue 1. Derived from Issue 1 of the SVID.

65450 **Issue 5**

65451 The RETURN VALUE section is updated to indicate that *strspn()* returns the length of *s*, and not
65452 *s* itself as was previously stated.

65453 **Issue 7**

65454 SD5-XSH-ERN-182 is applied.

65455 **NAME**

65456 strstr — find a substring

65457 **SYNOPSIS**

65458 #include <string.h>

65459 char *strstr(const char *s1, const char *s2);

65460 **DESCRIPTION**

65461 CX The functionality described on this reference page is aligned with the ISO C standard. Any
65462 conflict between the requirements described here and the ISO C standard is unintentional. This
65463 volume of POSIX.1-2008 defers to the ISO C standard.

65464 The *strstr()* function shall locate the first occurrence in the string pointed to by *s1* of the
65465 sequence of bytes (excluding the terminating NUL character) in the string pointed to by *s2*.

65466 **RETURN VALUE**

65467 Upon successful completion, *strstr()* shall return a pointer to the located string or a null pointer
65468 if the string is not found.

65469 If *s2* points to a string with zero length, the function shall return *s1*.

65470 **ERRORS**

65471 No errors are defined.

65472 **EXAMPLES**

65473 None.

65474 **APPLICATION USAGE**

65475 None.

65476 **RATIONALE**

65477 None.

65478 **FUTURE DIRECTIONS**

65479 None.

65480 **SEE ALSO**65481 *strchr()*

65482 XBD <string.h>

65483 **CHANGE HISTORY**

65484 First released in Issue 3. Included for alignment with the ANSI C standard.

NAME

strtod, strtodf, strtold — convert a string to a double-precision number

SYNOPSIS

```
#include <stdlib.h>

double strtod(const char *restrict nptr, char **restrict endptr);
float strtodf(const char *restrict nptr, char **restrict endptr);
long double strtold(const char *restrict nptr, char **restrict endptr);
```

DESCRIPTION

The functionality described on this reference page is aligned with the ISO C standard. Any conflict between the requirements described here and the ISO C standard is unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

These functions shall convert the initial portion of the string pointed to by *nptr* to **double**, **float**, and **long double** representation, respectively. First, they decompose the input string into three parts:

1. An initial, possibly empty, sequence of white-space characters (as specified by *isspace()*)
2. A subject sequence interpreted as a floating-point constant or representing infinity or NaN
3. A final string of one or more unrecognized characters, including the terminating NUL character of the input string

Then they shall attempt to convert the subject sequence to a floating-point number, and return the result.

The expected form of the subject sequence is an optional '+' or '-' sign, then one of the following:

- A non-empty sequence of decimal digits optionally containing a radix character; then an optional exponent part consisting of the character 'e' or the character 'E', optionally followed by a '+' or '-' character, and then followed by one or more decimal digits
- A 0x or 0X, then a non-empty sequence of hexadecimal digits optionally containing a radix character; then an optional binary exponent part consisting of the character 'p' or the character 'P', optionally followed by a '+' or '-' character, and then followed by one or more decimal digits
- One of INF or INFINITY, ignoring case
- One of NAN or NAN(*n-char-sequence_{opt}*), ignoring case in the NAN part, where:

```
n-char-sequence:
    digit
    nondigit
    n-char-sequence digit
    n-char-sequence nondigit
```

The subject sequence is defined as the longest initial subsequence of the input string, starting with the first non-white-space character, that is of the expected form. The subject sequence contains no characters if the input string is not of the expected form.

If the subject sequence has the expected form for a floating-point number, the sequence of characters starting with the first digit or the decimal-point character (whichever occurs first) shall be interpreted as a floating constant of the C language, except that the radix character shall be used in place of a period, and that if neither an exponent part nor a radix character appears in

a decimal floating-point number, or if a binary exponent part does not appear in a hexadecimal floating-point number, an exponent part of the appropriate type with value zero is assumed to follow the last digit in the string. If the subject sequence begins with a minus-sign, the sequence shall be interpreted as negated. A character sequence INF or INFINITY shall be interpreted as an infinity, if representable in the return type, else as if it were a floating constant that is too large for the range of the return type. A character sequence NAN or NAN(*n-char-sequence_{opt}*) shall be interpreted as a quiet NaN, if supported in the return type, else as if it were a subject sequence part that does not have the expected form; the meaning of the *n-char* sequences is implementation-defined. A pointer to the final string is stored in the object pointed to by *endptr*, provided that *endptr* is not a null pointer.

If the subject sequence has the hexadecimal form and FLT_RADIX is a power of 2, the value resulting from the conversion is correctly rounded.

CX The radix character is defined in the current locale (category *LC_NUMERIC*). In the POSIX locale, or in a locale where the radix character is not defined, the radix character shall default to a <period> (' . ').

CX In other than the C or POSIX locales, other implementation-defined subject sequences may be accepted.

If the subject sequence is empty or does not have the expected form, no conversion shall be performed; the value of *nptr* is stored in the object pointed to by *endptr*, provided that *endptr* is not a null pointer.

These functions shall not change the setting of *errno* if successful.

Since 0 is returned on error and is also a valid return on success, an application wishing to check for error situations should set *errno* to 0, then call *strtod()*, *strtof()*, or *strtold()*, then check *errno*.

RETURN VALUE

Upon successful completion, these functions shall return the converted value. If no conversion could be performed, 0 shall be returned, and *errno* may be set to [EINVAL].

If the correct value is outside the range of representable values, $\pm$ HUGE_VAL, $\pm$ HUGE_VALF, or $\pm$ HUGE_VALL shall be returned (according to the sign of the value), and *errno* shall be set to [ERANGE].

If the correct value would cause an underflow, a value whose magnitude is no greater than the smallest normalized positive number in the return type shall be returned and *errno* set to [ERANGE].

ERRORS

These functions shall fail if:

CX [ERANGE] The value to be returned would cause overflow or underflow.

These functions may fail if:

CX [EINVAL] No conversion could be performed.

EXAMPLES

None.

APPLICATION USAGE

If the subject sequence has the hexadecimal form and FLT_RADIX is not a power of 2, and the result is not exactly representable, the result should be one of the two numbers in the appropriate internal format that are adjacent to the hexadecimal floating source value, with the extra stipulation that the error should have a correct sign for the current rounding direction.

If the subject sequence has the decimal form and at most DECIMAL_DIG (defined in `<float.h>`) significant digits, the result should be correctly rounded. If the subject sequence *D* has the decimal form and more than DECIMAL_DIG significant digits, consider the two bounding, adjacent decimal strings *L* and *U*, both having DECIMAL_DIG significant digits, such that the values of *L*, *D*, and *U* satisfy $L \leq D \leq U$. The result should be one of the (equal or adjacent) values that would be obtained by correctly rounding *L* and *U* according to the current rounding direction, with the extra stipulation that the error with respect to *D* should have a correct sign for the current rounding direction.

The changes to `strtod()` introduced by the ISO/IEC 9899:1999 standard can alter the behavior of well-formed applications complying with the ISO/IEC 9899:1990 standard and thus earlier versions of this standard. One such example would be:

```
int
what_kind_of_number (char *s)
{
    char *endp;
    double d;
    long l;

    d = strtod(s, &endp);
    if (s != endp && *endp == '\0')
        printf("It's a float with value %g\n", d);
    else
    {
        l = strtol(s, &endp, 0);
        if (s != endp && *endp == '\0')
            printf("It's an integer with value %ld\n", l);
        else
            return 1;
    }
    return 0;
}
```

If the function is called with:

```
what_kind_of_number ("0x10")
```

an ISO/IEC 9899:1990 standard-compliant library will result in the function printing:

```
It's an integer with value 16
```

With the ISO/IEC 9899:1999 standard, the result is:

```
It's a float with value 16
```

The change in behavior is due to the inclusion of floating-point numbers in hexadecimal notation without requiring that either a decimal point or the binary exponent be present.

65611 **RATIONALE**

65612 None.

65613 **FUTURE DIRECTIONS**

65614 None.

65615 **SEE ALSO**65616 *fscanf()*, *isspace()*, *localeconv()*, *setlocale()*, *strtol()*65617 XBD Chapter 7 (on page 135), *<float.h>*, *<stdlib.h>*65618 **CHANGE HISTORY**

65619 First released in Issue 1. Derived from Issue 1 of the SVID.

65620 **Issue 5**65621 The DESCRIPTION is updated to indicate that *errno* is not changed if the function is successful.65622 **Issue 6**

65623 Extensions beyond the ISO C standard are marked.

65624 The following new requirements on POSIX implementations derive from alignment with the
65625 Single UNIX Specification:

- 65626 • In the RETURN VALUE and ERRORS sections, the [EINVAL] optional error condition is
- 65627 added if no conversion could be performed.

65628 The following changes are made for alignment with the ISO/IEC 9899:1999 standard:

- 65629 • The *strtod()* function is updated.
- 65630 • The *strtof()* and *strtold()* functions are added.
- 65631 • The DESCRIPTION is extensively revised.

65632 ISO/IEC 9899:1999 standard, Technical Corrigendum 1 is incorporated.

65633 IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/61 is applied, correcting the second
65634 paragraph in the RETURN VALUE section. This change clarifies the sign of the return value.65635 **Issue 7**

65636 Austin Group Interpretation 1003.1-2001 #015 is applied.

65637 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0610 [302], XSH/TC1-2008/0611 [94], +
65638 and XSH/TC1-2008/0612 [105] are applied.

NAME

strtoimax, strtoumax — convert string to integer type

SYNOPSIS

```
#include <inttypes.h>
```

```
intmax_t strtoimax(const char *restrict nptr, char **restrict endptr,
    int base);
```

```
uintmax_t strtoumax(const char *restrict nptr, char **restrict endptr,
    int base);
```

DESCRIPTION

The functionality described on this reference page is aligned with the ISO C standard. Any conflict between the requirements described here and the ISO C standard is unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

These functions shall be equivalent to the *strtol()*, *strtoll()*, *strtoul()*, and *strtoull()* functions, except that the initial portion of the string shall be converted to **intmax_t** and **uintmax_t** representation, respectively.

RETURN VALUE

These functions shall return the converted value, if any.

If no conversion could be performed, zero shall be returned and *errno* may be set to [EINVAL].

If the value of *base* is not supported, 0 shall be returned and *errno* shall be set to [EINVAL].

If the correct value is outside the range of representable values, {INTMAX_MAX}, {INTMAX_MIN}, or {UINTMAX_MAX} shall be returned (according to the return type and sign of the value, if any), and *errno* shall be set to [ERANGE].

ERRORS

These functions shall fail if:

[EINVAL] The value of *base* is not supported.

[ERANGE] The value to be returned is not representable.

These functions may fail if:

[EINVAL] No conversion could be performed.

EXAMPLES

None.

APPLICATION USAGE

Since the value of **endptr* is unspecified if the value of *base* is not supported, applications should either ensure that *base* has a supported value (0 or between 2 and 36) before the call, or check for an [EINVAL] error before examining **endptr*.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

strtol(), *strtoul()*

XBD <inttypes.h>

CHANGE HISTORY

65680
65681 First released in Issue 6. Derived from the ISO/IEC 9899: 1999 standard.

Issue 7

65682
65683 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0613 [453] and XSH/TC1-2008/0614 +
65684 [453] are applied.

65685 NAME

65686 strtok, strtok_r — split string into tokens

65687 SYNOPSIS

```
65688        #include <string.h>
65689        char *strtok(char *restrict s1, const char *restrict s2);
65690 CX       char *strtok_r(char *restrict s, const char *restrict sep,
65691                        char **restrict lasts);
```

65692 DESCRIPTION

65693 CX For *strtok()*: The functionality described on this reference page is aligned with the ISO C
65694 standard. Any conflict between the requirements described here and the ISO C standard is
65695 unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

65696 A sequence of calls to *strtok()* breaks the string pointed to by *s1* into a sequence of tokens, each
65697 of which is delimited by a byte from the string pointed to by *s2*. The first call in the sequence
65698 has *s1* as its first argument, and is followed by calls with a null pointer as their first argument.
65699 The separator string pointed to by *s2* may be different from call to call.

65700 The first call in the sequence searches the string pointed to by *s1* for the first byte that is *not*
65701 contained in the current separator string pointed to by *s2*. If no such byte is found, then there
65702 are no tokens in the string pointed to by *s1* and *strtok()* shall return a null pointer. If such a byte
65703 is found, it is the start of the first token.

65704 The *strtok()* function then searches from there for a byte that *is* contained in the current separator
65705 string. If no such byte is found, the current token extends to the end of the string pointed to by
65706 *s1*, and subsequent searches for a token shall return a null pointer. If such a byte is found, it is
65707 overwritten by a NUL character, which terminates the current token. The *strtok()* function saves
65708 a pointer to the following byte, from which the next search for a token shall start.

65709 Each subsequent call, with a null pointer as the value of the first argument, starts searching from
65710 the saved pointer and behaves as described above.

65711 The implementation shall behave as if no function defined in this volume of POSIX.1-2008 calls
65712 *strtok()*.

65713 CX The *strtok()* function need not be thread-safe.

65714 The *strtok_r()* function considers the null-terminated string *s* as a sequence of zero or more text
65715 tokens separated by spans of one or more characters from the separator string *sep*. The
65716 argument *lasts* points to a user-provided pointer which points to stored information necessary
65717 for *strtok_r()* to continue scanning the same string.

65718 In the first call to *strtok_r()*, *s* points to a null-terminated string, *sep* to a null-terminated string of
65719 separator characters, and the value pointed to by *lasts* is ignored. The *strtok_r()* function shall
65720 return a pointer to the first character of the first token, write a null character into *s* immediately
65721 following the returned token, and update the pointer to which *lasts* points.

65722 In subsequent calls, *s* is a null pointer and *lasts* shall be unchanged from the previous call so that
65723 subsequent calls shall move through the string *s*, returning successive tokens until no tokens
65724 remain. The separator string *sep* may be different from call to call. When no token remains in *s*, a
65725 null pointer shall be returned.

65726 **RETURN VALUE**

65727 Upon successful completion, *strtok()* shall return a pointer to the first byte of a token. Otherwise,
 65728 if there is no token, *strtok()* shall return a null pointer.

65729 CX The *strtok_r()* function shall return a pointer to the token found, or a null pointer when no token
 65730 is found.

65731 **ERRORS**

65732 No errors are defined.

65733 **EXAMPLES**65734 **Searching for Word Separators**

65735 The following example searches for tokens separated by <space> characters.

```
65736 #include <string.h>
65737 ...
65738 char *token;
65739 char line[] = "LINE TO BE SEPARATED";
65740 char *search = " ";

65741 /* Token will point to "LINE". */
65742 token = strtok(line, search);

65743 /* Token will point to "TO". */
65744 token = strtok(NULL, search);
```

65745 **Find First two Fields in a Buffer**

65746 The following example uses *strtok()* to find two character strings (a key and data associated with
 65747 that key) separated by any combination of <space>, <tab>, or <newline> characters at the start
 65748 of the array of characters pointed to by *buffer*.

```
65749 #include <string.h>
65750 ...
65751 char    *buffer;
65752 ...
65753 struct element {
65754     char *key;
65755     char *data;
65756 } e;
65757 ...
65758 // Load the buffer...
65759 ...
65760 // Get the key and its data...
65761 e.key = strtok(buffer, " \t\n");
65762 e.data = strtok(NULL, " \t\n");
65763 // Process the rest of the contents of the buffer...
65764 ...
```

65765 **APPLICATION USAGE**

65766 The *strtok_r()* function is thread-safe and stores its state in a user-supplied buffer instead of
 65767 possibly using a static data area that may be overwritten by an unrelated call from another
 65768 thread.

RATIONALE

The *strtok()* function searches for a separator string within a larger string. It returns a pointer to the last substring between separator strings. This function uses static storage to keep track of the current string position between calls. The new function, *strtok_r()*, takes an additional argument, *lasts*, to keep track of the current position in the string.

FUTURE DIRECTIONS

None.

SEE ALSO

XBD [<string.h>](#)

CHANGE HISTORY

First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 5

The *strtok_r()* function is included for alignment with the POSIX Threads Extension.

A note indicating that the *strtok()* function need not be reentrant is added to the DESCRIPTION.

Issue 6

Extensions beyond the ISO C standard are marked.

The *strtok_r()* function is marked as part of the Thread-Safe Functions option.

In the DESCRIPTION, the note about reentrancy is expanded to cover thread-safety.

The APPLICATION USAGE section is updated to include a note on the thread-safe function and its avoidance of possibly using a static data area.

The **restrict** keyword is added to the *strtok()* and *strtok_r()* prototypes for alignment with the ISO/IEC 9899:1999 standard.

Issue 7

Austin Group Interpretation 1003.1-2001 #156 is applied.

SD5-XSH-ERN-235 is applied, correcting an example.

The *strtok_r()* function is moved from the Thread-Safe Functions option to the Base.

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0615 [177] is applied.

+

65796 NAME

65797 strtol, strtoll — convert a string to a long integer

65798 SYNOPSIS

65799 #include <stdlib.h>

65800 long strtol(const char *restrict str, char **restrict endptr, int base);

65801 long long strtoll(const char *restrict str, char **restrict endptr,

65802 int base)

65803 DESCRIPTION

65804 CX The functionality described on this reference page is aligned with the ISO C standard. Any
 65805 conflict between the requirements described here and the ISO C standard is unintentional. This
 65806 volume of POSIX.1-2008 defers to the ISO C standard.

65807 These functions shall convert the initial portion of the string pointed to by *str* to a type **long** and
 65808 **long long** representation, respectively. First, they decompose the input string into three parts:

- 65809 1. An initial, possibly empty, sequence of white-space characters (as specified by *isspace()*)
- 65810 2. A subject sequence interpreted as an integer represented in some radix determined by the
 65811 value of *base*
- 65812 3. A final string of one or more unrecognized characters, including the terminating NUL
 65813 character of the input string.

65814 Then they shall attempt to convert the subject sequence to an integer, and return the result.

65815 If the value of *base* is 0, the expected form of the subject sequence is that of a decimal constant,
 65816 octal constant, or hexadecimal constant, any of which may be preceded by a '+' or '-' sign. A
 65817 decimal constant begins with a non-zero digit, and consists of a sequence of decimal digits. An
 65818 octal constant consists of the prefix '0' optionally followed by a sequence of the digits '0' to
 65819 '7' only. A hexadecimal constant consists of the prefix 0x or 0X followed by a sequence of the
 65820 decimal digits and letters 'a' (or 'A') to 'f' (or 'F') with values 10 to 15 respectively.

65821 If the value of *base* is between 2 and 36, the expected form of the subject sequence is a sequence
 65822 of letters and digits representing an integer with the radix specified by *base*, optionally preceded
 65823 by a '+' or '-' sign. The letters from 'a' (or 'A') to 'z' (or 'Z') inclusive are ascribed the
 65824 values 10 to 35; only letters whose ascribed values are less than that of *base* are permitted. If the
 65825 value of *base* is 16, the characters 0x or 0X may optionally precede the sequence of letters and
 65826 digits, following the sign if present.

65827 The subject sequence is defined as the longest initial subsequence of the input string, starting
 65828 with the first non-white-space character that is of the expected form. The subject sequence shall
 65829 contain no characters if the input string is empty or consists entirely of white-space characters,
 65830 or if the first non-white-space character is other than a sign or a permissible letter or digit.

65831 If the subject sequence has the expected form and the value of *base* is 0, the sequence of
 65832 characters starting with the first digit shall be interpreted as an integer constant. If the subject
 65833 sequence has the expected form and the value of *base* is between 2 and 36, it shall be used as the
 65834 base for conversion, ascribing to each letter its value as given above. If the subject sequence
 65835 begins with a minus-sign, the value resulting from the conversion shall be negated. A pointer to
 65836 the final string shall be stored in the object pointed to by *endptr*, provided that *endptr* is not a null
 65837 pointer.

65838 CX In other than the C or POSIX locales, other implementation-defined subject sequences may be
 65839 accepted.

65840 If the subject sequence is empty or does not have the expected form, no conversion is performed;

65841		the value of <i>str</i> shall be stored in the object pointed to by <i>endptr</i> , provided that <i>endptr</i> is not a	
65842		null pointer.	
65843		These functions shall not change the setting of <i>errno</i> if successful.	
65844		Since 0, {LONG_MIN} or {LLONG_MIN}, and {LONG_MAX} or {LLONG_MAX} are returned	
65845		on error and are also valid returns on success, an application wishing to check for error	
65846		situations should set <i>errno</i> to 0, then call <i>strtol()</i> or <i>strtoll()</i> , then check <i>errno</i> .	-
65847		RETURN VALUE	
65848		Upon successful completion, these functions shall return the converted value, if any. If no	
65849	CX	conversion could be performed, 0 shall be returned and <i>errno</i> may be set to [EINVAL].	
65850	CX	If the value of <i>base</i> is not supported, 0 shall be returned and <i>errno</i> shall be set to [EINVAL].	+
65851		If the correct value is outside the range of representable values, {LONG_MIN}, {LONG_MAX},	
65852		{LLONG_MIN}, or {LLONG_MAX} shall be returned (according to the sign of the value), and	
65853		<i>errno</i> set to [ERANGE].	
65854		ERRORS	
65855		These functions shall fail if:	
65856	CX	[EINVAL] The value of <i>base</i> is not supported.	-
65857		[ERANGE] The value to be returned is not representable.	+
65858		These functions may fail if:	+
65859		[EINVAL] No conversion could be performed.	
65860		EXAMPLES	
65861		None.	
65862		APPLICATION USAGE	
65863		Since the value of <i>*endptr</i> is unspecified if the value of <i>base</i> is not supported, applications should	
65864		either ensure that <i>base</i> has a supported value (0 or between 2 and 36) before the call, or check for	
65865		an [EINVAL] error before examining <i>*endptr</i> .	
65866		RATIONALE	
65867		None.	
65868		FUTURE DIRECTIONS	
65869		None.	
65870		SEE ALSO	
65871		<i>fscanf()</i> , <i>isalpha()</i> , <i>strtod()</i>	
65872		XBD <stdlib.h>	
65873		CHANGE HISTORY	
65874		First released in Issue 1. Derived from Issue 1 of the SVID.	
65875		Issue 5	
65876		The DESCRIPTION is updated to indicate that <i>errno</i> is not changed if the function is successful.	
65877		Issue 6	
65878		Extensions beyond the ISO C standard are marked.	
65879		The following new requirements on POSIX implementations derive from alignment with the	
65880		Single UNIX Specification:	

- In the RETURN VALUE and ERRORS sections, the [EINVAL] optional error condition is added if no conversion could be performed.

The following changes are made for alignment with the ISO/IEC 9899:1999 standard:

- The *strtol()* prototype is updated.
- The *strtoll()* function is added.

Issue 7

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0616 [453], XSH/TC1-2008/0617 [105], +
XSH/TC1-2008/0618 [453], XSH/TC1-2008/0619 [453], and XSH/TC1-2008/0620 [453] are +
applied.

65890 **NAME**

65891 strtold — convert a string to a double-precision number

65892 **SYNOPSIS**

65893 #include <stdlib.h>

65894 long double strtold(const char *restrict *nptr*, char **restrict *endptr*);

65895 **DESCRIPTION**

65896 Refer to *strtod()*.

65897 **NAME**

65898 strtoll — convert a string to a long integer

65899 **SYNOPSIS**

65900 #include <stdlib.h>

```
65901        long long strtoll(const char *restrict str, char **restrict endptr,
65902                          int base);
```

65903 **DESCRIPTION**65904 Refer to *strtol()*.

NAME

strtol, strtoull — convert a string to an unsigned long

SYNOPSIS

```
#include <stdlib.h>

unsigned long strtol(const char *restrict str,
    char **restrict endptr, int base);
unsigned long long strtoull(const char *restrict str,
    char **restrict endptr, int base);
```

DESCRIPTION

CX The functionality described on this reference page is aligned with the ISO C standard. Any conflict between the requirements described here and the ISO C standard is unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

These functions shall convert the initial portion of the string pointed to by *str* to a type **unsigned long** and **unsigned long long** representation, respectively. First, they decompose the input string into three parts:

1. An initial, possibly empty, sequence of white-space characters (as specified by *isspace()*)
2. A subject sequence interpreted as an integer represented in some radix determined by the value of *base*
3. A final string of one or more unrecognized characters, including the terminating NUL character of the input string

Then they shall attempt to convert the subject sequence to an unsigned integer, and return the result.

If the value of *base* is 0, the expected form of the subject sequence is that of a decimal constant, octal constant, or hexadecimal constant, any of which may be preceded by a '+' or '-' sign. A decimal constant begins with a non-zero digit, and consists of a sequence of decimal digits. An octal constant consists of the prefix '0' optionally followed by a sequence of the digits '0' to '7' only. A hexadecimal constant consists of the prefix 0x or 0X followed by a sequence of the decimal digits and letters 'a' (or 'A') to 'f' (or 'F') with values 10 to 15 respectively.

If the value of *base* is between 2 and 36, the expected form of the subject sequence is a sequence of letters and digits representing an integer with the radix specified by *base*, optionally preceded by a '+' or '-' sign. The letters from 'a' (or 'A') to 'z' (or 'Z') inclusive are ascribed the values 10 to 35; only letters whose ascribed values are less than that of *base* are permitted. If the value of *base* is 16, the characters 0x or 0X may optionally precede the sequence of letters and digits, following the sign if present.

The subject sequence is defined as the longest initial subsequence of the input string, starting with the first non-white-space character that is of the expected form. The subject sequence shall contain no characters if the input string is empty or consists entirely of white-space characters, or if the first non-white-space character is other than a sign or a permissible letter or digit.

If the subject sequence has the expected form and the value of *base* is 0, the sequence of characters starting with the first digit shall be interpreted as an integer constant. If the subject sequence has the expected form and the value of *base* is between 2 and 36, it shall be used as the base for conversion, ascribing to each letter its value as given above. If the subject sequence begins with a minus-sign, the value resulting from the conversion shall be negated. A pointer to the final string shall be stored in the object pointed to by *endptr*, provided that *endptr* is not a null pointer.

65950 CX In other than the C or POSIX locales, other implementation-defined subject sequences may be
65951 accepted.

65952 If the subject sequence is empty or does not have the expected form, no conversion shall be
65953 performed; the value of *str* shall be stored in the object pointed to by *endptr*, provided that *endptr*
65954 is not a null pointer.

65955 These functions shall not change the setting of *errno* if successful. |

65956 Since 0, {ULONG_MAX}, and {ULLONG_MAX} are returned on error and are also valid returns
65957 on success, an application wishing to check for error situations should set *errno* to 0, then call
65958 *strtoul()* or *strtoull()*, then check *errno*. -

65959 RETURN VALUE

65960 Upon successful completion, these functions shall return the converted value, if any. If no
65961 CX conversion could be performed, 0 shall be returned and *errno* may be set to [EINVAL].

65962 CX If the value of *base* is not supported, 0 shall be returned and *errno* shall be set to [EINVAL]. +

65963 If the correct value is outside the range of representable values, {ULONG_MAX} or
65964 {ULLONG_MAX} shall be returned and *errno* set to [ERANGE].

65965 ERRORS

65966 These functions shall fail if:

65967 CX [EINVAL] The value of *base* is not supported.

65968 [ERANGE] The value to be returned is not representable.

65969 These functions may fail if:

65970 CX [EINVAL] No conversion could be performed.

65971 EXAMPLES

65972 None.

65973 APPLICATION USAGE

65974 Since the value of **endptr* is unspecified if the value of *base* is not supported, applications should |
65975 either ensure that *base* has a supported value (0 or between 2 and 36) before the call, or check for |
65976 an [EINVAL] error before examining **endptr*.

65977 RATIONALE

65978 None.

65979 FUTURE DIRECTIONS

65980 None.

65981 SEE ALSO

65982 *fscanf()*, *isalpha()*, *strtod()*, *strtol()*

65983 XBD <stdlib.h>

65984 CHANGE HISTORY

65985 First released in Issue 4. Derived from the ANSI C standard.

65986 Issue 5

65987 The DESCRIPTION is updated to indicate that *errno* is not changed if the function is successful.

65988 **Issue 6**

65989 Extensions beyond the ISO C standard are marked.

65990 The following new requirements on POSIX implementations derive from alignment with the
65991 Single UNIX Specification:

- 65992 • The [EINVAL] error condition is added for when the value of *base* is not supported.
- 65993 • In the RETURN VALUE and ERRORS sections, the [EINVAL] optional error condition is
65994 added if no conversion could be performed.

65995 The following changes are made for alignment with the ISO/IEC 9899:1999 standard:

- 65996 • The *strtoul()* prototype is updated.
- 65997 • The *strtoull()* function is added.

65998 **Issue 7**

65999 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0621 [105], XSH/TC1-2008/0622 [453], +
66000 and XSH/TC1-2008/0623 [453] are applied.

66001 **NAME**

66002 strtoumax — convert a string to an integer type

66003 **SYNOPSIS**

66004 #include <inttypes.h>

```
66005       uintmax_t strtoumax(const char *restrict nptr, char **restrict endptr,
66006                           int base);
```

66007 **DESCRIPTION**66008 Refer to *strtoimax()*.

66009 **NAME**

66010 strxfrm, strxfrm_l — string transformation

66011 **SYNOPSIS**

```
66012 #include <string.h>
66013 size_t strxfrm(char *restrict s1, const char *restrict s2, size_t n);
66014 CX size_t strxfrm_l(char *restrict s1, const char *restrict s2,
66015 size_t n, locale_t locale);
```

66016 **DESCRIPTION**

66017 CX For *strxfrm()*: The functionality described on this reference page is aligned with the ISO C
 66018 standard. Any conflict between the requirements described here and the ISO C standard is
 66019 unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

66020 CX The *strxfrm()* and *strxfrm_l()* functions shall transform the string pointed to by *s2* and place the
 66021 resulting string into the array pointed to by *s1*. The transformation is such that if *strcmp()* is
 66022 applied to two transformed strings, it shall return a value greater than, equal to, or less than 0,
 66023 CX corresponding to the result of *strcoll()* or *strcoll_l()*, respectively, applied to the same two
 66024 CX original strings with the same locale. No more than *n* bytes are placed into the resulting array
 66025 pointed to by *s1*, including the terminating NUL character. If *n* is 0, *s1* is permitted to be a null
 66026 pointer. If copying takes place between objects that overlap, the behavior is undefined.

66027 CX The *strxfrm()* and *strxfrm_l()* functions shall not change the setting of *errno* if successful.

66028 Since no return value is reserved to indicate an error, an application wishing to check for error
 66029 CX situations should set *errno* to 0, then call *strxfrm()* or *strxfrm_l()*, then check *errno*.

66030 CX The behavior is undefined if the *locale* argument to *strxfrm_l()* is the special locale object +
 66031 LC_GLOBAL_LOCALE or is not a valid locale object handle.

66032 **RETURN VALUE**

66033 CX Upon successful completion, *strxfrm()* and *strxfrm_l()* shall return the length of the
 66034 transformed string (not including the terminating NUL character). If the value returned is *n* or
 66035 more, the contents of the array pointed to by *s1* are unspecified.

66036 CX On error, *strxfrm()* and *strxfrm_l()* may set *errno* but no return value is reserved to indicate an
 66037 error.

66038 **ERRORS**

66039 These functions may fail if:

66040 CX [EINVAL] The string pointed to by the *s2* argument contains characters outside the
 66041 domain of the collating sequence.

66042 **EXAMPLES**

66043 None.

66044 **APPLICATION USAGE**

66045 The transformation function is such that two transformed strings can be ordered by *strcmp()* as
 66046 appropriate to collating sequence information in the current locale (category *LC_COLLATE*). |

66047 The fact that when *n* is 0 *s1* is permitted to be a null pointer is useful to determine the size of the
 66048 *s1* array prior to making the transformation.

66049 **RATIONALE**

66050 None.

66051 **FUTURE DIRECTIONS**

66052 None.

66053 **SEE ALSO**66054 *strcmp()*, *strcoll()*

66055 XBD <string.h>

66056 **CHANGE HISTORY**

66057 First released in Issue 3. Included for alignment with the ISO C standard.

66058 **Issue 5**66059 The DESCRIPTION is updated to indicate that *errno* does not change if the function is successful.66060 **Issue 6**

66061 Extensions beyond the ISO C standard are marked.

66062 The following new requirements on POSIX implementations derive from alignment with the
66063 Single UNIX Specification:

- 66064 • In the RETURN VALUE and ERRORS sections, the [EINVAL] optional error condition is
66065 added if no conversion could be performed.

66066 The *strxfrm()* prototype is updated for alignment with the ISO/IEC 9899:1999 standard.66067 **Issue 7**66068 The *strxfrm_l()* function is added from The Open Group Technical Standard, 2006, Extended API
66069 Set Part 4.66070 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0624 [283], XSH/TC1-2008/0625 [283], +
66071 and XSH/TC1-2008/0626 [302] are applied.

66072 **NAME**

66073 swab — swap bytes

66074 **SYNOPSIS**

```
66075 XSI      #include <unistd.h>
66076          void swab(const void *restrict src, void *restrict dest,
66077                   ssize_t nbytes);
```

66078 **DESCRIPTION**

66079 The *swab()* function shall copy *nbytes* bytes, which are pointed to by *src*, to the object pointed to
66080 by *dest*, exchanging adjacent bytes. The *nbytes* argument should be even. If *nbytes* is odd, *swab()*
66081 copies and exchanges *nbytes*–1 bytes and the disposition of the last byte is unspecified. If
66082 copying takes place between objects that overlap, the behavior is undefined. If *nbytes* is
66083 negative, *swab()* does nothing.

66084 **RETURN VALUE**

66085 None.

66086 **ERRORS**

66087 No errors are defined.

66088 **EXAMPLES**

66089 None.

66090 **APPLICATION USAGE**

66091 None.

66092 **RATIONALE**

66093 None.

66094 **FUTURE DIRECTIONS**

66095 None.

66096 **SEE ALSO**66097 XBD [<unistd.h>](#)66098 **CHANGE HISTORY**

66099 First released in Issue 1. Derived from Issue 1 of the SVID.

66100 **Issue 6**

66101 The **restrict** keyword is added to the *swab()* prototype for alignment with the
66102 ISO/IEC 9899:1999 standard.

66103 **NAME**

66104 swprintf — print formatted wide-character output

66105 **SYNOPSIS**

66106 #include <stdio.h>

66107 #include <wchar.h>

66108 int swprintf(wchar_t *restrict *ws*, size_t *n*,66109 const wchar_t *restrict *format*, ...);66110 **DESCRIPTION**66111 Refer to *fwprintf()*.

66112 **NAME**

66113 swscanf — convert formatted wide-character input

66114 **SYNOPSIS**

66115 #include <stdio.h>

66116 #include <wchar.h>

```
66117       int swscanf(const wchar_t *restrict ws,  
66118                   const wchar_t *restrict format, ...);
```

66119 **DESCRIPTION**66120 Refer to *fwscanf()*.

66121 **NAME**66122 `symlink`, `symlinkat` — make a symbolic link relative to directory file descriptor66123 **SYNOPSIS**66124 `#include <unistd.h>`66125 `int symlink(const char *path1, const char *path2);`66126 `int symlinkat(const char *path1, int fd, const char *path2);`66127 **DESCRIPTION**66128 The `symlink()` function shall create a symbolic link called *path2* that contains the string pointed to
66129 by *path1* (*path2* is the name of the symbolic link created, *path1* is the string contained in the
66130 symbolic link).66131 The string pointed to by *path1* shall be treated only as a character string and shall not be
66132 validated as a pathname.66133 If the `symlink()` function fails for any reason other than [EIO], any file named by *path2* shall be
66134 unaffected.66135 If *path2* names a symbolic link, `symlink()` shall fail and set *errno* to [EEXIST].

+

66136 The symbolic link's user ID shall be set to the process' effective user ID. The symbolic link's
66137 group ID shall be set to the group ID of the parent directory or to the effective group ID of the
66138 process. Implementations shall provide a way to initialize the symbolic link's group ID to the
66139 group ID of the parent directory. Implementations may, but need not, provide an
66140 implementation-defined way to initialize the symbolic link's group ID to the effective group ID
66141 of the calling process.66142 The values of the file mode bits for the created symbolic link are unspecified. All interfaces
66143 specified by POSIX.1-2008 shall behave as if the contents of symbolic links can always be read,
66144 except that the value of the file mode bits returned in the *st_mode* field of the **stat** structure is
66145 unspecified.66146 Upon successful completion, `symlink()` shall mark for update the last data access, last data
66147 modification, and last file status change timestamps of the symbolic link. Also, the last data
66148 modification and last file status change timestamps of the directory that contains the new entry
66149 shall be marked for update.66150 The `symlinkat()` function shall be equivalent to the `symlink()` function except in the case where
66151 *path2* specifies a relative path. In this case the symbolic link is created relative to the directory
66152 associated with the file descriptor *fd* instead of the current working directory. If the file
66153 descriptor was opened without O_SEARCH, the function shall check whether directory searches
66154 are permitted using the current permissions of the directory underlying the file descriptor. If the
66155 file descriptor was opened with O_SEARCH, the function shall not perform the check.66156 If `symlinkat()` is passed the special value AT_FDCWD in the *fd* parameter, the current working
66157 directory shall be used and the behavior shall be identical to a call to `symlink()`.

|

66158 **RETURN VALUE**66159 Upon successful completion, these functions shall return 0. Otherwise, these functions shall
66160 return -1 and set *errno* to indicate the error.66161 **ERRORS**

66162 These functions shall fail if:

66163 [EACCES] Write permission is denied in the directory where the symbolic link is being
66164 created, or search permission is denied for a component of the path prefix of
66165 *path2*.

66166	[EEXIST]	The <i>path2</i> argument names an existing file.	
66167	[EIO]	An I/O error occurs while reading from or writing to the file system.	
66168	[ELOOP]	A loop exists in symbolic links encountered during resolution of the <i>path2</i> argument.	
66169			
66170	[ENAMETOOLONG]		
66171		The length of a component of the pathname specified by the <i>path2</i> argument is longer than {NAME_MAX} or the length of the <i>path1</i> argument is longer than {SYMLINK_MAX}.	
66172			
66173			
66174	[ENOENT]	A component of the path prefix of <i>path2</i> does not name an existing file or <i>path2</i> is an empty string.	
66175			
66176	[ENOSPC]	The directory in which the entry for the new symbolic link is being placed cannot be extended because no space is left on the file system containing the directory, or the new symbolic link cannot be created because no space is left on the file system which shall contain the link, or the file system is out of file-allocation resources.	
66177			
66178			
66179			
66180			
66181	[ENOTDIR]	A component of the path prefix of <i>path2</i> names an existing file that is neither a directory nor a symbolic link to a directory.	
66182			
66183	[EROFS]	The new symbolic link would reside on a read-only file system.	
66184		The <i>symlinkat</i> () function shall fail if:	
66185	[EACCES]	<i>fd</i> was not opened with O_SEARCH and the permissions of the directory underlying <i>fd</i> do not permit directory searches.	
66186			
66187	[EBADF]	The <i>path2</i> argument does not specify an absolute path and the <i>fd</i> argument is neither AT_FDCWD nor a valid file descriptor open for reading or searching.	
66188			
66189	[ENOTDIR]	The <i>path2</i> argument is not an absolute path and <i>fd</i> is a file descriptor associated with a non-directory file.	+
66190			
66191		These functions may fail if:	
66192	[ELOOP]	More than {SYMLOOP_MAX} symbolic links were encountered during resolution of the <i>path2</i> argument.	
66193			
66194	[ENAMETOOLONG]		
66195		The length of the <i>path2</i> argument exceeds {PATH_MAX} or pathname resolution of a symbolic link in the <i>path2</i> argument produced an intermediate result with a length that exceeds {PATH_MAX}.	
66196			
66197			

EXAMPLES

None.

APPLICATION USAGE

Like a hard link, a symbolic link allows a file to have multiple logical names. The presence of a hard link guarantees the existence of a file, even after the original name has been removed. A symbolic link provides no such assurance; in fact, the file named by the *path1* argument need not exist when the link is created. A symbolic link can cross file system boundaries.

Normal permission checks are made on each component of the symbolic link pathname during its resolution.

RATIONALE

The purpose of the *symlinkat()* function is to create symbolic links in directories other than the current working directory without exposure to race conditions. Any part of the path of a file could be changed in parallel to a call to *symlink()*, resulting in unspecified behavior. By opening a file descriptor for the target directory and using the *symlinkat()* function it can be guaranteed that the created symbolic link is located relative to the desired directory.

FUTURE DIRECTIONS

None.

SEE ALSO

fdopendir(), *fstatat()*, *lchown()*, *link()*, *open()*, *readlink()*, *rename()*, *unlink()*

XBD <unistd.h>

CHANGE HISTORY

First released in Issue 4, Version 2.

Issue 5

Moved from X/OPEN UNIX extension to BASE.

Issue 6

The following changes were made to align with the IEEE P1003.1a draft standard:

- The DESCRIPTION text is updated.
- The [ELOOP] optional error condition is added.

Issue 7

Austin Group Interpretation 1003.1-2001 #143 is applied.

The *symlinkat()* function is added from The Open Group Technical Standard, 2006, Extended API Set Part 2.

Additions have been made describing how *symlink()* sets the user and group IDs and file mode of the symbolic link, and its effect on timestamps.

Changes are made to allow a directory to be opened for searching.

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0627 [146,428], XSH/TC1-2008/0628 [461], XSH/TC1-2008/0629 [146,428], XSH/TC1-2008/0630 [146,428,436], XSH/TC1-2008/0631 [324], XSH/TC1-2008/0632 [278], XSH/TC1-2008/0633 [278], and XSH/TC1-2008/0634 [151] are applied.

66237 **NAME**

66238 sync — schedule file system updates

66239 **SYNOPSIS**

```
66240 XSI      #include <unistd.h>
66241          void sync(void);
```

66242 **DESCRIPTION**

66243 The *sync()* function shall cause all information in memory that updates file systems to be
66244 scheduled for writing out to all file systems.

66245 The writing, although scheduled, is not necessarily complete upon return from *sync()*.

66246 **RETURN VALUE**

66247 The *sync()* function shall not return a value.

66248 **ERRORS**

66249 No errors are defined.

66250 **EXAMPLES**

66251 None.

66252 **APPLICATION USAGE**

66253 None.

66254 **RATIONALE**

66255 None.

66256 **FUTURE DIRECTIONS**

66257 None.

66258 **SEE ALSO**

66259 [*fsync\(\)*](#)

66260 XBD [**<unistd.h>**](#)

66261 **CHANGE HISTORY**

66262 First released in Issue 4, Version 2.

66263 **Issue 5**

66264 Moved from X/OPEN UNIX extension to BASE.

66265 **NAME**

66266 sysconf — get configurable system variables

66267 **SYNOPSIS**

66268 #include <unistd.h>

66269 long sysconf(int name);

66270 **DESCRIPTION**

66271 The *sysconf()* function provides a method for the application to determine the current value of a
 66272 configurable system limit or option (*variable*). The implementation shall support all of the
 66273 variables listed in the following table and may support others.

66274 The *name* argument represents the system variable to be queried. The following table lists the
 66275 minimal set of system variables from <limits.h> or <unistd.h> that can be returned by *sysconf()*,
 66276 and the symbolic constants defined in <unistd.h> that are the corresponding values used for
 66277 *name*.

Variable	Value of Name
{AIO_LISTIO_MAX}	_SC_AIO_LISTIO_MAX
{AIO_MAX}	_SC_AIO_MAX
{AIO_PRIO_DELTA_MAX}	_SC_AIO_PRIO_DELTA_MAX
{ARG_MAX}	_SC_ARG_MAX
{ATEXIT_MAX}	_SC_ATEXIT_MAX
{BC_BASE_MAX}	_SC_BC_BASE_MAX
{BC_DIM_MAX}	_SC_BC_DIM_MAX
{BC_SCALE_MAX}	_SC_BC_SCALE_MAX
{BC_STRING_MAX}	_SC_BC_STRING_MAX
{CHILD_MAX}	_SC_CHILD_MAX
Clock ticks/second	_SC_CLK_TCK
{COLL_WEIGHTS_MAX}	_SC_COLL_WEIGHTS_MAX
{DELAYTIMER_MAX}	_SC_DELAYTIMER_MAX
{EXPR_NEST_MAX}	_SC_EXPR_NEST_MAX
{HOST_NAME_MAX}	_SC_HOST_NAME_MAX
{IOV_MAX}	_SC_IOV_MAX
{LINE_MAX}	_SC_LINE_MAX
{LOGIN_NAME_MAX}	_SC_LOGIN_NAME_MAX
{NGROUPS_MAX}	_SC_NGROUPS_MAX
Initial size of <i>getgrgid_r()</i> and <i>getgrnam_r()</i> data buffers	_SC_GETGR_R_SIZE_MAX
Initial size of <i>getpwuid_r()</i> and <i>getpwnam_r()</i> data buffers	_SC_GETPW_R_SIZE_MAX
{MQ_OPEN_MAX}	_SC_MQ_OPEN_MAX
{MQ_PRIO_MAX}	_SC_MQ_PRIO_MAX
{OPEN_MAX}	_SC_OPEN_MAX
_POSIX_ADVISORY_INFO	_SC_ADVISORY_INFO
_POSIX_BARRIERS	_SC_BARRIERS
_POSIX_ASYNCHRONOUS_IO	_SC_ASYNCHRONOUS_IO
_POSIX_CLOCK_SELECTION	_SC_CLOCK_SELECTION
_POSIX_CPUTIME	_SC_CPUTIME
_POSIX_FSYNC	_SC_FSYNC
_POSIX_IPV6	_SC_IPV6
_POSIX_JOB_CONTROL	_SC_JOB_CONTROL

	Variable	Value of Name
66313		
66314	_POSIX_MAPPED_FILES	_SC_MAPPED_FILES
66315	_POSIX_MEMLOCK	_SC_MEMLOCK
66316	_POSIX_MEMLOCK_RANGE	_SC_MEMLOCK_RANGE
66317	_POSIX_MEMORY_PROTECTION	_SC_MEMORY_PROTECTION
66318	_POSIX_MESSAGE_PASSING	_SC_MESSAGE_PASSING
66319	_POSIX_MONOTONIC_CLOCK	_SC_MONOTONIC_CLOCK
66320	_POSIX_PRIORITIZED_IO	_SC_PRIORITIZED_IO
66321	_POSIX_PRIORITY_SCHEDULING	_SC_PRIORITY_SCHEDULING
66322	_POSIX_RAW_SOCKETS	_SC_RAW_SOCKETS
66323	_POSIX_READER_WRITER_LOCKS	_SC_READER_WRITER_LOCKS
66324	_POSIX_REALTIME_SIGNALS	_SC_REALTIME_SIGNALS
66325	_POSIX_REGEX	_SC_REGEX
66326	_POSIX_SAVED_IDS	_SC_SAVED_IDS
66327	_POSIX_SEMAPHORES	_SC_SEMAPHORES
66328	_POSIX_SHARED_MEMORY_OBJECTS	_SC_SHARED_MEMORY_OBJECTS
66329	_POSIX_SHELL	_SC_SHELL
66330	_POSIX_SPAWN	_SC_SPAWN
66331	_POSIX_SPIN_LOCKS	_SC_SPIN_LOCKS
66332	_POSIX_SPORADIC_SERVER	_SC_SPORADIC_SERVER
66333	_POSIX_SS_REPL_MAX	_SC_SS_REPL_MAX
66334	_POSIX_SYNCHRONIZED_IO	_SC_SYNCHRONIZED_IO
66335	_POSIX_THREAD_ATTR_STACKADDR	_SC_THREAD_ATTR_STACKADDR
66336	_POSIX_THREAD_ATTR_STACKSIZE	_SC_THREAD_ATTR_STACKSIZE
66337	_POSIX_THREAD_CPUTIME	_SC_THREAD_CPUTIME
66338	_POSIX_THREAD_PRIO_INHERIT	_SC_THREAD_PRIO_INHERIT
66339	_POSIX_THREAD_PRIO_PROTECT	_SC_THREAD_PRIO_PROTECT
66340	_POSIX_THREAD_PRIORITY_SCHEDULING	_SC_THREAD_PRIORITY_SCHEDULING
66341	_POSIX_THREAD_PROCESS_SHARED	_SC_THREAD_PROCESS_SHARED
66342	_POSIX_THREAD_ROBUST_PRIO_INHERIT	_SC_THREAD_ROBUST_PRIO_INHERIT
66343	_POSIX_THREAD_ROBUST_PRIO_PROTECT	_SC_THREAD_ROBUST_PRIO_PROTECT
66344	_POSIX_THREAD_SAFE_FUNCTIONS	_SC_THREAD_SAFE_FUNCTIONS
66345	_POSIX_THREAD_SPAWNING	_SC_THREAD_SPAWNING
66346	_POSIX_THREADS	_SC_THREADS
66347	_POSIX_TIMEOUTS	_SC_TIMEOUTS
66348	_POSIX_TIMERS	_SC_TIMERS
66349	_POSIX_TRACE	_SC_TRACE
66350	_POSIX_TRACE_EVENT_FILTER	_SC_TRACE_EVENT_FILTER
66351	_POSIX_TRACE_EVENT_NAME_MAX	_SC_TRACE_EVENT_NAME_MAX
66352	_POSIX_TRACE_INHERIT	_SC_TRACE_INHERIT
66353	_POSIX_TRACE_LOG	_SC_TRACE_LOG
66354	_POSIX_TRACE_NAME_MAX	_SC_TRACE_NAME_MAX
66355	_POSIX_TRACE_SYS_MAX	_SC_TRACE_SYS_MAX
66356	_POSIX_TRACE_USER_EVENT_MAX	_SC_TRACE_USER_EVENT_MAX
66357	_POSIX_TYPED_MEMORY_OBJECTS	_SC_TYPED_MEMORY_OBJECTS
66358	_POSIX_VERSION	_SC_VERSION
66359	_POSIX_V7_ILP32_OFF32	_SC_V7_ILP32_OFF32
66360	_POSIX_V7_ILP32_OFFBIG	_SC_V7_ILP32_OFFBIG
66361	_POSIX_V7_LP64_OFF64	_SC_V7_LP64_OFF64
66362	_POSIX_V7_LPBIG_OFFBIG	_SC_V7_LPBIG_OFFBIG

Variable	Value of Name
_POSIX_V6_ILP32_OFF32	_SC_V6_ILP32_OFF32
_POSIX_V6_ILP32_OFFBIG	_SC_V6_ILP32_OFFBIG
_POSIX_V6_LP64_OFF64	_SC_V6_LP64_OFF64
_POSIX_V6_LPBIG_OFFBIG	_SC_V6_LPBIG_OFFBIG
_POSIX2_C_BIND	_SC_2_C_BIND
_POSIX2_C_DEV	_SC_2_C_DEV
_POSIX2_CHAR_TERM	_SC_2_CHAR_TERM
_POSIX2_FORT_DEV	_SC_2_FORT_DEV
_POSIX2_FORT_RUN	_SC_2_FORT_RUN
_POSIX2_LOCALEDEF	_SC_2_LOCALEDEF
_POSIX2_PBS	_SC_2_PBS
_POSIX2_PBS_ACCOUNTING	_SC_2_PBS_ACCOUNTING
_POSIX2_PBS_CHECKPOINT	_SC_2_PBS_CHECKPOINT
_POSIX2_PBS_LOCATE	_SC_2_PBS_LOCATE
_POSIX2_PBS_MESSAGE	_SC_2_PBS_MESSAGE
_POSIX2_PBS_TRACK	_SC_2_PBS_TRACK
_POSIX2_SW_DEV	_SC_2_SW_DEV
_POSIX2_UPE	_SC_2_UPE
_POSIX2_VERSION	_SC_2_VERSION
{PAGE_SIZE}	_SC_PAGE_SIZE
{PAGESIZE}	_SC_PAGESIZE
{PTHREAD_DESTRUCTOR_ITERATIONS}	_SC_THREAD_DESTRUCTOR_ITERATIONS
{PTHREAD_KEYS_MAX}	_SC_THREAD_KEYS_MAX
{PTHREAD_STACK_MIN}	_SC_THREAD_STACK_MIN
{PTHREAD_THREADS_MAX}	_SC_THREAD_THREADS_MAX
{RE_DUP_MAX}	_SC_RE_DUP_MAX
{RTSIG_MAX}	_SC_RTSIG_MAX
{SEM_NSEMS_MAX}	_SC_SEM_NSEMS_MAX
{SEM_VALUE_MAX}	_SC_SEM_VALUE_MAX
{SIGQUEUE_MAX}	_SC_SIGQUEUE_MAX
{STREAM_MAX}	_SC_STREAM_MAX
{SYMLOOP_MAX}	_SC_SYMLOOP_MAX
{TIMER_MAX}	_SC_TIMER_MAX
{TTY_NAME_MAX}	_SC_TTY_NAME_MAX
{TZNAME_MAX}	_SC_TZNAME_MAX
_XOPEN_CRYPT	_SC_XOPEN_CRYPT
_XOPEN_ENH_I18N	_SC_XOPEN_ENH_I18N
_XOPEN_REALTIME	_SC_XOPEN_REALTIME
_XOPEN_REALTIME_THREADS	_SC_XOPEN_REALTIME_THREADS
_XOPEN_SHM	_SC_XOPEN_SHM
_XOPEN_STREAMS	_SC_XOPEN_STREAMS
_XOPEN_UNIX	_SC_XOPEN_UNIX
_XOPEN_UUCP	_SC_XOPEN_UUCP
_XOPEN_VERSION	_SC_XOPEN_VERSION

RETURN VALUE

If *name* is an invalid value, *sysconf()* shall return `-1` and set *errno* to indicate the error. If the variable corresponding to *name* is described in **<limits.h>** as a maximum or minimum value and the variable has no limit, *sysconf()* shall return `-1` without changing the value of *errno*. Note that indefinite limits do not imply infinite limits; see **<limits.h>**.

Otherwise, *sysconf()* shall return the current variable value on the system. The value returned

shall not be more restrictive than the corresponding value described to the application when it was compiled with the implementation's `<limits.h>` or `<unistd.h>`. The value shall not change during the lifetime of the calling process, except that `sysconf(_SC_OPEN_MAX)` may return different values before and after a call to `setrlimit()` which changes the `RLIMIT_NOFILE` soft limit.

If the variable corresponding to *name* is dependent on an unsupported option, the results are unspecified.

ERRORS

The `sysconf()` function shall fail if:

[EINVAL] The value of the *name* argument is invalid.

EXAMPLES

None.

APPLICATION USAGE

As `-1` is a permissible return value in a successful situation, an application wishing to check for error situations should set `errno` to 0, then call `sysconf()`, and, if it returns `-1`, check to see if `errno` is non-zero.

Application developers should check whether an option, such as `_POSIX_TRACE`, is supported prior to obtaining and using values for related variables, such as `_POSIX_TRACE_NAME_MAX`.

RATIONALE

This functionality was added in response to requirements of application developers and of system vendors who deal with many international system configurations. It is closely related to `pathconf()` and `fpathconf()`.

Although a conforming application can run on all systems by never demanding more resources than the minimum values published in this volume of POSIX.1-2008, it is useful for that application to be able to use the actual value for the quantity of a resource available on any given system. To do this, the application makes use of the value of a symbolic constant in `<limits.h>` or `<unistd.h>`.

However, once compiled, the application must still be able to cope if the amount of resource available is increased. To that end, an application may need a means of determining the quantity of a resource, or the presence of an option, at execution time.

Two examples are offered:

1. Applications may wish to act differently on systems with or without job control. Applications vendors who wish to distribute only a single binary package to all instances of a computer architecture would be forced to assume job control is never available if it were to rely solely on the `<unistd.h>` value published in this volume of POSIX.1-2008.
2. International applications vendors occasionally require knowledge of the number of clock ticks per second. Without these facilities, they would be required to either distribute their applications partially in source form or to have 50 Hz and 60 Hz versions for the various countries in which they operate.

It is the knowledge that many applications are actually distributed widely in executable form that leads to this facility. If limited to the most restrictive values in the headers, such applications would have to be prepared to accept the most limited environments offered by the smallest microcomputers. Although this is entirely portable, there was a consensus that they should be able to take advantage of the facilities offered by large systems, without the restrictions associated with source and object distributions.

During the discussions of this feature, it was pointed out that it is almost always possible for an application to discern what a value might be at runtime by suitably testing the various functions themselves. And, in any event, it could always be written to adequately deal with error returns from the various functions. In the end, it was felt that this imposed an unreasonable level of complication and sophistication on the application developer.

This runtime facility is not meant to provide ever-changing values that applications have to check multiple times. The values are seen as changing no more frequently than once per system initialization, such as by a system administrator or operator with an automatic configuration program. This volume of POSIX.1-2008 specifies that they shall not change within the lifetime of the process.

Some values apply to the system overall and others vary at the file system or directory level. The latter are described in *fpathconf()*.

Note that all values returned must be expressible as integers. String values were considered, but the additional flexibility of this approach was rejected due to its added complexity of implementation and use.

Some values, such as {PATH_MAX}, are sometimes so large that they must not be used to, say, allocate arrays. The *sysconf()* function returns a negative value to show that this symbolic constant is not even defined in this case.

Similar to *pathconf()*, this permits the implementation not to have a limit. When one resource is infinite, returning an error indicating that some other resource limit has been reached is conforming behavior.

FUTURE DIRECTIONS

None.

SEE ALSO

confstr(), *fpathconf()*

XBD <limits.h>, <unistd.h>

XCU *getconf*

CHANGE HISTORY

First released in Issue 3. Included for alignment with the POSIX.1-1988 standard.

Issue 5

The DESCRIPTION is updated for alignment with the POSIX Realtime Extension and the POSIX Threads Extension.

The `_XBS_` variables and name values are added to the table of system variables in the DESCRIPTION. These are all marked EX.

Issue 6

The symbol `CLK_TCK` is obsolescent and removed. It is replaced with the phrase “clock ticks per second”.

The symbol {PASS_MAX} is removed.

The following changes were made to align with the IEEE P1003.1a draft standard:

- Table entries are added for the following variables: `_SC_REGEX`, `_SC_SHELL`, `_SC_REGEX_VERSION`, `_SC_SYMLINK_MAX`.

The following *sysconf()* variables and their associated names are added for alignment with IEEE Std 1003.1d-1999:

```

66502     _POSIX_ADVISORY_INFO
66503     _POSIX_CPUTIME
66504     _POSIX_SPAWN
66505     _POSIX_SPARADIC_SERVER
66506     _POSIX_THREAD_CPUTIME
66507     _POSIX_THREAD_SPARADIC_SERVER
66508     _POSIX_TIMEOUTS

```

66509 The following changes are made to the DESCRIPTION for alignment with IEEE Std 1003.1j-2000:

66510 • A statement expressing the dependency of support for some system variables on
66511 implementation options is added.

66512 • The following system variables are added:

```

66513     _POSIX_BARRIERS
66514     _POSIX_CLOCK_SELECTION
66515     _POSIX_MONOTONIC_CLOCK
66516     _POSIX_READER_WRITER_LOCKS
66517     _POSIX_SPIN_LOCKS
66518     _POSIX_TYPED_MEMORY_OBJECTS

```

66519 The following system variables are added for alignment with IEEE Std 1003.2d-1994:

```

66520     _POSIX2_PBS
66521     _POSIX2_PBS_ACCOUNTING
66522     _POSIX2_PBS_LOCATE
66523     _POSIX2_PBS_MESSAGE
66524     _POSIX2_PBS_TRACK

```

66525 The following *sysconf()* variables and their associated names are added for alignment with
66526 IEEE Std 1003.1q-2000:

```

66527     _POSIX_TRACE
66528     _POSIX_TRACE_EVENT_FILTER
66529     _POSIX_TRACE_INHERIT
66530     _POSIX_TRACE_LOG

```

66531 The macros associated with the *c89* programming models are marked LEGACY, and new
66532 equivalent macros associated with *c99* are introduced.

66533 IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/62 is applied, updating the
66534 DESCRIPTION to denote that the *_PC** and *_SC** symbols are now required to be supported. A
66535 corresponding change has been made in the Base Definitions volume of POSIX.1-2008. The
66536 deletion in the second paragraph removes some duplicated text. Additional symbols that were
66537 erroneously omitted from this reference page have been added.

66538 IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/63 is applied, making it clear in the
66539 RETURN VALUE section that the value returned for *sysconf(_SC_OPEN_MAX)* may change if a
66540 call to *setrlimit()* adjusts the RLIMIT_NOFILE soft limit.

66541 IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/134 is applied, updating the
66542 DESCRIPTION to remove an erroneous entry for *_POSIX_SYMLINK_MAX*. This corrects an
66543 error in IEEE Std 1003.1-2001/Cor 1-2002.

66544 IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/135 is applied, removing

66545 _POSIX_FILE_LOCKING, _POSIX_MULTI_PROCESS, _POSIX2_C_VERSION, and
66546 _XOPEN_XCU_VERSION (and their associated _SC_* variables) from the DESCRIPTION and
66547 APPLICATION USAGE sections.

66548 IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/136 is applied, adding the following
66549 constants (and their associated _SC_* variables) to the DESCRIPTION:

66550 _POSIX_SS_REPL_MAX
66551 _POSIX_TRACE_EVENT_NAME_MAX
66552 _POSIX_TRACE_NAME_MAX
66553 _POSIX_TRACE_SYS_MAX
66554 _POSIX_TRACE_USER_EVENT_MAX

66555 The RETURN VALUE and APPLICATION USAGE sections are updated to note that if variables
66556 are dependent on unsupported options, the results are unspecified.

66557 IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/137 is applied, removing
66558 _REGEX_VERSION and _SC_REGEX_VERSION.

66559 **Issue 7**

66560 Austin Group Interpretation 1003.1-2001 #160 is applied.

66561 SD5-XSH-ERN-166 is applied, changing “Maximum size” to “Initial size” for the “Maximum
66562 size of ...” entries in the table in the DESCRIPTION.

66563 The variables for the supported programming environments are updated to be V7 and the
66564 LEGACY variables are removed.

66565 The following constants are added:

66566 _POSIX_THREAD_ROBUST_PRIO_INHERIT
66567 _POSIX_THREAD_ROBUST_PRIO_PROTECT

66568 The _XOPEN_UUCP variable and its associated _SC_XOPEN_UUCP value is added to the table
66569 of system variables.

66570 **NAME**

66571 syslog — log a message

66572 **SYNOPSIS**

66573 XSI `#include <syslog.h>`

66574 `void syslog(int priority, const char *message, ... /* argument */);`

66575 **DESCRIPTION**

66576 Refer to *closelog()*.

66577 **NAME**

66578 system — issue a command

66579 **SYNOPSIS**

66580 #include <stdlib.h>

66581 int system(const char *command);

66582 **DESCRIPTION**

66583 CX The functionality described on this reference page is aligned with the ISO C standard. Any
 66584 conflict between the requirements described here and the ISO C standard is unintentional. This
 66585 volume of POSIX.1-2008 defers to the ISO C standard.

66586 If *command* is a null pointer, the *system()* function shall determine whether the host environment
 66587 has a command processor. If *command* is not a null pointer, the *system()* function shall pass the
 66588 string pointed to by *command* to that command processor to be executed in an implementation-
 66589 defined manner; this might then cause the program calling *system()* to behave in a non-
 66590 conforming manner or to terminate.

66591 CX The *system()* function shall behave as if a child process were created using *fork()*, and the child
 66592 process invoked the *sh* utility using *execl()* as follows:

66593 execl(<shell path>, "sh", "-c", command, (char *)0);

66594 where <shell path> is an unspecified pathname for the *sh* utility. It is unspecified whether the
 66595 handlers registered with *pthread_atfork()* are called as part of the creation of the child process.

66596 The *system()* function shall ignore the SIGINT and SIGQUIT signals, and shall block the
 66597 SIGCHLD signal, while waiting for the command to terminate. If this might cause the
 66598 application to miss a signal that would have killed it, then the application should examine the
 66599 return value from *system()* and take whatever action is appropriate to the application if the
 66600 command terminated due to receipt of a signal.

66601 The *system()* function shall not affect the termination status of any child of the calling processes
 66602 other than the process or processes it itself creates.

66603 The *system()* function shall not return until the child process has terminated.66604 The *system()* function need not be thread-safe.66605 **RETURN VALUE**

66606 If *command* is a null pointer, *system()* shall return non-zero to indicate that a command processor
 66607 CX is available, or zero if none is available. The *system()* function shall always return non-zero
 66608 when *command* is NULL.

66609 CX If *command* is not a null pointer, *system()* shall return the termination status of the command
 66610 language interpreter in the format specified by *waitpid()*. The termination status shall be as
 66611 defined for the *sh* utility; otherwise, the termination status is unspecified. If some error prevents
 66612 the command language interpreter from executing after the child process is created, the return
 66613 value from *system()* shall be as if the command language interpreter had terminated using
 66614 *exit(127)* or *_exit(127)*. If a child process cannot be created, or if the termination status for the
 66615 command language interpreter cannot be obtained, *system()* shall return -1 and set *errno* to
 66616 indicate the error.

66617 **ERRORS**66618 CX The *system()* function may set *errno* values as described by *fork()*.

66619 In addition, *system()* may fail if:

66620 CX [ECHILD] The status of the child process created by *system()* is no longer available.

66621 EXAMPLES

66622 None.

66623 APPLICATION USAGE

66624 If the return value of *system()* is not `-1`, its value can be decoded through the use of the macros
66625 described in `<sys/wait.h>`. For convenience, these macros are also provided in `<stdlib.h>`.

66626 Note that, while *system()* must ignore `SIGINT` and `SIGQUIT` and block `SIGCHLD` while waiting
66627 for the child to terminate, the handling of signals in the executed command is as specified by
66628 *fork()* and *exec*. For example, if `SIGINT` is being caught or is set to `SIG_DFL` when *system()* is
66629 called, then the child is started with `SIGINT` handling set to `SIG_DFL`.

66630 Ignoring `SIGINT` and `SIGQUIT` in the parent process prevents coordination problems (two
66631 processes reading from the same terminal, for example) when the executed command ignores or
66632 catches one of the signals. It is also usually the correct action when the user has given a
66633 command to the application to be executed synchronously (as in the `'!'` command in many
66634 interactive applications). In either case, the signal should be delivered only to the child process,
66635 not to the application itself. There is one situation where ignoring the signals might have less
66636 than the desired effect. This is when the application uses *system()* to perform some task invisible
66637 to the user. If the user typed the interrupt character ("`^C`", for example) while *system()* is being
66638 used in this way, one would expect the application to be killed, but only the executed command
66639 is killed. Applications that use *system()* in this way should carefully check the return status from
66640 *system()* to see if the executed command was successful, and should take appropriate action
66641 when the command fails.

66642 Blocking `SIGCHLD` while waiting for the child to terminate prevents the application from
66643 catching the signal and obtaining status from *system()*'s child process before *system()* can get the
66644 status itself.

66645 The context in which the utility is ultimately executed may differ from that in which *system()*
66646 was called. For example, file descriptors that have the `FD_CLOEXEC` flag set are closed, and the
66647 process ID and parent process ID are different. Also, if the executed utility changes its
66648 environment variables or its current working directory, that change is not reflected in the caller's
66649 context.

66650 There is no defined way for an application to find the specific path for the shell. However,
66651 *confstr()* can provide a value for *PATH* that is guaranteed to find the *sh* utility.

66652 Using the *system()* function in more than one thread in a process or when the `SIGCHLD` signal is
66653 being manipulated by more than one thread in a process may produce unexpected results.

66654 RATIONALE

66655 The *system()* function should not be used by programs that have set user (or group) ID
66656 privileges. The *fork()* and *exec* family of functions (except *execlp()* and *execvp()*), should be used
66657 instead. This prevents any unforeseen manipulation of the environment of the user that could
66658 cause execution of commands not anticipated by the calling program.

66659 There are three levels of specification for the *system()* function. The ISO C standard gives the
66660 most basic. It requires that the function exists, and defines a way for an application to query
66661 whether a command language interpreter exists. It says nothing about the command language
66662 or the environment in which the command is interpreted.

66663 POSIX.1-2008 places additional restrictions on *system()*. It requires that if there is a command
66664 language interpreter, the environment must be as specified by *fork()* and *exec*. This ensures, for

example, that *close-on-exec* works, that file locks are not inherited, and that the process ID is different. It also specifies the return value from *system()* when the command line can be run, thus giving the application some information about the command's completion status.

Finally, POSIX.1-2008 requires the command to be interpreted as in the shell command language defined in the Shell and Utilities volume of POSIX.1-2008.

Note that, *system(NULL)* is required to return non-zero, indicating that there is a command language interpreter. At first glance, this would seem to conflict with the ISO C standard which allows *system(NULL)* to return zero. There is no conflict, however. A system must have a command language interpreter, and is non-conforming if none is present. It is therefore permissible for the *system()* function on such a system to implement the behavior specified by the ISO C standard as long as it is understood that the implementation does not conform to POSIX.1-2008 if *system(NULL)* returns zero.

It was explicitly decided that when *command* is *NULL*, *system()* should not be required to check to make sure that the command language interpreter actually exists with the correct mode, that there are enough processes to execute it, and so on. The call *system(NULL)* could, theoretically, check for such problems as too many existing child processes, and return zero. However, it would be inappropriate to return zero due to such a (presumably) transient condition. If some condition exists that is not under the control of this application and that would cause any *system()* call to fail, that system has been rendered non-conforming.

Early drafts required, or allowed, *system()* to return with *errno* set to [EINTR] if it was interrupted with a signal. This error return was removed, and a requirement that *system()* not return until the child has terminated was added. This means that if a *waitpid()* call in *system()* exits with *errno* set to [EINTR], *system()* must reissue the *waitpid()*. This change was made for two reasons:

1. There is no way for an application to clean up if *system()* returns [EINTR], short of calling *wait()*, and that could have the undesirable effect of returning the status of children other than the one started by *system()*.
2. While it might require a change in some historical implementations, those implementations already have to be changed because they use *wait()* instead of *waitpid()*.

Note that if the application is catching SIGCHLD signals, it will receive such a signal before a successful *system()* call returns.

To conform to POSIX.1-2008, *system()* must use *waitpid()*, or some similar function, instead of *wait()*.

The following code sample illustrates how *system()* might be implemented on an implementation conforming to POSIX.1-2008.

```
#include <signal.h>
int system(const char *cmd)
{
    int stat;
    pid_t pid;
    struct sigaction sa, savintr, savequit;
    sigset_t saveblock;
    if (cmd == NULL)
        return(1);
    sa.sa_handler = SIG_IGN;
    sigemptyset(&sa.sa_mask);
    sa.sa_flags = 0;
```

```

66712     sigemptyset(&saveintr.sa_mask);
66713     sigemptyset(&savequit.sa_mask);
66714     sigaction(SIGINT, &sa, &saveintr);
66715     sigaction(SIGQUIT, &sa, &savequit);
66716     sigaddset(&sa.sa_mask, SIGCHLD);
66717     sigprocmask(SIG_BLOCK, &sa.sa_mask, &saveblock);
66718     if ((pid = fork()) == 0) {
66719         sigaction(SIGINT, &saveintr, (struct sigaction *)0);
66720         sigaction(SIGQUIT, &savequit, (struct sigaction *)0);
66721         sigprocmask(SIG_SETMASK, &saveblock, (sigset_t *)0);
66722         execl("/bin/sh", "sh", "-c", cmd, (char *)0);
66723         _exit(127);
66724     }
66725     if (pid == -1) {
66726         stat = -1; /* errno comes from fork() */
66727     } else {
66728         while (waitpid(pid, &stat, 0) == -1) {
66729             if (errno != EINTR) {
66730                 stat = -1;
66731                 break;
66732             }
66733         }
66734     }
66735     sigaction(SIGINT, &saveintr, (struct sigaction *)0);
66736     sigaction(SIGQUIT, &savequit, (struct sigaction *)0);
66737     sigprocmask(SIG_SETMASK, &saveblock, (sigset_t *)0);
66738     return(stat);
66739 }

```

Note that, while a particular implementation of *system()* (such as the one above) can assume a particular path for the shell, such a path is not necessarily valid on another system. The above example is not portable, and is not intended to be.

One reviewer suggested that an implementation of *system()* might want to use an environment variable such as *SHELL* to determine which command interpreter to use. The supposed implementation would use the default command interpreter if the one specified by the environment variable was not available. This would allow a user, when using an application that prompts for command lines to be processed using *system()*, to specify a different command interpreter. Such an implementation is discouraged. If the alternate command interpreter did not follow the command line syntax specified in the Shell and Utilities volume of POSIX.1-2008, then changing *SHELL* would render *system()* non-conforming. This would affect applications that expected the specified behavior from *system()*, and since the Shell and Utilities volume of POSIX.1-2008 does not mention that *SHELL* affects *system()*, the application would not know that it needed to unset *SHELL*.

FUTURE DIRECTIONS

None.

SEE ALSO

exec, *pipe()*, *pthread_atfork()*, *wait()*

XBD [<limits.h>](#), [<signal.h>](#), [<stdlib.h>](#), [<sys/wait.h>](#)

XCU *sh*

CHANGE HISTORY

66760 First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6

66762 Extensions beyond the ISO C standard are marked.

Issue 7

66764 Austin Group Interpretation 1003.1-2001 #055 is applied, clarifying the thread-safety of this function and treatment of *at_fork()* handlers.

66765 Austin Group Interpretation 1003.1-2001 #156 is applied.

66766 SD5-XSH-ERN-30 is applied.

66769 **NAME**

66770 tan, tanf, tanl — tangent function

66771 **SYNOPSIS**

```
66772 #include <math.h>
66773 double tan(double x);
66774 float tanf(float x);
66775 long double tanl(long double x);
```

66776 **DESCRIPTION**

66777 CX The functionality described on this reference page is aligned with the ISO C standard. Any
 66778 conflict between the requirements described here and the ISO C standard is unintentional. This
 66779 volume of POSIX.1-2008 defers to the ISO C standard.

66780 These functions shall compute the tangent of their argument x , measured in radians.

66781 An application wishing to check for error situations should set *errno* to zero and call
 66782 *feclearexcept*(FE_ALL_EXCEPT) before calling these functions. On return, if *errno* is non-zero or
 66783 *fetestexcept*(FE_INVALID | FE_DIVBYZERO | FE_OVERFLOW | FE_UNDERFLOW) is non-
 66784 zero, an error has occurred.

66785 **RETURN VALUE**

66786 Upon successful completion, these functions shall return the tangent of x .

66787 MXX If the correct value would cause underflow, and is not representable, a range error may occur, |
 66788 MXX and *tan()*, *tanf()*, and *tanl()* shall return 0.0, or (if IEC 60559 Floating-Point is not supported) an |
 66789 implementation-defined value no greater in magnitude than DBL_MIN, FLT_MIN, and |
 66790 LDBL_MIN, respectively.

66791 MX If x is NaN, a NaN shall be returned.

66792 If x is ± 0 , x shall be returned.

66793 If x is subnormal, a range error may occur

66794 MXX and x should be returned.

66795 MX If x is not returned, *tan()*, *tanf()*, and *tanl()* shall return an implementation-defined value no +
 66796 greater in magnitude than DBL_MIN, FLT_MIN, and LDBL_MIN, respectively.

66797 If x is $\pm\text{Inf}$, a domain error shall occur, and either a NaN (if supported), or an implementation-
 66798 defined value shall be returned.

66799 MXX If the correct value would cause underflow, and is representable, a range error may occur and |
 66800 the correct value shall be returned.

66801 XSI If the correct value would cause overflow, a range error shall occur and *tan()*, *tanf()*, and *tanl()*
 66802 shall return $\pm\text{HUGE\_VAL}$, $\pm\text{HUGE\_VALF}$, and $\pm\text{HUGE\_VALL}$, respectively, with the same sign
 66803 as the correct value of the function.

66804 **ERRORS**

66805 These functions shall fail if:

66806 MX **Domain Error** The value of x is $\pm\text{Inf}$.

66807 If the integer expression (*math_errhandling* & MATH_ERRNO) is non-zero,
 66808 then *errno* shall be set to [EDOM]. If the integer expression (*math_errhandling*
 66809 & MATH_ERREXCEPT) is non-zero, then the invalid floating-point exception
 66810 shall be raised.

66811 XSI **Range Error** The result overflows

66812 If the integer expression (*math_errhandling* & MATH_ERRNO) is non-zero,

66813 then *errno* shall be set to [ERANGE]. If the integer expression

66814 (*math_errhandling* & MATH_ERREXCEPT) is non-zero, then the overflow

66815 floating-point exception shall be raised.

66816 These functions may fail if:

66817 MX **Range Error** The result underflows, or the value of *x* is subnormal.

66818 If the integer expression (*math_errhandling* & MATH_ERRNO) is non-zero,

66819 then *errno* shall be set to [ERANGE]. If the integer expression

66820 (*math_errhandling* & MATH_ERREXCEPT) is non-zero, then the underflow

66821 floating-point exception shall be raised.

66822 EXAMPLES

66823 Taking the Tangent of a 45-Degree Angle

```
66824 #include <math.h>
66825 ...
66826 double radians = 45.0 * M_PI / 180;
66827 double result;
66828 ...
66829 result = tan (radians);
```

66830 APPLICATION USAGE

66831 There are no known floating-point representations such that for a normal argument, *tan(x)* is

66832 either overflow or underflow.

66833 These functions may lose accuracy when their argument is near a multiple of $\pi/2$ or is far from

66834 0.0.

66835 On error, the expressions (*math_errhandling* & MATH_ERRNO) and (*math_errhandling* &

66836 MATH_ERREXCEPT) are independent of each other, but at least one of them must be non-zero.

66837 RATIONALE

66838 None.

66839 FUTURE DIRECTIONS

66840 None.

66841 SEE ALSO

66842 [*atan\(\)*](#), [*feclearexcept\(\)*](#), [*fetestexcept\(\)*](#), [*isnan\(\)*](#)

66843 XBD [Section 4.19](#) (on page 116), [**<math.h>**](#)

66844 CHANGE HISTORY

66845 First released in Issue 1. Derived from Issue 1 of the SVID.

66846 Issue 5

66847 The last two paragraphs of the DESCRIPTION were included as APPLICATION USAGE notes

66848 in previous issues.

66849 Issue 6

66850 The *tanf()* and *tanl()* functions are added for alignment with the ISO/IEC 9899:1999 standard.

66851 The DESCRIPTION, RETURN VALUE, ERRORS, and APPLICATION USAGE sections are

66852 revised to align with the ISO/IEC 9899:1999 standard.

66853 IEC 60559:1989 standard floating-point extensions over the ISO/IEC 9899:1999 standard are
66854 marked.

66855 IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/64 is applied, correcting the last
66856 paragraph in the RETURN VALUE section.

66857 **Issue 7**

66858 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0635 [68], XSH/TC1-2008/0636 [68], +
66859 and XSH/TC1-2008/0637 [68] are applied.

66860 **NAME**

66861 tanh, tanhf, tanhl — hyperbolic tangent functions

66862 **SYNOPSIS**

```
66863       #include <math.h>
66864       double tanh(double x);
66865       float tanhf(float x);
66866       long double tanhl(long double x);
```

66867 **DESCRIPTION**

66868 CX The functionality described on this reference page is aligned with the ISO C standard. Any
 66869 conflict between the requirements described here and the ISO C standard is unintentional. This
 66870 volume of POSIX.1-2008 defers to the ISO C standard.

66871 These functions shall compute the hyperbolic tangent of their argument *x*.

66872 An application wishing to check for error situations should set *errno* to zero and call
 66873 *feclearexcept*(FE_ALL_EXCEPT) before calling these functions. On return, if *errno* is non-zero or
 66874 *fetestexcept*(FE_INVALID | FE_DIVBYZERO | FE_OVERFLOW | FE_UNDERFLOW) is non-
 66875 zero, an error has occurred.

66876 **RETURN VALUE**

66877 Upon successful completion, these functions shall return the hyperbolic tangent of *x*.

66878 MX If *x* is NaN, a NaN shall be returned.

66879 If *x* is ± 0 , *x* shall be returned.

66880 If *x* is $\pm \text{Inf}$, ± 1 shall be returned.

66881 If *x* is subnormal, a range error may occur
 66882 MXX and *x* should be returned.

66883 MX If *x* is not returned, *tanh()*, *tanhf()*, and *tanhl()* shall return an implementation-defined value no
 66884 greater in magnitude than DBL_MIN, FLT_MIN, and LDBL_MIN, respectively.

66885 **ERRORS**

66886 These functions may fail if:

66887 MX Range Error The value of *x* is subnormal.

66888 If the integer expression (*math_errhandling* & MATH_ERRNO) is non-zero,
 66889 then *errno* shall be set to [ERANGE]. If the integer expression
 66890 (*math_errhandling* & MATH_ERREXCEPT) is non-zero, then the underflow
 66891 floating-point exception shall be raised.

66892 **EXAMPLES**

66893 None.

66894 **APPLICATION USAGE**

66895 On error, the expressions (*math_errhandling* & MATH_ERRNO) and (*math_errhandling* &
 66896 MATH_ERREXCEPT) are independent of each other, but at least one of them must be non-zero.

66897 **RATIONALE**

66898 None.

66899 **FUTURE DIRECTIONS**

66900 None.

66901 **SEE ALSO**66902 *atanh()*, *feclearexcept()*, *fetestexcept()*, *isnan()*, *tan()*66903 XBD Section 4.19 (on page 116), **<math.h>**66904 **CHANGE HISTORY**

66905 First released in Issue 1. Derived from Issue 1 of the SVID.

66906 **Issue 5**66907 The DESCRIPTION is updated to indicate how an application should check for an error. This
66908 text was previously published in the APPLICATION USAGE section.66909 **Issue 6**66910 The *tanhf()* and *tanhhl()* functions are added for alignment with the ISO/IEC 9899:1999 standard.66911 The DESCRIPTION, RETURN VALUE, ERRORS, and APPLICATION USAGE sections are
66912 revised to align with the ISO/IEC 9899:1999 standard.66913 IEC 60559:1989 standard floating-point extensions over the ISO/IEC 9899:1999 standard are
66914 marked.66915 **Issue 7**

66916 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0638 [68] is applied.

+

66917 **NAME**

66918 tanl — tangent function

66919 **SYNOPSIS**

66920 #include <math.h>

66921 long double tanl(long double *x*);66922 **DESCRIPTION**66923 Refer to *tan()*.

66924 **NAME**

66925 tcdrain — wait for transmission of output

66926 **SYNOPSIS**

66927 #include <termios.h>

66928 int tcdrain(int *fildes*);66929 **DESCRIPTION**

66930 The *tcdrain()* function shall block until all output written to the object referred to by *fildes* is
 66931 transmitted. The *fildes* argument is an open file descriptor associated with a terminal.

66932 Any attempts to use *tcdrain()* from a process which is a member of a background process group
 66933 on a *fildes* associated with its controlling terminal, shall cause the process group to be sent a
 66934 SIGTTOU signal. If the calling thread is blocking SIGTTOU signals or the process is ignoring
 66935 SIGTTOU signals, the process shall be allowed to perform the operation, and no signal is sent.

66936 **RETURN VALUE**

66937 Upon successful completion, 0 shall be returned. Otherwise, -1 shall be returned and *errno* set to
 66938 indicate the error.

66939 **ERRORS**66940 The *tcdrain()* function shall fail if:66941 [EBADF] The *fildes* argument is not a valid file descriptor.66942 [EINTR] A signal interrupted *tcdrain()*.

66943 [EIO] The process group of the writing process is orphaned, the calling thread is not
 66944 blocking SIGTTOU, and the process is not ignoring SIGTTOU.

66945 [ENOTTY] The file associated with *fildes* is not a terminal.66946 **EXAMPLES**

66947 None.

66948 **APPLICATION USAGE**

66949 None.

66950 **RATIONALE**

66951 None.

66952 **FUTURE DIRECTIONS**

66953 None.

66954 **SEE ALSO**66955 *tcflush()*

66956 XBD Chapter 11 (on page 199), <termios.h>

66957 **CHANGE HISTORY**

66958 First released in Issue 3. Included for alignment with the POSIX.1-1988 standard.

Issue 6

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- In the DESCRIPTION, the final paragraph is no longer conditional on _POSIX_JOB_CONTROL. This is a FIPS requirement.
- The [EIO] error is added.

Issue 7

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0639 [79], XSH/TC1-2008/0640 [79], + and XSH/TC1-2008/0641 [79] are applied.

NAME

tcflow — suspend or restart the transmission or reception of data

SYNOPSIS

```
#include <termios.h>
```

```
int tcflow(int fildes, int action);
```

DESCRIPTION

The *tcflow()* function shall suspend or restart transmission or reception of data on the object referred to by *fildes*, depending on the value of *action*. The *fildes* argument is an open file descriptor associated with a terminal.

- If *action* is TCOOFF, output shall be suspended.
- If *action* is TCOON, suspended output shall be restarted.
- If *action* is TCIOFF and *fildes* refers to a terminal device, the system shall transmit a STOP character, which is intended to cause the terminal device to stop transmitting data to the system. If *fildes* is associated with a pseudo-terminal, the STOP character need not be transmitted.
- If *action* is TCION and *fildes* refers to a terminal device, the system shall transmit a START character, which is intended to cause the terminal device to start transmitting data to the system. If *fildes* is associated with a pseudo-terminal, the START character need not be transmitted.

The default on the opening of a terminal file is that neither its input nor its output are suspended.

Attempts to use *tcflow()* from a process which is a member of a background process group on a *fildes* associated with its controlling terminal, shall cause the process group to be sent a SIGTTOU signal. If the calling thread is blocking SIGTTOU signals or the process is ignoring SIGTTOU signals, the process shall be allowed to perform the operation, and no signal is sent.

RETURN VALUE

Upon successful completion, 0 shall be returned. Otherwise, -1 shall be returned and *errno* set to indicate the error.

ERRORS

The *tcflow()* function shall fail if:

[EBADF]	The <i>fildes</i> argument is not a valid file descriptor.
[EINVAL]	The <i>action</i> argument is not a supported value.
[EIO]	The process group of the writing process is orphaned, the calling thread is not blocking SIGTTOU, and the process is not ignoring SIGTTOU.
[ENOTTY]	The file associated with <i>fildes</i> is not a terminal.

67003 **EXAMPLES**

67004 None.

67005 **APPLICATION USAGE**

67006 None.

67007 **RATIONALE**

67008 None.

67009 **FUTURE DIRECTIONS**

67010 None.

67011 **SEE ALSO**67012 *tcsendbreak()*67013 XBD [Chapter 11](#) (on page 199), [<termios.h>](#)67014 **CHANGE HISTORY**

67015 First released in Issue 3. Included for alignment with the POSIX.1-1988 standard.

67016 **Issue 6**67017 The following new requirements on POSIX implementations derive from alignment with the
67018 Single UNIX Specification:

- 67019
- The [EIO] error is added.

67020 **Issue 7**67021 SD5-XSH-ERN-190 is applied, clarifying in the DESCRIPTION the transmission of START and
67022 STOP characters.67023 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0642 [79], XSH/TC1-2008/0643 [79], +
67024 and XSH/TC1-2008/0644 [79] are applied.

NAME

tcflush — flush non-transmitted output data, non-read input data, or both

SYNOPSIS

```
#include <termios.h>
```

```
int tcflush(int fildes, int queue_selector);
```

DESCRIPTION

Upon successful completion, *tcflush()* shall discard data written to the object referred to by *fildes* (an open file descriptor associated with a terminal) but not transmitted, or data received but not read, depending on the value of *queue_selector*:

- If *queue_selector* is TCIFLUSH, it shall flush data received but not read.
- If *queue_selector* is TCOFLUSH, it shall flush data written but not transmitted.
- If *queue_selector* is TCIOFLUSH, it shall flush both data received but not read and data written but not transmitted.

Attempts to use *tcflush()* from a process which is a member of a background process group on a *fildes* associated with its controlling terminal shall cause the process group to be sent a SIGTTOU signal. If the calling thread is blocking SIGTTOU signals or the process is ignoring SIGTTOU signals, the process shall be allowed to perform the operation, and no signal is sent.

RETURN VALUE

Upon successful completion, 0 shall be returned. Otherwise, -1 shall be returned and *errno* set to indicate the error.

ERRORS

The *tcflush()* function shall fail if:

[EBADF]	The <i>fildes</i> argument is not a valid file descriptor.
[EINVAL]	The <i>queue_selector</i> argument is not a supported value.
[EIO]	The process group of the writing process is orphaned, the calling thread is not blocking SIGTTOU, and the process is not ignoring SIGTTOU.
[ENOTTY]	The file associated with <i>fildes</i> is not a terminal.

EXAMPLES

None.

APPLICATION USAGE

None.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

[*tcdrain\(\)*](#)

XBD [Chapter 11](#) (on page 199), [*<termios.h>*](#)

CHANGE HISTORY

First released in Issue 3. Included for alignment with the POSIX.1-1988 standard.

Issue 6

The Open Group Corrigendum U035/1 is applied. In the ERRORS and APPLICATION USAGE sections, references to *tcflow()* are replaced with *tcflush()*.

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- In the DESCRIPTION, the final paragraph is no longer conditional on _POSIX_JOB_CONTROL. This is a FIPS requirement.
- The [EIO] error is added.

Issue 7

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0645 [79], XSH/TC1-2008/0646 [79], + and XSH/TC1-2008/0647 [79] are applied.

NAME

tcgetattr — get the parameters associated with the terminal

SYNOPSIS

```
#include <termios.h>
```

```
int tcgetattr(int fildes, struct termios *termios_p);
```

DESCRIPTION

The *tcgetattr()* function shall get the parameters associated with the terminal referred to by *fildes* and store them in the **termios** structure referenced by *termios_p*. The *fildes* argument is an open file descriptor associated with a terminal.

The *termios_p* argument is a pointer to a **termios** structure.

The *tcgetattr()* operation is allowed from any process.

If the terminal device supports different input and output baud rates, the baud rates stored in the **termios** structure returned by *tcgetattr()* shall reflect the actual baud rates, even if they are equal. If differing baud rates are not supported, the rate returned as the output baud rate shall be the actual baud rate. If the terminal device does not support split baud rates, the input baud rate stored in the **termios** structure shall be the output rate (as one of the symbolic values).

RETURN VALUE

Upon successful completion, 0 shall be returned. Otherwise, -1 shall be returned and *errno* set to indicate the error.

ERRORS

The *tcgetattr()* function shall fail if:

- [EBADF] The *fildes* argument is not a valid file descriptor.
- [ENOTTY] The file associated with *fildes* is not a terminal.

EXAMPLES

None.

APPLICATION USAGE

None.

RATIONALE

Care must be taken when changing the terminal attributes. Applications should always do a *tcgetattr()*, save the **termios** structure values returned, and then do a *tcsetattr()*, changing only the necessary fields. The application should use the values saved from the *tcgetattr()* to reset the terminal state whenever it is done with the terminal. This is necessary because terminal attributes apply to the underlying port and not to each individual open instance; that is, all processes that have used the terminal see the latest attribute changes.

A program that uses these functions should be written to catch all signals and take other appropriate actions to ensure that when the program terminates, whether planned or not, the terminal device's state is restored to its original state.

Existing practice dealing with error returns when only part of a request can be honored is based on calls to the *ioctl()* function. In historical BSD and System V implementations, the corresponding *ioctl()* returns zero if the requested actions were semantically correct, even if some of the requested changes could not be made. Many existing applications assume this behavior and would no longer work correctly if the return value were changed from zero to -1 in this case.

Note that either specification has a problem. When zero is returned, it implies everything

67120 succeeded even if some of the changes were not made. When -1 is returned, it implies
67121 everything failed even though some of the changes were made.

67122 Applications that need all of the requested changes made to work properly should follow
67123 *tcsetattr()* with a call to *tcgetattr()* and compare the appropriate field values.

67124 **FUTURE DIRECTIONS**

67125 None.

67126 **SEE ALSO**

67127 *tcsetattr()*

67128 XBD [Chapter 11](#) (on page 199), [<termios.h>](#)

67129 **CHANGE HISTORY**

67130 First released in Issue 3. Included for alignment with the POSIX.1-1988 standard.

67131 **Issue 6**

67132 In the DESCRIPTION, the rate returned as the input baud rate shall be the output rate.
67133 Previously, the number zero was also allowed but was obsolescent.

NAME

tcgetpgrp — get the foreground process group ID

SYNOPSIS

```
#include <unistd.h>
```

```
pid_t tcgetpgrp(int fildes);
```

DESCRIPTION

The *tcgetpgrp()* function shall return the value of the process group ID of the foreground process group associated with the terminal.

If there is no foreground process group, *tcgetpgrp()* shall return a value greater than 1 that does not match the process group ID of any existing process group.

The *tcgetpgrp()* function is allowed from a process that is a member of a background process group; however, the information may be subsequently changed by a process that is a member of a foreground process group.

RETURN VALUE

Upon successful completion, *tcgetpgrp()* shall return the value of the process group ID of the foreground process associated with the terminal. Otherwise, *-1* shall be returned and *errno* set to indicate the error.

ERRORS

The *tcgetpgrp()* function shall fail if:

[EBADF]	The <i>fildes</i> argument is not a valid file descriptor.
[ENOTTY]	The calling process does not have a controlling terminal, or the file is not the controlling terminal.

EXAMPLES

None.

APPLICATION USAGE

None.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

setsid(), *setpgid()*, *tcsetpgrp()*

XBD [<sys/types.h>](#), [<unistd.h>](#)

CHANGE HISTORY

First released in Issue 3. Included for alignment with the POSIX.1-1988 standard.

Issue 6

In the SYNOPSIS, the optional include of the [<sys/types.h>](#) header is removed.

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The requirement to include [<sys/types.h>](#) has been removed. Although [<sys/types.h>](#) was required for conforming implementations of previous POSIX specifications, it was not required for UNIX applications.

67176
67177

- In the DESCRIPTION, text previously conditional on support for _POSIX_JOB_CONTROL is now mandatory. This is a FIPS requirement.

67178 **NAME**

67179 tcgetsid — get the process group ID for the session leader for the controlling terminal

67180 **SYNOPSIS**

67181 #include <termios.h>

67182 pid_t tcgetsid(int *fildes*);67183 **DESCRIPTION**67184 The *tcgetsid()* function shall obtain the process group ID of the session for which the terminal
67185 specified by *fildes* is the controlling terminal.67186 **RETURN VALUE**67187 Upon successful completion, *tcgetsid()* shall return the process group ID of the session
67188 associated with the terminal. Otherwise, a value of -1 shall be returned and *errno* set to indicate
67189 the error.67190 **ERRORS**67191 The *tcgetsid()* function shall fail if:67192 [EBADF] The *fildes* argument is not a valid file descriptor.67193 [ENOTTY] The calling process does not have a controlling terminal, or the file is not the
67194 controlling terminal.67195 **EXAMPLES**

67196 None.

67197 **APPLICATION USAGE**

67198 None.

67199 **RATIONALE**

67200 None.

67201 **FUTURE DIRECTIONS**

67202 None.

67203 **SEE ALSO**67204 XBD [<termios.h>](#)67205 **CHANGE HISTORY**

67206 First released in Issue 4, Version 2.

67207 **Issue 5**

67208 Moved from X/OPEN UNIX extension to BASE.

67209 The [EACCES] error has been removed from the list of mandatory errors, and the description of
67210 [ENOTTY] has been reworded.67211 **Issue 7**

67212 SD5-XSH-ERN-180 is applied, clarifying the RETURN VALUE section.

67213 The *tcgetsid()* function is moved from the XSI option to the Base.

67214 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0648 [421] is applied.

+

67215 **NAME**

67216 tcsendbreak — send a break for a specific duration

67217 **SYNOPSIS**

67218 #include <termios.h>

67219 int tcsendbreak(int *fildes*, int *duration*);67220 **DESCRIPTION**

67221 If the terminal is using asynchronous serial data transmission, *tcsendbreak()* shall cause
 67222 transmission of a continuous stream of zero-valued bits for a specific duration. If *duration* is 0, it
 67223 shall cause transmission of zero-valued bits for at least 0.25 seconds, and not more than 0.5
 67224 seconds. If *duration* is not 0, it shall send zero-valued bits for an implementation-defined period
 67225 of time.

67226 The *fildes* argument is an open file descriptor associated with a terminal.

67227 If the terminal is not using asynchronous serial data transmission, it is implementation-defined
 67228 whether *tcsendbreak()* sends data to generate a break condition or returns without taking any
 67229 action.

67230 Attempts to use *tcsendbreak()* from a process which is a member of a background process group
 67231 on a *fildes* associated with its controlling terminal shall cause the process group to be sent a
 67232 SIGTTOU signal. If the calling thread is blocking SIGTTOU signals or the process is ignoring
 67233 SIGTTOU signals, the process shall be allowed to perform the operation, and no signal is sent.

67234 **RETURN VALUE**

67235 Upon successful completion, 0 shall be returned. Otherwise, -1 shall be returned and *errno* set to
 67236 indicate the error.

67237 **ERRORS**

67238 The *tcsendbreak()* function shall fail if:

67239 [EBADF] The *fildes* argument is not a valid file descriptor.

67240 [EIO] The process group of the writing process is orphaned, the calling thread is not
 67241 blocking SIGTTOU, and the process is not ignoring SIGTTOU.

67242 [ENOTTY] The file associated with *fildes* is not a terminal.

67243 **EXAMPLES**

67244 None.

67245 **APPLICATION USAGE**

67246 None.

67247 **RATIONALE**

67248 None.

67249 **FUTURE DIRECTIONS**

67250 None.

67251 **SEE ALSO**

67252 XBD [Chapter 11](#) (on page 199), [<termios.h>](#)

67253 **CHANGE HISTORY**

67254 First released in Issue 3. Included for alignment with the POSIX.1-1988 standard.

Issue 6

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- In the DESCRIPTION, text previously conditional on _POSIX_JOB_CONTROL is now mandated. This is a FIPS requirement.
- The [EIO] error is added.

Issue 7

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0649 [79], XSH/TC1-2008/0650 [79], + and XSH/TC1-2008/0651 [79] are applied.

NAME

tcsetattr — set the parameters associated with the terminal

SYNOPSIS

```
#include <termios.h>
```

```
int tcsetattr(int fildes, int optional_actions,
              const struct termios *termios_p);
```

DESCRIPTION

The *tcsetattr()* function shall set the parameters associated with the terminal referred to by the open file descriptor *fildes* (an open file descriptor associated with a terminal) from the **termios** structure referenced by *termios_p* as follows:

- If *optional_actions* is TCSANOW, the change shall occur immediately.
- If *optional_actions* is TCSADRAIN, the change shall occur after all output written to *fildes* is transmitted. This function should be used when changing parameters that affect output.
- If *optional_actions* is TCSAFLUSH, the change shall occur after all output written to *fildes* is transmitted, and all input so far received but not read shall be discarded before the change is made.

If the output baud rate stored in the **termios** structure pointed to by *termios_p* is the zero baud rate, B0, the modem control lines shall no longer be asserted. Normally, this shall disconnect the line.

If the input baud rate stored in the **termios** structure pointed to by *termios_p* is 0, the input baud rate given to the hardware is the same as the output baud rate stored in the **termios** structure.

The *tcsetattr()* function shall return successfully if it was able to perform any of the requested actions, even if some of the requested actions could not be performed. It shall set all the attributes that the implementation supports as requested and leave all the attributes not supported by the implementation unchanged. If no part of the request can be honored, it shall return -1 and set *errno* to [EINVAL]. If the input and output baud rates differ and are a combination that is not supported, neither baud rate shall be changed. A subsequent call to *tcgetattr()* shall return the actual state of the terminal device (reflecting both the changes made and not made in the previous *tcsetattr()* call). The *tcsetattr()* function shall not change the values found in the **termios** structure under any circumstances.

The effect of *tcsetattr()* is undefined if the value of the **termios** structure pointed to by *termios_p* was not derived from the result of a call to *tcgetattr()* on *fildes*; an application should modify only fields and flags defined by this volume of POSIX.1-2008 between the call to *tcgetattr()* and *tcsetattr()*, leaving all other fields and flags unmodified.

No actions defined by this volume of POSIX.1-2008, other than a call to *tcsetattr()*, a close of the last file descriptor in the system associated with this terminal device, or an open of the first file descriptor in the system associated with this terminal device (using the O_TTY_INIT flag if it is non-zero and the device is not a pseudo-terminal), shall cause any of the terminal attributes defined by this volume of POSIX.1-2008 to change.

If *tcsetattr()* is called from a process which is a member of a background process group on a *fildes* associated with its controlling terminal:

- If the calling thread is blocking SIGTTOU signals or the process is ignoring SIGTTOU signals, the operation completes normally and no signal is sent.

- Otherwise, a SIGTTOU signal shall be sent to the process group.

RETURN VALUE

Upon successful completion, 0 shall be returned. Otherwise, -1 shall be returned and *errno* set to indicate the error.

ERRORS

The *tcsetattr()* function shall fail if:

[EBADF]	The <i>fildev</i> argument is not a valid file descriptor.
[EINTR]	A signal interrupted <i>tcsetattr()</i> .
[EINVAL]	The <i>optional_actions</i> argument is not a supported value, or an attempt was made to change an attribute represented in the termios structure to an unsupported value.
[EIO]	The process group of the writing process is orphaned, the calling thread is not blocking SIGTTOU, and the process is not ignoring SIGTTOU.
[ENOTTY]	The file associated with <i>fildev</i> is not a terminal.

EXAMPLES

None.

APPLICATION USAGE

If trying to change baud rates, applications should call *tcsetattr()* then call *tcgetattr()* in order to determine what baud rates were actually selected.

In general, there are two reasons for an application to change the parameters associated with a terminal device:

1. The device already has working parameter settings but the application needs a different behavior, such as non-canonical mode instead of canonical mode. The application sets (or clears) only a few flags or *c_cc[]* values. Typically, the terminal device in this case is either the controlling terminal for the process or a pseudo-terminal.
2. The device is a modem or similar piece of equipment connected by a serial line, and it was not open before the application opened it. In this case, the application needs to initialize all of the parameter settings “from scratch”. However, since the **termios** structure may include both standard and non-standard parameters, the application cannot just initialize the whole structure in an arbitrary way (e.g., using *memset()*) as this may cause some of the non-standard parameters to be set incorrectly, resulting in non-conforming behavior of the terminal device. Conversely, the application cannot just set the standard parameters, assuming that the non-standard parameters will already have suitable values, as the device might previously have been used with non-conforming parameter settings (and some implementations retain the settings from one use to the next). The solution is to open the terminal device using the O_TTY_INIT flag to initialize the terminal device to have conforming parameter settings, obtain those settings using *tcgetattr()*, and then set all of the standard parameters to the desired settings.

RATIONALE

The *tcsetattr()* function can be interrupted in the following situations:

- It is interrupted while waiting for output to drain.
- It is called from a process in a background process group and SIGTTOU is caught.

See also the RATIONALE section in *tcgetattr()*.

FUTURE DIRECTIONS

Using an input baud rate of 0 to set the input rate equal to the output rate may not necessarily be supported in a future version of this volume of POSIX.1-2008.

SEE ALSO

cfgetispeed(), *tcgetattr()*

XBD [Chapter 11](#) (on page 199), [<termios.h>](#)

CHANGE HISTORY

First released in Issue 3. Included for alignment with the POSIX.1-1988 standard.

Issue 6

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- In the DESCRIPTION, text previously conditional on `_POSIX_JOB_CONTROL` is now mandated. This is a FIPS requirement.
- The [EIO] error is added.

In the DESCRIPTION, the text describing use of *tcsetattr()* from a process which is a member of a background process group is clarified.

Issue 7

Austin Group Interpretation 1003.1-2001 #144 is applied.

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0652 [79], XSH/TC1-2008/0653 [79], + and XSH/TC1-2008/0654 [79] are applied.

67370 **NAME**

67371 tcsetpgrp — set the foreground process group ID

67372 **SYNOPSIS**

67373 #include <unistd.h>

67374 int tcsetpgrp(int *fildes*, pid_t *pgid_id*);67375 **DESCRIPTION**

67376 If the process has a controlling terminal, *tcsetpgrp()* shall set the foreground process group ID
 67377 associated with the terminal to *pgid_id*. The application shall ensure that the file associated with
 67378 *fildes* is the controlling terminal of the calling process and the controlling terminal is currently
 67379 associated with the session of the calling process. The application shall ensure that the value of
 67380 *pgid_id* matches a process group ID of a process in the same session as the calling process.

67381 Attempts to use *tcsetpgrp()* from a process which is a member of a background process group on
 67382 a *fildes* associated with its controlling terminal shall cause the process group to be sent a
 67383 SIGTTOU signal. If the calling thread is blocking SIGTTOU signals or the process is ignoring
 67384 SIGTTOU signals, the process shall be allowed to perform the operation, and no signal is sent.

67385 **RETURN VALUE**

67386 Upon successful completion, 0 shall be returned. Otherwise, -1 shall be returned and *errno* set to
 67387 indicate the error.

67388 **ERRORS**67389 The *tcsetpgrp()* function shall fail if:67390 [EBADF] The *fildes* argument is not a valid file descriptor.67391 [EINVAL] This implementation does not support the value in the *pgid_id* argument.

67392 [EIO] The process group of the writing process is orphaned, the calling thread is not
 67393 blocking SIGTTOU, and the process is not ignoring SIGTTOU.

67394 [ENOTTY] The calling process does not have a controlling terminal, or the file is not the
 67395 controlling terminal, or the controlling terminal is no longer associated with
 67396 the session of the calling process.

67397 [EPERM] The value of *pgid_id* is a value supported by the implementation, but does not
 67398 match the process group ID of a process in the same session as the calling
 67399 process.

67400 **EXAMPLES**

67401 None.

67402 **APPLICATION USAGE**

67403 None.

67404 **RATIONALE**

67405 None.

67406 **FUTURE DIRECTIONS**

67407 None.

67408 **SEE ALSO**67409 *tcgetpgrp()*

67410 XBD <sys/types.h>, <unistd.h>

CHANGE HISTORY

67411 First released in Issue 3. Included for alignment with the POSIX.1-1988 standard.

Issue 6

67413 In the SYNOPSIS, the inclusion of `<sys/types.h>` is no longer required.

67415 The following new requirements on POSIX implementations derive from alignment with the
67416 Single UNIX Specification:

67417 • The requirement to include `<sys/types.h>` has been removed. Although `<sys/types.h>` was
67418 required for conforming implementations of previous POSIX specifications, it was not
67419 required for UNIX applications.

67420 • In the DESCRIPTION and ERRORS sections, text previously conditional on
67421 `_POSIX_JOB_CONTROL` is now mandated. This is a FIPS requirement.

67422 The normative text is updated to avoid use of the term “must” for application requirements.

67423 The Open Group Corrigendum U047/4 is applied.

Issue 7

67424 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0655 [79] and XSH/TC1-2008/0656 +
67425 [79] are applied.
67426

NAME

tdelete, tfind, tsearch, twalk — manage a binary search tree

SYNOPSIS

```
#include <search.h>

void *tdelete(const void *restrict key, void **restrict rootp,
              int (*compar)(const void *, const void *));
void *tfind(const void *key, void *const *rootp,
            int (*compar)(const void *, const void *));
void *tsearch(const void *key, void **rootp,
              int (*compar)(const void *, const void *));
void twalk(const void *root,
           void (*action)(const void *, VISIT, int));
```

DESCRIPTION

The *tdelete()*, *tfind()*, *tsearch()*, and *twalk()* functions manipulate binary search trees. Comparisons are made with a user-supplied routine, the address of which is passed as the *compar* argument. This routine is called with two arguments, which are the pointers to the elements being compared. The application shall ensure that the user-supplied routine returns an integer less than, equal to, or greater than 0, according to whether the first argument is to be considered less than, equal to, or greater than the second argument. The comparison function need not compare every byte, so arbitrary data may be contained in the elements in addition to the values being compared.

The *tsearch()* function shall build and access the tree. The *key* argument is a pointer to an element to be accessed or stored. If there is a node in the tree whose element is equal to the value pointed to by *key*, a pointer to this found node shall be returned. Otherwise, the value pointed to by *key* shall be inserted (that is, a new node is created and the value of *key* is copied to this node), and a pointer to this node returned. Only pointers are copied, so the application shall ensure that the calling routine stores the data. The *rootp* argument points to a variable that points to the root node of the tree. A null pointer value for the variable pointed to by *rootp* denotes an empty tree; in this case, the variable shall be set to point to the node which shall be at the root of the new tree.

Like *tsearch()*, *tfind()* shall search for a node in the tree, returning a pointer to it if found. However, if it is not found, *tfind()* shall return a null pointer. The arguments for *tfind()* are the same as for *tsearch()*.

The *tdelete()* function shall delete a node from a binary search tree. The arguments are the same as for *tsearch()*. The variable pointed to by *rootp* shall be changed if the deleted node was the root of the tree. The *tdelete()* function shall return a pointer to the parent of the deleted node, or an unspecified non-null pointer if the deleted node was the root node, or a null pointer if the node is not found.

If *tsearch()* adds an element to a tree, or *tdelete()* successfully deletes an element from a tree, the concurrent use of that tree in another thread, or use of pointers produced by a previous call to *tfind()* or *tsearch()*, produces undefined results.

The *twalk()* function shall traverse a binary search tree. The *root* argument is a pointer to the root node of the tree to be traversed. (Any node in a tree may be used as the root for a walk below that node.) The argument *action* is the name of a routine to be invoked at each node. This routine is, in turn, called with three arguments. The first argument shall be the address of the node being visited. The structure pointed to by this argument is unspecified and shall not be modified by the application, but it shall be possible to cast a pointer-to-node into a pointer-to-pointer-to-

element to access the element stored in the node. The second argument shall be a value from an enumeration data type:

```
typedef enum { preorder, postorder, endorder, leaf } VISIT;
```

(defined in `<search.h>`), depending on whether this is the first, second, or third time that the node is visited (during a depth-first, left-to-right traversal of the tree), or whether the node is a leaf. The third argument shall be the level of the node in the tree, with the root being level 0.

If the calling function alters the pointer to the root, the result is undefined.

If the functions pointed to by *action* or *compar* (for any of these binary search functions) change the tree, the results are undefined.

These functions are thread-safe only as long as multiple threads do not access the same tree.

RETURN VALUE

If the node is found, both *tsearch()* and *tfind()* shall return a pointer to it. If not, *tfind()* shall return a null pointer, and *tsearch()* shall return a pointer to the inserted item.

A null pointer shall be returned by *tsearch()* if there is not enough space available to create a new node.

A null pointer shall be returned by *tdelete()*, *tfind()*, and *tsearch()* if *rootp* is a null pointer on entry.

The *tdelete()* function shall return a pointer to the parent of the deleted node, or an unspecified non-null pointer if the deleted node was the root node, or a null pointer if the node is not found.

The *twalk()* function shall not return a value.

ERRORS

No errors are defined.

EXAMPLES

The following code reads in strings and stores structures containing a pointer to each string and a count of its length. It then walks the tree, printing out the stored strings and their lengths in alphabetical order.

```
#include <search.h>
#include <string.h>
#include <stdio.h>

#define STRSZ    10000
#define NODSZ    500

struct node {      /* Pointers to these are stored in the tree. */
    char    *string;
    int     length;
};

char    string_space[STRSZ]; /* Space to store strings. */
struct node nodes[NODSZ];   /* Nodes to store. */
void    *root = NULL;       /* This points to the root. */

int main(int argc, char *argv[])
{
    char    *strptr = string_space;
    struct node *nodeptr = nodes;
    void    print_node(const void *, VISIT, int);
```

```

67517         int    i = 0, node_compare(const void *, const void *);
67518         while (gets(strptr) != NULL && i++ < NODSZ) {
67519             /* Set node. */
67520             nodeptr->string = strptr;
67521             nodeptr->length = strlen(strptr);
67522             /* Put node into the tree. */
67523             (void) tsearch((void *)nodeptr, (void **)&root,
67524                 node_compare);
67525             /* Adjust pointers, so we do not overwrite tree. */
67526             strptr += nodeptr->length + 1;
67527             nodeptr++;
67528         }
67529         twalk(root, print_node);
67530         return 0;
67531     }
67532     /*
67533     * This routine compares two nodes, based on an
67534     * alphabetical ordering of the string field.
67535     */
67536     int
67537     node_compare(const void *node1, const void *node2)
67538     {
67539         return strcmp(((const struct node *) node1)->string,
67540             ((const struct node *) node2)->string);
67541     }
67542     /*
67543     * This routine prints out a node, the second time
67544     * twalk encounters it or if it is a leaf.
67545     */
67546     void
67547     print_node(const void *ptr, VISIT order, int level)
67548     {
67549         const struct node *p = *(const struct node **) ptr;
67550         if (order == postorder || order == leaf) {
67551             (void) printf("string = %s, length = %d\n",
67552                 p->string, p->length);
67553         }
67554     }

```

APPLICATION USAGE

The *root* argument to *twalk()* is one level of indirection less than the *rootp* arguments to *tdelete()* and *tsearch()*.

There are two nomenclatures used to refer to the order in which tree nodes are visited. The *tsearch()* function uses **preorder**, **postorder**, and **endorder** to refer respectively to visiting a node before any of its children, after its left child and before its right, and after both its children. The alternative nomenclature uses **preorder**, **inorder**, and **postorder** to refer to the same visits, which could result in some confusion over the meaning of **postorder**.

Since the return value of *tdelete()* is an unspecified non-null pointer in the case that the root of the tree has been deleted, applications should only use the return value of *tdelete()* as indication

67565 of success or failure and should not assume it can be dereferenced. Some implementations in this
67566 case will return a pointer to the new root of the tree (or to an empty tree if the deleted root node
67567 was the only node in the tree); other implementations return arbitrary non-null pointers.

67568 **RATIONALE**

67569 None.

67570 **FUTURE DIRECTIONS**

67571 None.

67572 **SEE ALSO**

67573 *hcreate()*, *lsearch()*

67574 XBD <*search.h*>

67575 **CHANGE HISTORY**

67576 First released in Issue 1. Derived from Issue 1 of the SVID.

67577 **Issue 5**

67578 The last paragraph of the DESCRIPTION was included as an APPLICATION USAGE note in
67579 previous issues.

67580 **Issue 6**

67581 The normative text is updated to avoid use of the term “must” for application requirements.

67582 The **restrict** keyword is added to the *tdelete()* prototype for alignment with the
67583 ISO/IEC 9899:1999 standard.

67584 **Issue 7**

67585 Austin Group Interpretation 1003.1-2001 #149 is applied, clarifying concurrent use of the tree in
67586 another thread.

67587 Austin Group Interpretation 1003.1-2001 #151 is applied, clarifying behavior for *tdelete()* when
67588 the deleted node is the root node.

67589 Austin Group Interpretation 1003.1-2001 #153 is applied.

67590 **NAME**

67591 telldir — current location of a named directory stream

67592 **SYNOPSIS**

```
67593 XSI      #include <dirent.h>
67594          long telldir(DIR *dirp);
```

67595 **DESCRIPTION**

67596 The *telldir()* function shall obtain the current location associated with the directory stream
67597 specified by *dirp*.

67598 If the most recent operation on the directory stream was a *seekdir()*, the directory position
67599 returned from the *telldir()* shall be the same as that supplied as a *loc* argument for *seekdir()*.

67600 **RETURN VALUE**

67601 Upon successful completion, *telldir()* shall return the current location of the specified directory
67602 stream.

67603 **ERRORS**

67604 No errors are defined.

67605 **EXAMPLES**

67606 None.

67607 **APPLICATION USAGE**

67608 None.

67609 **RATIONALE**

67610 None.

67611 **FUTURE DIRECTIONS**

67612 None.

67613 **SEE ALSO**

67614 *fdopendir()*, *readdir()*, *seekdir()*

67615 XBD <dirent.h>

67616 **CHANGE HISTORY**

67617 First released in Issue 2.

NAME

tempnam — create a name for a temporary file

SYNOPSIS

```
#include <stdio.h>
```

```
char *tempnam(const char *dir, const char *pfx);
```

DESCRIPTION

The *tempnam()* function shall generate a pathname that may be used for a temporary file.

The *tempnam()* function allows the user to control the choice of a directory. The *dir* argument points to the name of the directory in which the file is to be created. If *dir* is a null pointer or points to a string which is not a name for an appropriate directory, the path prefix defined as *P_tmpdir* in the *<stdio.h>* header shall be used. If that directory is not accessible, an implementation-defined directory may be used.

Many applications prefer their temporary files to have certain initial letter sequences in their names. The *pfx* argument should be used for this. This argument may be a null pointer or point to a string of up to five bytes to be used as the beginning of the filename.

Some implementations of *tempnam()* may use *tmpnam()* internally. On such implementations, if called more than {TMP_MAX} times in a single process, the behavior is implementation-defined.

RETURN VALUE

Upon successful completion, *tempnam()* shall allocate space for a string, put the generated pathname in that space, and return a pointer to it. The pointer shall be suitable for use in a subsequent call to *free()*. Otherwise, it shall return a null pointer and set *errno* to indicate the error.

ERRORS

The *tempnam()* function shall fail if:

[ENOMEM] Insufficient storage space is available.

EXAMPLES**Generating a Pathname**

The following example generates a pathname for a temporary file in directory */tmp*, with the prefix *file*. After the pathname has been created, the call to *free()* deallocates the space used to store the pathname.

```
#include <stdio.h>
#include <stdlib.h>
```

```
...
const char *directory = "/tmp";
const char *fileprefix = "file";
char *file;
```

```
file = tempnam(directory, fileprefix);
free(file);
```

APPLICATION USAGE

This function only creates pathnames. It is the application's responsibility to create and remove the files. Between the time a pathname is created and the file is opened, it is possible for some other process to create a file with the same name. Applications may find *tmpfile()* more useful.

Applications should use the *tmpfile()*, *mkdtemp()*, or *mkstemp()* functions instead of the

67661 obsolescent *tempnam()* function.

67662 **RATIONALE**

67663 None.

67664 **FUTURE DIRECTIONS**

67665 The *tempnam()* function may be removed in a future version.

67666 **SEE ALSO**

67667 *fopen()*, *free()*, *mkdtemp()*, *open()*, *tmpfile()*, *tmpnam()*, *unlink()*

67668 XBD <stdio.h>

67669 **CHANGE HISTORY**

67670 First released in Issue 1. Derived from Issue 1 of the SVID.

67671 **Issue 5**

67672 The last paragraph of the DESCRIPTION was included as an APPLICATION USAGE note in
67673 previous issues.

67674 **Issue 7**

67675 The *tempnam()* function is marked obsolescent.

67676 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0657 [291], XSH/TC1-2008/0658 [137], +
67677 and XSH/TC1-2008/0659 [137] are applied.

67678 **NAME**

67679 tfind — search binary search tree

67680 **SYNOPSIS**

```
67681 XSI      #include <search.h>
67682          void *tfind(const void *key, void *const *rootp,
67683                     int (*compar)(const void *, const void *));
```

67684 **DESCRIPTION**67685 Refer to *tdelete()*.

67686 **NAME**

67687 tgamma, tgammaf, tgammal — compute gamma() function

67688 **SYNOPSIS**

```
67689       #include <math.h>

67690       double tgamma(double x);
67691       float tgammaf(float x);
67692       long double tgammal(long double x);
```

67693 **DESCRIPTION**

67694 CX The functionality described on this reference page is aligned with the ISO C standard. Any
 67695 conflict between the requirements described here and the ISO C standard is unintentional. This
 67696 volume of POSIX.1-2008 defers to the ISO C standard.

67697 These functions shall compute the *gamma()* function of *x*.

67698 An application wishing to check for error situations should set *errno* to zero and call
 67699 *feclearexcept*(FE_ALL_EXCEPT) before calling these functions. On return, if *errno* is non-zero or
 67700 *fetestexcept*(FE_INVALID | FE_DIVBYZERO | FE_OVERFLOW | FE_UNDERFLOW) is non-
 67701 zero, an error has occurred.

67702 **RETURN VALUE**

67703 Upon successful completion, these functions shall return *Gamma(x)*.

67704 CX If *x* is a negative integer, a domain error may occur and either a NaN (if supported) or an
 67705 MX implementation-defined value shall be returned. On systems that support the IEC 60559
 67706 Floating-Point option, a domain error shall occur and a NaN shall be returned.

67707 If *x* is ± 0 , *tgamma()*, *tgammaf()*, and *tgammal()* shall return $\pm\text{HUGE\_VAL}$, $\pm\text{HUGE\_VALF}$, and
 67708 MX $\pm\text{HUGE\_VALL}$, respectively. On systems that support the IEC 60559 Floating-Point option, a
 67709 CX pole error shall occur; otherwise, a pole error may occur.

67710 If the correct value would cause overflow, a range error shall occur and *tgamma()*, *tgammaf()*,
 67711 and *tgammal()* shall return $\pm\text{HUGE\_VAL}$, $\pm\text{HUGE\_VALF}$, or $\pm\text{HUGE\_VALL}$, respectively, with
 67712 the same sign as the correct value of the function.

67713 MXX If the correct value would cause underflow, and is not representable, a range error may occur, +
 67714 MXX and *tgamma()*, *tgammaf()*, and *tgammal()* shall return 0.0, or (if IEC 60559 Floating-Point is not +
 67715 supported) an implementation-defined value no greater in magnitude than DBL_MIN, +
 67716 FLT_MIN, and LDBL_MIN, respectively. +

67717 MXX If the correct value would cause underflow, and is representable, a range error may occur and +
 67718 the correct value shall be returned. +

67719 MX If *x* is NaN, a NaN shall be returned.

67720 If *x* is +Inf, *x* shall be returned.

67721 If *x* is -Inf, a domain error shall occur, and a NaN shall be returned.

67722 **ERRORS**

67723 These functions shall fail if:

67724	MX	Domain Error	The value of <i>x</i> is a negative integer, or <i>x</i> is -Inf.
67725			If the integer expression (<i>math_errhandling</i> & MATH_ERRNO) is non-zero,
67726			then <i>errno</i> shall be set to [EDOM]. If the integer expression (<i>math_errhandling</i>
67727			& MATH_ERREXCEPT) is non-zero, then the invalid floating-point exception
67728			shall be raised.

67729	MX	Pole Error	The value of x is zero.	
67730			If the integer expression (<i>math_errhandling</i> & MATH_ERRNO) is non-zero,	
67731			then <i>errno</i> shall be set to [ERANGE]. If the integer expression	
67732			(<i>math_errhandling</i> & MATH_ERREXCEPT) is non-zero, then the divide-by-zero	
67733			floating-point exception shall be raised.	
67734		Range Error	The value overflows.	
67735			If the integer expression (<i>math_errhandling</i> & MATH_ERRNO) is non-zero,	
67736			then <i>errno</i> shall be set to [ERANGE]. If the integer expression	
67737			(<i>math_errhandling</i> & MATH_ERREXCEPT) is non-zero, then the overflow	
67738			floating-point exception shall be raised.	
67739		These functions may fail if:		
67740		Domain Error	The value of x is a negative integer.	
67741			If the integer expression (<i>math_errhandling</i> & MATH_ERRNO) is non-zero,	
67742			then <i>errno</i> shall be set to [EDOM]. If the integer expression (<i>math_errhandling</i>	
67743			& MATH_ERREXCEPT) is non-zero, then the invalid floating-point exception	
67744			shall be raised.	
67745		Pole Error	The value of x is zero.	
67746			If the integer expression (<i>math_errhandling</i> & MATH_ERRNO) is non-zero,	
67747			then <i>errno</i> shall be set to [ERANGE]. If the integer expression	
67748			(<i>math_errhandling</i> & MATH_ERREXCEPT) is non-zero, then the divide-by-zero	
67749			floating-point exception shall be raised.	
67750		Range Error	The result underflows.	+
67751			If the integer expression (<i>math_errhandling</i> & MATH_ERRNO) is non-zero,	+
67752			then <i>errno</i> shall be set to [ERANGE]. If the integer expression	+
67753			(<i>math_errhandling</i> & MATH_ERREXCEPT) is non-zero, then the underflow	+
67754			floating-point exception shall be raised.	
67755	EXAMPLES			
67756	None.			
67757	APPLICATION USAGE			
67758	For IEEE Std 754-1985 double , overflow happens when $0 < x < 1/\text{DBL\_MAX}$, and $171.7 < x$.			
67759	On error, the expressions (<i>math_errhandling</i> & MATH_ERRNO) and (<i>math_errhandling</i> &			
67760	MATH_ERREXCEPT) are independent of each other, but at least one of them must be non-zero.			
67761	RATIONALE			
67762	This function is named <i>tgamma()</i> in order to avoid conflicts with the historical <i>gamma()</i> and			
67763	<i>lgamma()</i> functions.			
67764	FUTURE DIRECTIONS			
67765	It is possible that the error response for a negative integer argument may be changed to a pole			
67766	error and a return value of $\pm\text{Inf}$.			
67767	SEE ALSO			
67768	<i>feclearexcept()</i> , <i>fetestexcept()</i> , <i>lgamma()</i>			
67769	XBD Section 4.19 (on page 116), <math.h>			

CHANGE HISTORY

67770 First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

67771
67772 IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/65 is applied, correcting the third
67773 paragraph in the RETURN VALUE section.

Issue 7

67774 ISO/IEC 9899:1999 standard, Technical Corrigendum 2 #52 (SD5-XSH-ERN-85) is applied.

67775
67776 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0660 [68], XSH/TC1-2008/0661 [320], +
67777 and XSH/TC1-2008/0662 [68] are applied.

67778 **NAME**

67779 time — get time

67780 **SYNOPSIS**

67781 #include <time.h>

67782 time_t time(time_t *tloc);

67783 **DESCRIPTION**

67784 CX The functionality described on this reference page is aligned with the ISO C standard. Any
 67785 conflict between the requirements described here and the ISO C standard is unintentional. This
 67786 volume of POSIX.1-2008 defers to the ISO C standard.

67787 CX The *time()* function shall return the value of time in seconds since the Epoch.

67788 The *tloc* argument points to an area where the return value is also stored. If *tloc* is a null pointer,
 67789 no value is stored.

67790 **RETURN VALUE**

67791 Upon successful completion, *time()* shall return the value of time. Otherwise, (**time_t**)−1 shall be
 67792 returned.

67793 **ERRORS**

67794 The *time()* function may fail if:

67795 CX [**EOverflow**] The number of seconds since the Epoch will not fit in an object of type **time_t**.

67796

67797 **EXAMPLES**67798 **Getting the Current Time**

67799 The following example uses the *time()* function to calculate the time elapsed, in seconds, since
 67800 the Epoch, *localtime()* to convert that value to a broken-down time, and *asctime()* to convert the
 67801 broken-down time values into a printable string.

67802 #include <stdio.h>

67803 #include <time.h>

67804 int main(void)

67805 {

67806 time_t result;

67807 result = time(NULL);

67808 printf("%s%ju secs since the Epoch\n",

67809 asctime(localtime(&result)),

67810 (uintmax_t)result);

67811 return(0);

67812 }

67813 This example writes the current time to *stdout* in a form like this:

67814 Wed Jun 26 10:32:15 1996

67815 835810335 secs since the Epoch

Timing an Event

The following example gets the current time, prints it out in the user's format, and prints the number of minutes to an event being timed.

```
#include <time.h>
#include <stdio.h>
...
time_t now;
int minutes_to_event;
...
time(&now);
minutes_to_event = ...;
printf("The time is ");
puts(asctime(localtime(&now)));
printf("There are %d minutes to the event.\n",
      minutes_to_event);
...
```

APPLICATION USAGE

None.

RATIONALE

The *time()* function returns a value in seconds while *clock_gettime()* and *gettimeofday()* return a **struct timespec** (seconds and nanoseconds) and **struct timeval** (seconds and microseconds), respectively, and are therefore capable of returning more precise times. The *times()* function is also capable of more precision than *time()* as it returns a value in clock ticks, although it returns the elapsed time since an arbitrary point such as system boot time, not since the epoch.

Implementations in which **time_t** is a 32-bit signed integer (many historical implementations) fail in the year 2038. POSIX.1-2008 does not address this problem. However, the use of the **time_t** type is mandated in order to ease the eventual fix.

On some systems the *time()* function is implemented using a system call that does not return an error condition in addition to the return value. On these systems it is impossible to differentiate between valid and invalid return values and hence overflow conditions cannot be reliably detected.

The use of the **<time.h>** header instead of **<sys/types.h>** allows compatibility with the ISO C standard.

Many historical implementations (including Version 7) and the 1984 /usr/group standard use **long** instead of **time_t**. This volume of POSIX.1-2008 uses the latter type in order to agree with the ISO C standard.

FUTURE DIRECTIONS

In a future version of this volume of POSIX.1-2008, **time_t** is likely to be required to be capable of representing times far in the future. Whether this will be mandated as a 64-bit type or a requirement that a specific date in the future be representable (for example, 10000 AD) is not yet determined. Systems purchased after the approval of this volume of POSIX.1-2008 should be evaluated to determine whether their lifetime will extend past 2038.

SEE ALSO

asctime(), *clock()*, *clock_getres()*, *ctime()*, *difftime()*, *futimens()*, *gettimeofday()*, *gmtime()*, *localtime()*, *mktime()*, *strftime()*, *strptime()*, *times()*, *utime()*

XBD **<time.h>**

CHANGE HISTORY

First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6

Extensions beyond the ISO C standard are marked.

The EXAMPLES, RATIONALE, and FUTURE DIRECTIONS sections are added.

Issue 7

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0663 [106], XSH/TC1-2008/0664 [350], +
XSH/TC1-2008/0665 [106], XSH/TC1-2008/0666 [350], and XSH/TC1-2008/0667 [350] are +
applied.

67871 NAME

67872 timer_create — create a per-process timer

67873 SYNOPSIS

```

67874 CX      #include <signal.h>
67875          #include <time.h>
67876
67876          int timer_create(clockid_t clockid, struct sigevent *restrict evp,
67877                          timer_t *restrict timerid);

```

67878 DESCRIPTION

67879 The *timer_create()* function shall create a per-process timer using the specified clock, *clock_id*, as
 67880 the timing base. The *timer_create()* function shall return, in the location referenced by *timerid*, a
 67881 timer ID of type **timer_t** used to identify the timer in timer requests. This timer ID shall be
 67882 unique within the calling process until the timer is deleted. The particular clock, *clock_id*, is
 67883 defined in **<time.h>**. The timer whose ID is returned shall be in a disarmed state upon return
 67884 from *timer_create()*.

67885 The *evp* argument, if non-NULL, points to a **sigevent** structure. This structure, allocated by the
 67886 application, defines the asynchronous notification to occur as specified in [Section 2.4.1](#) (on page
 67887 488) when the timer expires. If the *evp* argument is NULL, the effect is as if the *evp* argument
 67888 pointed to a **sigevent** structure with the *sigev_notify* member having the value SIGEV_SIGNAL,
 67889 the *sigev_signo* having a default signal number, and the *sigev_value* member having the value of
 67890 the timer ID.

67891 Each implementation shall define a set of clocks that can be used as timing bases for per-process
 67892 MON timers. All implementations shall support a *clock_id* of CLOCK_REALTIME. If the Monotonic
 67893 Clock option is supported, implementations shall support a *clock_id* of CLOCK_MONOTONIC.

67894 Per-process timers shall not be inherited by a child process across a *fork()* and shall be disarmed
 67895 and deleted by an *exec*.

67896 CPT If _POSIX_CPUTIME is defined, implementations shall support *clock_id* values representing the
 67897 CPU-time clock of the calling process.

67898 TCT If _POSIX_THREAD_CPUTIME is defined, implementations shall support *clock_id* values
 67899 representing the CPU-time clock of the calling thread.

67900 CPT|TCT It is implementation-defined whether a *timer_create()* function will succeed if the value defined
 67901 by *clock_id* corresponds to the CPU-time clock of a process or thread different from the process
 67902 or thread invoking the function.

67903 TSA If *evp->sigev_sigev_notify* is SIGEV_THREAD and *sev->sigev_notify_attributes* is not NULL, if the
 67904 attribute pointed to by *sev->sigev_notify_attributes* has a thread stack address specified by a call
 67905 to *pthread_attr_setstack()*, the results are unspecified if the signal is generated more than once.

67906 RETURN VALUE

67907 If the call succeeds, *timer_create()* shall return zero and update the location referenced by *timerid*
 67908 to a **timer_t**, which can be passed to the per-process timer calls. If an error occurs, the function
 67909 shall return a value of -1 and set *errno* to indicate the error. The value of *timerid* is undefined if
 67910 an error occurs.

67911 ERRORS

67912 The *timer_create()* function shall fail if:

67913 [EAGAIN] The system lacks sufficient signal queuing resources to honor the request.

67914	[EAGAIN]	The calling process has already created all of the timers it is allowed by this implementation.
67915		
67916	[EINVAL]	The specified clock ID is not defined.
67917	CPT TCT [ENOTSUP]	The implementation does not support the creation of a timer attached to the CPU-time clock that is specified by <i>clock_id</i> and associated with a process or thread different from the process or thread invoking <i>timer_create()</i> .
67918		
67919		

EXAMPLES

None.

APPLICATION USAGE

If a timer is created which has *evp->sigev_sigev_notify* set to SIGEV_THREAD and the attribute pointed to by *evp->sigev_notify_attributes* has a thread stack address specified by a call to *pthread_attr_setstack()*, the memory dedicated as a thread stack cannot be recovered. The reason for this is that the threads created in response to a timer expiration are created detached, or in an unspecified way if the thread attribute's *detachstate* is PTHREAD_CREATE_JOINABLE. In neither case is it valid to call *pthread_join()*, which makes it impossible to determine the lifetime of the created thread which thus means the stack memory cannot be reused.

RATIONALE**Periodic Timer Overrun and Resource Allocation**

The specified timer facilities may deliver realtime signals (that is, queued signals) on implementations that support this option. Since realtime applications cannot afford to lose notifications of asynchronous events, like timer expirations or asynchronous I/O completions, it must be possible to ensure that sufficient resources exist to deliver the signal when the event occurs. In general, this is not a difficulty because there is a one-to-one correspondence between a request and a subsequent signal generation. If the request cannot allocate the signal delivery resources, it can fail the call with an [EAGAIN] error.

Periodic timers are a special case. A single request can generate an unspecified number of signals. This is not a problem if the requesting process can service the signals as fast as they are generated, thus making the signal delivery resources available for delivery of subsequent periodic timer expiration signals. But, in general, this cannot be assured—processing of periodic timer signals may “overrun”; that is, subsequent periodic timer expirations may occur before the currently pending signal has been delivered.

Also, for signals, according to the POSIX.1-1990 standard, if subsequent occurrences of a pending signal are generated, it is implementation-defined whether a signal is delivered for each occurrence. This is not adequate for some realtime applications. So a mechanism is required to allow applications to detect how many timer expirations were delayed without requiring an indefinite amount of system resources to store the delayed expirations.

The specified facilities provide for an overrun count. The overrun count is defined as the number of extra timer expirations that occurred between the time a timer expiration signal is generated and the time the signal is delivered. The signal-catching function, if it is concerned with overruns, can retrieve this count on entry. With this method, a periodic timer only needs one “signal queuing resource” that can be allocated at the time of the *timer_create()* function call.

A function is defined to retrieve the overrun count so that an application need not allocate static storage to contain the count, and an implementation need not update this storage asynchronously on timer expirations. But, for some high-frequency periodic applications, the overhead of an additional system call on each timer expiration may be prohibitive. The functions, as defined, permit an implementation to maintain the overrun count in user space,

associated with the *timerid*. The *timer_getoverrun()* function can then be implemented as a macro that uses the *timerid* argument (which may just be a pointer to a user space structure containing the counter) to locate the overrun count with no system call overhead. Other implementations, less concerned with this class of applications, can avoid the asynchronous update of user space by maintaining the count in a system structure at the cost of the extra system call to obtain it.

Timer Expiration Signal Parameters

The Realtime Signals Extension option supports an application-specific datum that is delivered to the extended signal handler. This value is explicitly specified by the application, along with the signal number to be delivered, in a **sigevent** structure. The type of the application-defined value can be either an integer constant or a pointer. This explicit specification of the value, as opposed to always sending the timer ID, was selected based on existing practice.

It is common practice for realtime applications (on non-POSIX systems or realtime extended POSIX systems) to use the parameters of event handlers as the case label of a switch statement or as a pointer to an application-defined data structure. Since *timer_ids* are dynamically allocated by the *timer_create()* function, they can be used for neither of these functions without additional application overhead in the signal handler; for example, to search an array of saved timer IDs to associate the ID with a constant or application data structure.

FUTURE DIRECTIONS

None.

SEE ALSO

[*clock_getres\(\)*](#), [*timer_delete\(\)*](#), [*timer_getoverrun\(\)*](#)

XBD [**<signal.h>**](#), [**<time.h>**](#)

CHANGE HISTORY

First released in Issue 5. Included for alignment with the POSIX Realtime Extension.

Issue 6

The *timer_create()* function is marked as part of the Timers option.

The [ENOSYS] error condition has been removed as stubs need not be provided if an implementation does not support the Timers option.

CPU-time clocks are added for alignment with IEEE Std 1003.1d-1999.

The DESCRIPTION is updated for alignment with IEEE Std 1003.1j-2000 by adding the requirement for the CLOCK_MONOTONIC clock under the Monotonic Clock option.

The **restrict** keyword is added to the *timer_create()* prototype for alignment with the ISO/IEC 9899:1999 standard.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/138 is applied, updating the DESCRIPTION and APPLICATION USAGE sections to describe the case when a timer is created with the notification method set to SIGEV_THREAD.

Issue 7

The *timer_create()* function is moved from the Timers option to the Base.

67998 **NAME**

67999 timer_delete — delete a per-process timer

68000 **SYNOPSIS**

```
68001 CX      #include <time.h>
68002          int timer_delete(timer_t timerid);
```

68003 **DESCRIPTION**

68004 The *timer_delete()* function deletes the specified timer, *timerid*, previously created by the
 68005 *timer_create()* function. If the timer is armed when *timer_delete()* is called, the behavior shall be
 68006 as if the timer is automatically disarmed before removal. The disposition of pending signals for
 68007 the deleted timer is unspecified.

68008 **RETURN VALUE**

68009 If successful, the *timer_delete()* function shall return a value of zero. Otherwise, the function shall
 68010 return a value of -1 and set *errno* to indicate the error.

68011 **ERRORS**

68012 The *timer_delete()* function may fail if:

68013 [EINVAL] The timer ID specified by *timerid* is not a valid timer ID.

68014 **EXAMPLES**

68015 None.

68016 **APPLICATION USAGE**

68017 None.

68018 **RATIONALE**

68019 None.

68020 **FUTURE DIRECTIONS**

68021 None.

68022 **SEE ALSO**

68023 [*timer_create\(\)*](#)

68024 XBD [**<time.h>**](#)

68025 **CHANGE HISTORY**

68026 First released in Issue 5. Included for alignment with the POSIX Realtime Extension.

68027 **Issue 6**

68028 The *timer_delete()* function is marked as part of the Timers option.

68029 The [ENOSYS] error condition has been removed as stubs need not be provided if an
 68030 implementation does not support the Timers option.

68031 IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/139 is applied, updating the ERRORS
 68032 section so that the [EINVAL] error becomes optional.

68033 **Issue 7**

68034 The *timer_delete()* function is moved from the Timers option to the Base.

NAME

timer_getoverrun, timer_gettime, timer_settime — per-process timers

SYNOPSIS

```
#include <time.h>

int timer_getoverrun(timer_t timerid);
int timer_gettime(timer_t timerid, struct itimerspec *value);
int timer_settime(timer_t timerid, int flags,
    const struct itimerspec *restrict value,
    struct itimerspec *restrict ovalue);
```

DESCRIPTION

The *timer_gettime()* function shall store the amount of time until the specified timer, *timerid*, expires and the reload value of the timer into the space pointed to by the *value* argument. The *it_value* member of this structure shall contain the amount of time before the timer expires, or zero if the timer is disarmed. This value is returned as the interval until timer expiration, even if the timer was armed with absolute time. The *it_interval* member of *value* shall contain the reload value last set by *timer_settime()*.

The *timer_settime()* function shall set the time until the next expiration of the timer specified by *timerid* from the *it_value* member of the *value* argument and arm the timer if the *it_value* member of *value* is non-zero. If the specified timer was already armed when *timer_settime()* is called, this call shall reset the time until next expiration to the *value* specified. If the *it_value* member of *value* is zero, the timer shall be disarmed. The effect of disarming or resetting a timer with pending expiration notifications is unspecified.

If the flag *TIMER_ABSTIME* is not set in the argument *flags*, *timer_settime()* shall behave as if the time until next expiration is set to be equal to the interval specified by the *it_value* member of *value*. That is, the timer shall expire in *it_value* nanoseconds from when the call is made. If the flag *TIMER_ABSTIME* is set in the argument *flags*, *timer_settime()* shall behave as if the time until next expiration is set to be equal to the difference between the absolute time specified by the *it_value* member of *value* and the current value of the clock associated with *timerid*. That is, the timer shall expire when the clock reaches the value specified by the *it_value* member of *value*. If the specified time has already passed, the function shall succeed and the expiration notification shall be made.

The reload value of the timer shall be set to the value specified by the *it_interval* member of *value*. When a timer is armed with a non-zero *it_interval*, a periodic (or repetitive) timer is specified.

Time values that are between two consecutive non-negative integer multiples of the resolution of the specified timer shall be rounded up to the larger multiple of the resolution. Quantization error shall not cause the timer to expire earlier than the rounded time value.

If the argument *ovalue* is not NULL, the *timer_settime()* function shall store, in the location referenced by *ovalue*, a value representing the previous amount of time before the timer would have expired, or zero if the timer was disarmed, together with the previous timer reload value. Timers shall not expire before their scheduled time.

Only a single signal shall be queued to the process for a given timer at any point in time. When a timer for which a signal is still pending expires, no signal shall be queued, and a timer overrun shall occur. When a timer expiration signal is delivered to or accepted by a process, the *timer_getoverrun()* function shall return the timer expiration overrun count for the specified timer. The overrun count returned contains the number of extra timer expirations that occurred between the time the signal was generated (queued) and when it was delivered or accepted, up

to but not including an implementation-defined maximum of {DELAYTIMER_MAX}. If the number of such extra expirations is greater than or equal to {DELAYTIMER_MAX}, then the overrun count shall be set to {DELAYTIMER_MAX}. The value returned by *timer_getoverrun()* shall apply to the most recent expiration signal delivery or acceptance for the timer. If no expiration signal has been delivered for the timer, the return value of *timer_getoverrun()* is unspecified.

RETURN VALUE

If the *timer_getoverrun()* function succeeds, it shall return the timer expiration overrun count as explained above.

If the *timer_gettime()* or *timer_settime()* functions succeed, a value of 0 shall be returned.

If an error occurs for any of these functions, the value -1 shall be returned, and *errno* set to indicate the error.

ERRORS

The *timer_settime()* function shall fail if:

[EINVAL] A *value* structure specified a nanosecond value less than zero or greater than or equal to 1 000 million, and the *it_value* member of that structure did not specify zero seconds and nanoseconds.

These functions may fail if:

[EINVAL] The *timerid* argument does not correspond to an ID returned by *timer_create()* but not yet deleted by *timer_delete()*.

The *timer_settime()* function may fail if:

[EINVAL] The *it_interval* member of *value* is not zero and the timer was created with notification by creation of a new thread (*sigev_sigev_notify* was SIGEV_THREAD) and a fixed stack address has been set in the thread attribute pointed to by *sigev_notify_attributes*.

EXAMPLES

None.

APPLICATION USAGE

Using fixed stack addresses is problematic when timer expiration is signaled by the creation of a new thread. Since it cannot be assumed that the thread created for one expiration is finished before the next expiration of the timer, it could happen that two threads use the same memory as a stack at the same time. This is invalid and produces undefined results.

RATIONALE

Practical clocks tick at a finite rate, with rates of 100 hertz and 1 000 hertz being common. The inverse of this tick rate is the clock resolution, also called the clock granularity, which in either case is expressed as a time duration, being 10 milliseconds and 1 millisecond respectively for these common rates. The granularity of practical clocks implies that if one reads a given clock twice in rapid succession, one may get the same time value twice; and that timers must wait for the next clock tick after the theoretical expiration time, to ensure that a timer never returns too soon. Note also that the granularity of the clock may be significantly coarser than the resolution of the data format used to set and get time and interval values. Also note that some implementations may choose to adjust time and/or interval values to exactly match the ticks of the underlying clock.

This volume of POSIX.1-2008 defines functions that allow an application to determine the implementation-supported resolution for the clocks and requires an implementation to

document the resolution supported for timers and *nanosleep()* if they differ from the supported clock resolution. This is more of a procurement issue than a runtime application issue.

FUTURE DIRECTIONS

None.

SEE ALSO

clock_getres(), *timer_create()*

XBD <time.h>

CHANGE HISTORY

First released in Issue 5. Included for alignment with the POSIX Realtime Extension.

Issue 6

The *timer_getoverrun()*, *timer_gettime()*, and *timer_settime()* functions are marked as part of the Timers option.

The [ENOSYS] error condition has been removed as stubs need not be provided if an implementation does not support the Timers option.

The [EINVAL] error condition is updated to include the following: “and the *it_value* member of that structure did not specify zero seconds and nanoseconds.” This change is for IEEE PASC Interpretation 1003.1 #89.

The DESCRIPTION for *timer_getoverrun()* is updated to clarify that “If no expiration signal has been delivered for the timer, or if the Realtime Signals Extension is not supported, the return value of *timer_getoverrun()* is unspecified”.

The **restrict** keyword is added to the *timer_settime()* prototype for alignment with the ISO/IEC 9899:1999 standard.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/140 is applied, updating the ERRORS section so that the mandatory [EINVAL] error (“The *timerid* argument does not correspond to an ID returned by *timer_create()* but not yet deleted by *timer_delete()*”) becomes optional.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/141 is applied, updating the ERRORS section to include an optional [EINVAL] error for the case when a timer is created with the notification method set to SIGEV_THREAD. APPLICATION USAGE text is also added.

Issue 7

The *timer_getoverrun()*, *timer_gettime()*, and *timer_settime()* functions are moved from the Timers option to the Base.

Functionality relating to the Realtime Signals Extension option is moved to the Base.

NAME

times — get process and waited-for child process times

SYNOPSIS

```
#include <sys/times.h>

clock_t times(struct tms *buffer);
```

DESCRIPTION

The *times()* function shall fill the **tms** structure pointed to by *buffer* with time-accounting information. The **tms** structure is defined in **<sys/times.h>**.

All times are measured in terms of the number of clock ticks used.

The times of a terminated child process shall be included in the *tms_cutime* and *tms_cstime* elements of the parent when *wait()*, *waitid()*, or *waitpid()* returns the process ID of this terminated child. If a child process has not waited for its children, their times shall not be included in its times.

- The *tms_utime* structure member is the CPU time charged for the execution of user instructions of the calling process.
- The *tms_stime* structure member is the CPU time charged for execution by the system on behalf of the calling process.
- The *tms_cutime* structure member is the sum of the *tms_utime* and *tms_cutime* times of the child processes.
- The *tms_cstime* structure member is the sum of the *tms_stime* and *tms_cstime* times of the child processes.

RETURN VALUE

Upon successful completion, *times()* shall return the elapsed real time, in clock ticks, since an arbitrary point in the past (for example, system start-up time). This point does not change from one invocation of *times()* within the process to another. The return value may overflow the possible range of type **clock_t**. If *times()* fails, **(clock_t)-1** shall be returned and *errno* set to indicate the error.

ERRORS

No errors are defined.

EXAMPLES**Timing a Database Lookup**

The following example defines two functions, *start_clock()* and *end_clock()*, that are used to time a lookup. It also defines variables of type **clock_t** and **tms** to measure the duration of transactions. The *start_clock()* function saves the beginning times given by the *times()* function. The *end_clock()* function gets the ending times and prints the difference between the two times.

```
#include <sys/times.h>
#include <stdio.h>
...
void start_clock(void);
void end_clock(char *msg);
...
static clock_t st_time;
static clock_t en_time;
static struct tms st_cpu;
```

```

68203     static struct tms en_cpu;
68204     ...
68205     void
68206     start_clock()
68207     {
68208         st_time = times(&st_cpu);
68209     }

68210     /* This example assumes that the result of each subtraction
68211        is within the range of values that can be represented in
68212        an integer type. */
68213     void
68214     end_clock(char *msg)
68215     {
68216         en_time = times(&en_cpu);
68217         fputs(msg, stdout);
68218         printf("Real Time: %jd, User Time %jd, System Time %jd\n",
68219             (intmax_t)(en_time - st_time),
68220             (intmax_t)(en_cpu.tms_utime - st_cpu.tms_utime),
68221             (intmax_t)(en_cpu.tms_stime - st_cpu.tms_stime));
68222     }

```

APPLICATION USAGE

Applications should use `sysconf(_SC_CLK_TCK)` to determine the number of clock ticks per second as it may vary from system to system.

RATIONALE

The accuracy of the times reported is intentionally left unspecified to allow implementations flexibility in design, from uniprocessor to multi-processor networks.

The inclusion of times of child processes is recursive, so that a parent process may collect the total times of all of its descendants. But the times of a child are only added to those of its parent when its parent successfully waits on the child. Thus, it is not guaranteed that a parent process can always see the total times of all its descendants; see also the discussion of the term “realtime” in [alarm\(\)](#).

If the type `clock_t` is defined to be a signed 32-bit integer, it overflows in somewhat more than a year if there are 60 clock ticks per second, or less than a year if there are 100. There are individual systems that run continuously for longer than that. This volume of POSIX.1-2008 permits an implementation to make the reference point for the returned value be the start-up time of the process, rather than system start-up time.

The term “charge” in this context has nothing to do with billing for services. The operating system accounts for time used in this way. That information must be correct, regardless of how that information is used.

FUTURE DIRECTIONS

None.

SEE ALSO

[alarm\(\)](#), [exec](#), [fork\(\)](#), [sysconf\(\)](#), [time\(\)](#), [wait\(\)](#), [waitid\(\)](#)

XBD [<sys/times.h>](#)

68247
68248

CHANGE HISTORY
First released in Issue 1. Derived from Issue 1 of the SVID.

68249 NAME

68250 timezone — difference from UTC and local standard time

68251 SYNOPSIS

```
68252 XSI      #include <time.h>
68253          extern long timezone;
```

68254 DESCRIPTION

68255 Refer to *tzset()*.

68256 **NAME**

68257 tmpfile — create a temporary file

68258 **SYNOPSIS**

68259 #include <stdio.h>

68260 FILE *tmpfile(void);

68261 **DESCRIPTION**

68262 CX The functionality described on this reference page is aligned with the ISO C standard. Any
 68263 conflict between the requirements described here and the ISO C standard is unintentional. This
 68264 volume of POSIX.1-2008 defers to the ISO C standard.

68265 The *tmpfile()* function shall create a temporary file and open a corresponding stream. The file
 68266 shall be automatically deleted when all references to the file are closed. The file is opened as in
 68267 *fopen()* for update (*w+*), except that implementations may restrict the permissions, either by
 68268 clearing the file mode bits or setting them to the value *S_IRUSR* | *S_IWUSR*.

68269 CX In some implementations, a permanent file may be left behind if the process calling *tmpfile()* is
 68270 killed while it is processing a call to *tmpfile()*.

68271 An error message may be written to standard error if the stream cannot be opened.

68272 **RETURN VALUE**

68273 Upon successful completion, *tmpfile()* shall return a pointer to the stream of the file that is
 68274 CX created. Otherwise, it shall return a null pointer and set *errno* to indicate the error.

68275 **ERRORS**68276 The *tmpfile()* function shall fail if:

68277 CX [EINTR] A signal was caught during *tmpfile()*.

68278 CX [EMFILE] All file descriptors available to the process are currently open.

68279 CX [EMFILE] {STREAM_MAX} streams are currently open in the calling process.

68280 CX [ENFILE] The maximum allowable number of files is currently open in the system.

68281 CX [ENOSPC] The directory or file system which would contain the new file cannot be
 68282 expanded.

68283 CX [EOVERFLOW] The file is a regular file and the size of the file cannot be represented correctly
 68284 in an object of type *off_t*.

68285 The *tmpfile()* function may fail if:

68286 CX [EMFILE] {FOPEN_MAX} streams are currently open in the calling process.

68287 CX [ENOMEM] Insufficient storage space is available.

68288 **EXAMPLES**68289 **Creating a Temporary File**

68290 The following example creates a temporary file for update, and returns a pointer to a stream for
 68291 the created file in the *fp* variable.

68292 #include <stdio.h>

68293 ...

68294 FILE *fp;

68295 fp = tmpfile ();

APPLICATION USAGE

It should be possible to open at least {TMP_MAX} temporary files during the lifetime of the program (this limit may be shared with *tmpnam()*) and there should be no limit on the number simultaneously open other than this limit and any limit on the number of open file descriptors or streams ({OPEN_MAX}, {FOPEN_MAX}, {STREAM_MAX}).

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

Section 2.5 (on page 495), *fopen()*, *mkdtemp()*, *tmpnam()*, *unlink()*

XBD <stdio.h>

CHANGE HISTORY

First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 5

Large File Summit extensions are added.

The last two paragraphs of the DESCRIPTION were included as APPLICATION USAGE notes in previous issues.

Issue 6

Extensions beyond the ISO C standard are marked.

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- In the ERRORS section, the [EOVERFLOW] condition is added. This change is to support large files.
- The [EMFILE] optional error condition is added.

The APPLICATION USAGE section is added for alignment with the ISO/IEC 9899:1999 standard.

Issue 7

Austin Group Interpretation 1003.1-2001 #025 is applied, clarifying that implementations may restrict the permissions of the file created.

SD5-XBD-ERN-4 is applied, changing the definition of the [EMFILE] error.

SD5-XSH-ERN-149 is applied, adding the mandatory [EMFILE] error condition for {STREAM_MAX} streams open.

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0668 [14] is applied.

+

68330 **NAME**

68331 tmpnam — create a name for a temporary file

68332 **SYNOPSIS**

```
68333 OB      #include <stdio.h>
68334      char *tmpnam(char *s);
```

68335 **DESCRIPTION**

68336 CX The functionality described on this reference page is aligned with the ISO C standard. Any
 68337 conflict between the requirements described here and the ISO C standard is unintentional. This
 68338 volume of POSIX.1-2008 defers to the ISO C standard.

68339 The *tmpnam()* function shall generate a string that is a valid pathname that does not name an
 68340 existing file. The function is potentially capable of generating {TMP_MAX} different strings, but
 68341 any or all of them may already be in use by existing files and thus not be suitable return values.

68342 The *tmpnam()* function generates a different string each time it is called from the same process,
 68343 up to {TMP_MAX} times. If it is called more than {TMP_MAX} times, the behavior is
 68344 implementation-defined.

68345 The implementation shall behave as if no function defined in this volume of POSIX.1-2008,
 68346 except *tmpnam()*, calls *tmpnam()*.

68347 CX The *tmpnam()* function need not be thread-safe if called with a NULL parameter.

68348 **RETURN VALUE**

68349 Upon successful completion, *tmpnam()* shall return a pointer to a string. If no suitable string can
 68350 be generated, the *tmpnam()* function shall return a null pointer.

68351 If the argument *s* is a null pointer, *tmpnam()* shall leave its result in an internal static object and
 68352 return a pointer to that object. Subsequent calls to *tmpnam()* may modify the same object. If the
 68353 argument *s* is not a null pointer, it is presumed to point to an array of at least L_tmpnam **chars**;
 68354 *tmpnam()* shall write its result in that array and shall return the argument as its value.

68355 **ERRORS**

68356 No errors are defined.

68357 **EXAMPLES**68358 **Generating a Pathname**

68359 The following example generates a unique pathname and stores it in the array pointed to by *ptr*.

```
68360 #include <stdio.h>
68361 ...
68362 char pathname[L_tmpnam+1];
68363 char *ptr;
68364 ptr = tmpnam(pathname);
```

68365 **APPLICATION USAGE**

68366 This function only creates pathnames. It is the application's responsibility to create and remove
 68367 the files.

68368 Between the time a pathname is created and the file is opened, it is possible for some other
 68369 process to create a file with the same name. Applications may find *tmpfile()* more useful.

68370 Applications should use the *tmpfile()*, *mkstemp()*, or *mkdtemp()* functions instead of the
 68371 obsolescent *tmpnam()* function.

68372 **RATIONALE**

68373 None.

68374 **FUTURE DIRECTIONS**68375 The *tmpnam()* function may be removed in a future version.68376 **SEE ALSO**68377 *fopen()*, *open()*, *mkdtemp()*, *tempnam()*, *tmpfile()*, *unlink()*

68378 XBD <stdio.h>

68379 **CHANGE HISTORY**

68380 First released in Issue 1. Derived from Issue 1 of the SVID.

68381 **Issue 5**

68382 The DESCRIPTION is updated for alignment with the POSIX Threads Extension.

68383 **Issue 6**

68384 Extensions beyond the ISO C standard are marked.

68385 The normative text is updated to avoid use of the term “must” for application requirements.

68386 The DESCRIPTION is expanded for alignment with the ISO/IEC 9899:1999 standard.

68387 IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/142 is applied, updating the
68388 DESCRIPTION to allow implementations of the *tempnam()* function to call *tmpnam()*.68389 **Issue 7**68390 Austin Group Interpretation 1003.1-2001 #148 is applied, clarifying that the *tmpnam()* function
68391 need not be thread-safe if called with a NULL parameter.68392 The *tmpnam()* function is marked obsolescent.68393 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0669 [291] and XSH/TC1-2008/0670 +
68394 [291,429] are applied.

68395 NAME

68396 toascii — translate an integer to a 7-bit ASCII character

68397 SYNOPSIS

```
68398 OB XSI  #include <ctype.h>
68399         int toascii(int c);
```

68400 DESCRIPTION

68401 The *toascii()* function shall convert its argument into a 7-bit ASCII character.

68402 RETURN VALUE

68403 The *toascii()* function shall return the value (*c* &0x7f).

68404 ERRORS

68405 No errors are returned.

68406 EXAMPLES

68407 None.

68408 APPLICATION USAGE

68409 The *toascii()* function cannot be used portably in a localized application.

68410 RATIONALE

68411 None.

68412 FUTURE DIRECTIONS

68413 The *toascii()* function may be removed in a future version.

68414 SEE ALSO

68415 *isascii()*

68416 XBD <ctype.h>

68417 CHANGE HISTORY

68418 First released in Issue 1. Derived from Issue 1 of the SVID.

68419 Issue 7

68420 The *toascii()* function is marked obsolescent.

68421 **NAME**

68422 tolower, tolower_l — transliterate uppercase characters to lowercase

68423 **SYNOPSIS**

68424 #include <ctype.h>

68425 int tolower(int c);

68426 CX int tolower_l(int c, locale_t locale);

68427 **DESCRIPTION**

68428 CX For *tolower()*: The functionality described on this reference page is aligned with the ISO C
 68429 standard. Any conflict between the requirements described here and the ISO C standard is
 68430 unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

68431 CX The *tolower()* and *tolower_l()* functions have as a domain a type **int**, the value of which is
 68432 representable as an **unsigned char** or the value of EOF. If the argument has any other value, the
 68433 CX behavior is undefined. If the argument of *tolower()* or *tolower_l()* represents an uppercase letter,
 68434 and there exists a corresponding lowercase letter as defined by character type information in the
 68435 CX current locale or in the locale represented by *locale*, respectively (category *LC_CTYPE*), the
 68436 result shall be the corresponding lowercase letter. All other arguments in the domain are
 68437 returned unchanged.

68438 CX The behavior is undefined if the *locale* argument to *tolower_l()* is the special locale object
 68439 LC_GLOBAL_LOCALE or is not a valid locale object handle.

68440 **RETURN VALUE**

68441 CX Upon successful completion, the *tolower()* and *tolower_l()* functions shall return the lowercase
 68442 letter corresponding to the argument passed; otherwise, they shall return the argument
 68443 unchanged.

68444 **ERRORS**

68445 No errors are defined.

68446 **EXAMPLES**

68447 None.

68448 **APPLICATION USAGE**

68449 None.

68450 **RATIONALE**

68451 None.

68452 **FUTURE DIRECTIONS**

68453 None.

68454 **SEE ALSO**68455 *setlocale()*, *uselocale()*

68456 XBD Chapter 7 (on page 135), <ctype.h>, <locale.h>

68457 **CHANGE HISTORY**

68458 First released in Issue 1. Derived from Issue 1 of the SVID.

68459 **Issue 6**

68460 Extensions beyond the ISO C standard are marked.

68461 **Issue 7**

68462 The *tolower_l()* function is added from The Open Group Technical Standard, 2006, Extended API
68463 Set Part 4.

68464 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0671 [283] and XSH/TC1-2008/0672 +
68465 [283] are applied.

68466 **NAME**

68467 toupper, toupper_l — transliterate lowercase characters to uppercase

68468 **SYNOPSIS**

```
68469       #include <ctype.h>
68470       int toupper(int c);
68471 CX     int toupper_l(int c, locale_t locale);
```

68472 **DESCRIPTION**

68473 CX For *toupper()*: The functionality described on this reference page is aligned with the ISO C
68474 standard. Any conflict between the requirements described here and the ISO C standard is
68475 unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

68476 CX The *toupper()* and *toupper_l()* functions have as a domain a type **int**, the value of which is
68477 representable as an **unsigned char** or the value of EOF. If the argument has any other value, the
68478 behavior is undefined.

68479 CX If the argument of *toupper()* or *toupper_l()* represents a lowercase letter, and there exists a
68480 CX corresponding uppercase letter as defined by character type information in the current locale or
68481 in the locale represented by *locale*, respectively (category *LC_CTYPE*), the result shall be the
68482 corresponding uppercase letter.

68483 All other arguments in the domain are returned unchanged.

68484 CX The behavior is undefined if the *locale* argument to *toupper_l()* is the special locale object +
68485 LC_GLOBAL_LOCALE or is not a valid locale object handle.

68486 **RETURN VALUE**

68487 CX Upon successful completion, *toupper()* and *toupper_l()* shall return the uppercase letter
68488 corresponding to the argument passed; otherwise, they shall return the argument unchanged.

68489 **ERRORS**

68490 No errors are defined.

68491 **EXAMPLES**

68492 None.

68493 **APPLICATION USAGE**

68494 None.

68495 **RATIONALE**

68496 None.

68497 **FUTURE DIRECTIONS**

68498 None.

68499 **SEE ALSO**

68500 *setlocale()*, *uselocale()*
68501 XBD Chapter 7 (on page 135), *<ctype.h>*, *<locale.h>*

68502 **CHANGE HISTORY**

68503 First released in Issue 1. Derived from Issue 1 of the SVID.

68504 **Issue 6**

68505 Extensions beyond the ISO C standard are marked.

Issue 7

SD5-XSH-ERN-181 is applied, clarifying the RETURN VALUE section.

The *toupper_1()* function is added from The Open Group Technical Standard, 2006, Extended API Set Part 4.

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0673 [283] and XSH/TC1-2008/0674 + [283] are applied.

68512 **NAME**

68513 towctrans, towctrans_l — wide-character transliteration

68514 **SYNOPSIS**

```
68515 #include <wctype.h>
68516 wint_t towctrans(wint_t wc, wctrans_t desc);
68517 CX wint_t towctrans_l(wint_t wc, wctrans_t desc,
68518 locale_t locale);
```

68519 **DESCRIPTION**

68520 CX For *towctrans()*: The functionality described on this reference page is aligned with the ISO C
 68521 standard. Any conflict between the requirements described here and the ISO C standard is
 68522 unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

68523 CX The *towctrans()* and *towctrans_l()* functions shall transliterate the wide-character code *wc* using
 68524 the mapping described by *desc*.

68525 CX The current setting of the *LC_CTYPE* category in the current locale or in the locale represented
 68526 CX by *locale*, respectively, should be the same as during the call to *wctrans()* or *wctrans_l()* that
 68527 returned the value *desc*.

68528 If the value of *desc* is invalid (that is, not obtained by a call to *wctrans()* or *desc* is invalidated by a
 68529 subsequent call to *setlocale()* that has affected category *LC_CTYPE*), the result is unspecified.

68530 CX If the value of *desc* is invalid (that is, not obtained by a call to *wctrans_l()* with the same locale
 68531 object *locale*) the result is unspecified.

68532 CX An application wishing to check for error situations should set *errno* to 0 before calling
 68533 *towctrans()* or *towctrans_l()*.

68534 If *errno* is non-zero on return, an error has occurred.

68535 The behavior is undefined if the *locale* argument to *towctrans_l()* is the special locale object +
 68536 *LC_GLOBAL_LOCALE* or is not a valid locale object handle.

68537 **RETURN VALUE**

68538 CX If successful, the *towctrans()* and *towctrans_l()* functions shall return the mapped value of *wc*
 68539 using the mapping described by *desc*. Otherwise, they shall return *wc* unchanged.

68540 **ERRORS**

68541 These functions may fail if:

68542 CX [EINVAL] *desc* contains an invalid transliteration descriptor.

68543 **EXAMPLES**

68544 None.

68545 **APPLICATION USAGE**

68546 The strings "tolower" and "toupper" are reserved for the standard mapping names. In the
 68547 table below, the functions in the left column are equivalent to the functions in the right column.

68548 tolower(<i>wc</i>)	towctrans(<i>wc</i> , wctrans("tolower"))
68549 tolower_l(<i>wc</i> , <i>locale</i>)	towctrans_l(<i>wc</i> , wctrans("tolower"), <i>locale</i>)
68550 toupper(<i>wc</i>)	towctrans(<i>wc</i> , wctrans("toupper"))
68551 toupper_l(<i>wc</i> , <i>locale</i>)	towctrans_l(<i>wc</i> , wctrans("toupper"), <i>locale</i>)

68552 RATIONALE

68553 None.

68554 FUTURE DIRECTIONS

68555 None.

68556 SEE ALSO

68557 *tolower()*, *towupper()*, *wctrans()*

68558 XBD **<wctype.h>**

68559 CHANGE HISTORY

68560 First released in Issue 5. Derived from ISO/IEC 9899:1990/Amendment 1:1995 (E).

68561 Issue 6

68562 Extensions beyond the ISO C standard are marked.

68563 Issue 7

68564 The *towctrans_l()* function is added from The Open Group Technical Standard, 2006, Extended
68565 API Set Part 4.

68566 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0675 [302], XSH/TC1-2008/0676 [283], +
68567 and XSH/TC1-2008/0677 [283] are applied.

68568 **NAME**

68569 **towlower, towlower_l** — transliterate uppercase wide-character code to lowercase

68570 **SYNOPSIS**

68571 `#include <wctype.h>`

68572 `wint_t towlower(wint_t wc);`

68573 CX `wint_t towlower_l(wint_t wc, locale_t locale);`

68574 **DESCRIPTION**

68575 CX For *towlower()*: The functionality described on this reference page is aligned with the ISO C
68576 standard. Any conflict between the requirements described here and the ISO C standard is
68577 unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

68578 CX The *towlower()* and *towlower_l()* functions have as a domain a type **wint_t**, the value of which
68579 the application shall ensure is a character representable as a **wchar_t**, and a wide-character code
68580 corresponding to a valid character in the current locale or the value of WEOF. If the argument
68581 CX has any other value, the behavior is undefined. If the argument of *towlower()* or *towlower_l()*
68582 represents an uppercase wide-character code, and there exists a corresponding lowercase wide-
68583 CX character code as defined by character type information in the current locale or in the locale
68584 represented by *locale*, respectively (category *LC_CTYPE*), the result shall be the corresponding
68585 lowercase wide-character code. All other arguments in the domain are returned unchanged.

68586 CX The behavior is undefined if the *locale* argument to *towlower_l()* is the special locale object
68587 *LC_GLOBAL_LOCALE* or is not a valid locale object handle.

68588 **RETURN VALUE**

68589 CX Upon successful completion, the *towlower()* and *towlower_l()* functions shall return the
68590 lowercase letter corresponding to the argument passed; otherwise, they shall return the
68591 argument unchanged.

68592 **ERRORS**

68593 No errors are defined.

68594 **EXAMPLES**

68595 None.

68596 **APPLICATION USAGE**

68597 None.

68598 **RATIONALE**

68599 None.

68600 **FUTURE DIRECTIONS**

68601 None.

68602 **SEE ALSO**

68603 *setlocale()*, *uselocale()*

68604 XBD Chapter 7 (on page 135), *<locale.h>*, *<wctype.h>*

68605 **CHANGE HISTORY**

68606 First released in Issue 4.

68607 **Issue 5**

68608 The following change has been made in this version for alignment with
68609 ISO/IEC 9899:1990/Amendment 1:1995 (E):

- 68610 • The SYNOPSIS has been changed to indicate that this function and associated data types
68611 are now made visible by inclusion of the **<wctype.h>** header rather than **<wchar.h>**.

68612 **Issue 6**

68613 The normative text is updated to avoid use of the term “must” for application requirements.

68614 **Issue 7**

68615 The *tolower_l()* function is added from The Open Group Technical Standard, 2006, Extended
68616 API Set Part 4.

68617 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0678 [302], XSH/TC1-2008/0679 [283], +
68618 and XSH/TC1-2008/0680 [283] are applied.

68619 **NAME**

68620 towupper, towupper_l — transliterate lowercase wide-character code to uppercase

68621 **SYNOPSIS**

68622 #include <wctype.h>

68623 wint_t towupper(wint_t wc);

68624 CX wint_t towupper_l(wint_t wc, locale_t locale);

68625 **DESCRIPTION**

68626 CX For *towupper()*: The functionality described on this reference page is aligned with the ISO C
 68627 standard. Any conflict between the requirements described here and the ISO C standard is
 68628 unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

68629 CX The *towupper()* and *towupper_l()* functions have as a domain a type **wint_t**, the value of which
 68630 the application shall ensure is a character representable as a **wchar_t**, and a wide-character code
 68631 corresponding to a valid character in the current locale or the value of WEOF. If the argument
 68632 CX has any other value, the behavior is undefined. If the argument of *towupper()* or *towupper_l()*
 68633 represents a lowercase wide-character code, and there exists a corresponding uppercase wide-
 68634 CX character code as defined by character type information in the current locale or in the locale
 68635 represented by *locale*, respectively (category *LC_CTYPE*), the result shall be the corresponding
 68636 uppercase wide-character code. All other arguments in the domain are returned unchanged.

68637 CX The behavior is undefined if the *locale* argument to *towupper_l()* is the special locale object
 68638 LC_GLOBAL_LOCALE or is not a valid locale object handle.

68639 **RETURN VALUE**

68640 CX Upon successful completion, the *towupper()* and *towupper_l()* functions shall return the
 68641 uppercase letter corresponding to the argument passed. Otherwise, they shall return the
 68642 argument unchanged.

68643 **ERRORS**

68644 No errors are defined.

68645 **EXAMPLES**

68646 None.

68647 **APPLICATION USAGE**

68648 None.

68649 **RATIONALE**

68650 None.

68651 **FUTURE DIRECTIONS**

68652 None.

68653 **SEE ALSO**68654 *setlocale()*, *uselocale()*

68655 XBD Chapter 7 (on page 135), <locale.h>, <wctype.h>

68656 **CHANGE HISTORY**

68657 First released in Issue 4.

68658 **Issue 5**

68659 The following change has been made in this version for alignment with
68660 ISO/IEC 9899:1990/Amendment 1:1995 (E):

- 68661 • The SYNOPSIS has been changed to indicate that this function and associated data types
68662 are now made visible by inclusion of the **<wctype.h>** header rather than **<wchar.h>**.

68663 **Issue 6**

68664 The normative text is updated to avoid use of the term “must” for application requirements.

68665 **Issue 7**

68666 The *towupper_l()* function is added from The Open Group Technical Standard, 2006, Extended
68667 API Set Part 4.

68668 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0681 [302], XSH/TC1-2008/0682 [283], +
68669 and XSH/TC1-2008/0683 [283] are applied.

68670 **NAME**

68671 trunc, truncf, trunc1 — round to truncated integer value

68672 **SYNOPSIS**

```
68673     #include <math.h>
68674     double trunc(double x);
68675     float truncf(float x);
68676     long double trunc1(long double x);
```

68677 **DESCRIPTION**

68678 CX The functionality described on this reference page is aligned with the ISO C standard. Any
 68679 conflict between the requirements described here and the ISO C standard is unintentional. This
 68680 volume of POSIX.1-2008 defers to the ISO C standard.

68681 These functions shall round their argument to the integer value, in floating format, nearest to but
 68682 no larger in magnitude than the argument.

68683 **RETURN VALUE**

68684 Upon successful completion, these functions shall return the truncated integer value.

68685 MX The result shall have the same sign as *x*.

+

68686 MX If *x* is NaN, a NaN shall be returned.

68687 If *x* is ± 0 or $\pm \text{Inf}$, *x* shall be returned.

68688 **ERRORS**

68689 No errors are defined.

68690 **EXAMPLES**

68691 None.

68692 **APPLICATION USAGE**

68693 The integral value returned by these functions need not be expressible as an **intmax_t**. The
 68694 return value should be tested before assigning it to an integer type to avoid the undefined
 68695 results of an integer overflow.

68696 These functions may raise the inexact floating-point exception if the result differs in value from
 68697 the argument.

68698 **RATIONALE**

68699 None.

68700 **FUTURE DIRECTIONS**

68701 None.

68702 **SEE ALSO**

68703 XBD [<math.h>](#)

68704 **CHANGE HISTORY**

68705 First released in Issue 6. Derived from the ISO/IEC 9899: 1999 standard.

68706 **Issue 7**

68707 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0684 [346] is applied.

+

68708 **NAME**

68709 truncate — truncate a file to a specified length

68710 **SYNOPSIS**

68711 #include <unistd.h>

68712 int truncate(const char *path, off_t length);

68713 **DESCRIPTION**68714 The *truncate()* function shall cause the regular file named by *path* to have a size which shall be
68715 equal to *length* bytes.68716 If the file previously was larger than *length*, the extra data is discarded. If the file was previously
68717 shorter than *length*, its size is increased, and the extended area appears as if it were zero-filled.

68718 The application shall ensure that the process has write permission for the file.

68719 XSI If the request would cause the file size to exceed the soft file size limit for the process, the
68720 request shall fail and the implementation shall generate the SIGXFSZ signal for the process.68721 The *truncate()* function shall not modify the file offset for any open file descriptions associated
68722 with the file. Upon successful completion, if the file size is changed, *truncate()* shall mark for
68723 update the last data modification and last file status change timestamps of the file, and the
68724 S_ISUID and S_ISGID bits of the file mode may be cleared.68725 **RETURN VALUE**68726 Upon successful completion, *truncate()* shall return 0. Otherwise, -1 shall be returned, and *errno*
68727 set to indicate the error.68728 **ERRORS**68729 The *truncate()* function shall fail if:

68730 [EINTR] A signal was caught during execution.

68731 [EINVAL] The *length* argument was less than 0.

68732 [EFBIG] or [EINVAL]

68733 The *length* argument was greater than the maximum file size.

68734 [EIO] An I/O error occurred while reading from or writing to a file system.

68735 [EACCES] A component of the path prefix denies search permission, or write permission
68736 is denied on the file.

68737 [EISDIR] The named file is a directory.

68738 [ELOOP] A loop exists in symbolic links encountered during resolution of the *path*
68739 argument.

68740 [ENAMETOOLONG]

68741 The length of a component of a pathname is longer than {NAME_MAX}.

68742 [ENOENT] A component of *path* does not name an existing file or *path* is an empty string.68743 [ENOTDIR] A component of the path prefix names an existing file that is neither a |
68744 directory nor a symbolic link to a directory, or the *path* argument contains at
68745 least one non-*<slash>* character and ends with one or more trailing *<slash>*
68746 characters and the last pathname component names an existing file that is
68747 neither a directory nor a symbolic link to a directory.

68748 [EROFS] The named file resides on a read-only file system.

68749 The *truncate()* function may fail if:

68750 [ELOOP] More than {SYMLOOP_MAX} symbolic links were encountered during
68751 resolution of the *path* argument.

68752 [ENAMETOOLONG]
68753 The length of a pathname exceeds {PATH_MAX}, or pathname resolution of a
68754 symbolic link produced an intermediate result with a length that exceeds
68755 {PATH_MAX}.

68756 EXAMPLES

68757 None.

68758 APPLICATION USAGE

68759 None.

68760 RATIONALE

68761 None.

68762 FUTURE DIRECTIONS

68763 None.

68764 SEE ALSO

68765 [*open\(\)*](#)

68766 XBD [*<unistd.h>*](#)

68767 CHANGE HISTORY

68768 First released in Issue 4, Version 2.

68769 Issue 5

68770 Moved from X/OPEN UNIX extension to BASE.

68771 Large File Summit extensions are added.

68772 Issue 6

68773 This reference page is split out from the *ftruncate()* reference page.

68774 The normative text is updated to avoid use of the term “must” for application requirements.

68775 The wording of the mandatory [ELOOP] error condition is updated, and a second optional
68776 [ELOOP] error condition is added.

68777 Issue 7

68778 Austin Group Interpretation 1003.1-2001 #143 is applied.

68779 The *truncate()* function is moved from the XSI option to the Base.

68780 Changes are made related to support for finegrained timestamps.

68781 The [ENOTDIR] error condition is clarified to cover the condition where the last component of a
68782 pathname exists but is not a directory or a symbolic link to a directory.

68783 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0685 [324] is applied.

+

68784 **NAME**

68785 truncf, trunc — round to truncated integer value

68786 **SYNOPSIS**

68787 #include <math.h>

68788 float truncf(float x);

68789 long double trunc1(long double x);

68790 **DESCRIPTION**68791 Refer to *trunc()*.

68792 **NAME**

68793 tsearch — search a binary search tree

68794 **SYNOPSIS**

```
68795 XSI      #include <search.h>
68796          void *tsearch(const void *key, void **rootp,
68797                        int (*compar)(const void *, const void *));
```

68798 **DESCRIPTION**68799 Refer to *tdelete()*.

NAME

68801 `ttyname, ttyname_r` — find the pathname of a terminal

SYNOPSIS

68803 `#include <unistd.h>`

68804 `char *ttyname(int fildes);`

68805 `int ttyname_r(int fildes, char *name, size_t namesize);`

DESCRIPTION

68807 The `ttyname()` function shall return a pointer to a string containing a null-terminated pathname
 68808 of the terminal associated with file descriptor *fildes*. The application shall not modify the string
 68809 returned. The returned pointer might be invalidated or the string content might be overwritten
 68810 by a subsequent call to `ttyname()`.

68811 The `ttyname()` function need not be thread-safe.

68812 The `ttyname_r()` function shall store the null-terminated pathname of the terminal associated
 68813 with the file descriptor *fildes* in the character array referenced by *name*. The array is *namesize*
 68814 characters long and should have space for the name and the terminating null character. The
 68815 maximum length of the terminal name shall be {TTY_NAME_MAX}.

RETURN VALUE

68817 Upon successful completion, `ttyname()` shall return a pointer to a string. Otherwise, a null
 68818 pointer shall be returned and *errno* set to indicate the error.

68819 If successful, the `ttyname_r()` function shall return zero. Otherwise, an error number shall be
 68820 returned to indicate the error.

ERRORS

68822 The `ttyname()` function may fail if:

68823 [EBADF] The *fildes* argument is not a valid file descriptor.

68824 [ENOTTY] The file associated with the *fildes* argument is not a terminal.

68825 The `ttyname_r()` function may fail if:

68826 [EBADF] The *fildes* argument is not a valid file descriptor.

68827 [ENOTTY] The file associated with the *fildes* argument is not a terminal.

68828 [ERANGE] The value of *namesize* is smaller than the length of the string to be returned
 68829 including the terminating null character.

EXAMPLES

68831 None.

APPLICATION USAGE

68833 None.

RATIONALE

68835 The term “terminal” is used instead of the historical term “terminal device” in order to avoid a
 68836 reference to an undefined term.

68837 The thread-safe version places the terminal name in a user-supplied buffer and returns a non-
 68838 zero value if it fails. The non-thread-safe version may return the name in a static data area that
 68839 may be overwritten by each call.

68840 FUTURE DIRECTIONS

68841 None.

68842 SEE ALSO

68843 XBD <unistd.h>

68844 CHANGE HISTORY

68845 First released in Issue 1. Derived from Issue 1 of the SVID.

68846 Issue 5

68847 The *ttyname_r()* function is included for alignment with the POSIX Threads Extension.

68848 A note indicating that the *ttyname()* function need not be reentrant is added to the
68849 DESCRIPTION.

68850 Issue 6

68851 The *ttyname_r()* function is marked as part of the Thread-Safe Functions option.

68852 The following new requirements on POSIX implementations derive from alignment with the
68853 Single UNIX Specification:

- 68854 • The statement that *errno* is set on error is added.
- 68855 • The [EBADF] and [ENOTTY] optional error conditions are added.

68856 Issue 7

68857 Austin Group Interpretation 1003.1-2001 #156 is applied.

68858 SD5-XSH-ERN-100 is applied, correcting the definition of the [ENOTTY] error condition.

68859 The *ttyname_r()* function is moved from the Thread-Safe Functions option to the Base.

68860 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0686 [75] is applied.

+

68861 **NAME**

68862 twalk — traverse a binary search tree

68863 **SYNOPSIS**

```
68864 XSI      #include <search.h>
68865          void twalk(const void *root,
68866                   void (*action)(const void *, VISIT, int ));
```

68867 **DESCRIPTION**68868 Refer to *tdelete()*.

NAME

daylight, timezone, tzname, tzset — set timezone conversion information

SYNOPSIS

```
#include <time.h>
```

```
XSI    extern int daylight;
        extern long timezone;
CX     extern char *tzname[2];
        void tzset(void);
```

DESCRIPTION

The `tzset()` function shall use the value of the environment variable `TZ` to set time conversion information used by `ctime()`, `localtime()`, `mktime()`, and `strftime()`. If `TZ` is absent from the environment, implementation-defined default timezone information shall be used.

The `tzset()` function shall set the external variable `tzname` as follows:

```
tzname[0] = "std";
tzname[1] = "dst";
```

where `std` and `dst` are as described in XBD Chapter 8 (on page 173).

XSI The `tzset()` function also shall set the external variable `daylight` to 0 if Daylight Savings Time conversions should never be applied for the timezone in use; otherwise, non-zero. The external variable `timezone` shall be set to the difference, in seconds, between Coordinated Universal Time (UTC) and local standard time.

RETURN VALUE

The `tzset()` function shall not return a value.

ERRORS

No errors are defined.

EXAMPLES

Example `TZ` variables and their timezone differences are given in the table below:

<i>TZ</i>	<i>timezone</i>
EST5EDT	5*60*60
GMT0	0*60*60
JST-9	-9*60*60
MET-1MEST	-1*60*60
MST7MDT	7*60*60
PST8PDT	8*60*60

APPLICATION USAGE

None.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

`ctime()`, `localtime()`, `mktime()`, `strftime()`

XBD Chapter 8 (on page 173), `<time.h>`

68911	CHANGE HISTORY
68912	First released in Issue 1. Derived from Issue 1 of the SVID.
68913	Issue 6
68914	The example is corrected.

68915 **NAME**

68916 ulimit — get and set process limits

68917 **SYNOPSIS**

```
68918 OB XSI #include <ulimit.h>
68919 long ulimit(int cmd, ...);
```

68920 **DESCRIPTION**

68921 The *ulimit()* function shall control process limits. The process limits that can be controlled by
 68922 this function include the maximum size of a single file that can be written (this is equivalent to
 68923 using *setrlimit()* with RLIMIT_FSIZE). The *cmd* values, defined in **<ulimit.h>**, include:

68924 **UL_GETFSIZE** Return the file size limit (RLIMIT_FSIZE) of the process. The limit shall be in
 68925 units of 512-byte blocks and shall be inherited by child processes. Files of any
 68926 size can be read. The return value shall be the integer part of the soft file size
 68927 limit divided by 512. If the result cannot be represented as a **long**, the result is
 68928 unspecified.

68929 **UL_SETFSIZE** Set the file size limit for output operations of the process to the value of the
 68930 second argument, taken as a **long**, multiplied by 512. If the result would
 68931 overflow an **rlim_t**, the actual value set is unspecified. Any process may
 68932 decrease its own limit, but only a process with appropriate privileges may
 68933 increase the limit. The return value shall be the integer part of the new file size
 68934 limit divided by 512.

68935 The *ulimit()* function shall not change the setting of *errno* if successful.

68936 As all return values are permissible in a successful situation, an application wishing to check for
 68937 error situations should set *errno* to 0, then call *ulimit()*, and, if it returns -1, check to see if *errno* is
 68938 non-zero.

68939 **RETURN VALUE**

68940 Upon successful completion, *ulimit()* shall return the value of the requested limit. Otherwise, -1
 68941 shall be returned and *errno* set to indicate the error.

68942 **ERRORS**

68943 The *ulimit()* function shall fail and the limit shall be unchanged if:

68944	[EINVAL]	The <i>cmd</i> argument is not valid.
68945	[EPERM]	A process not having appropriate privileges attempts to increase its file size limit.
68946		

68947 **EXAMPLES**

68948 None.

68949 **APPLICATION USAGE**

68950 Since the *ulimit()* function uses type **long** rather than **rlim_t**, this function is not sufficient for file
 68951 sizes on many current systems. Applications should use the *getrlimit()* or *setrlimit()* functions
 68952 instead of the obsolescent *ulimit()* function.

68953 **RATIONALE**

68954 None.

68955 FUTURE DIRECTIONS

68956 The *ulimit()* function may be removed in a future version.

68957 SEE ALSO

68958 *exec*, *getrlimit()*, *write()*

68959 XBD <ulimit.h>

68960 CHANGE HISTORY

68961 First released in Issue 1. Derived from Issue 1 of the SVID.

68962 Issue 5

68963 In the description of UL_SETFSIZE, the text is corrected to refer to **rlim_t** rather than the
68964 spurious **rlimit_t**.

68965 The DESCRIPTION is updated to indicate that *errno* is not changed if the function is successful.

68966 Issue 7

68967 The *ulimit()* function is marked obsolescent.

NAME

umask — set and get the file mode creation mask

SYNOPSIS

```
#include <sys/stat.h>
```

```
mode_t umask(mode_t cmask);
```

DESCRIPTION

The *umask()* function shall set the file mode creation mask of the process to *cmask* and return the previous value of the mask. Only the file permission bits of *cmask* (see <sys/stat.h>) are used; the meaning of the other bits is implementation-defined.

The file mode creation mask of the process is used to turn off permission bits in the *mode* argument supplied during calls to the following functions:

- *open()*, *openat()*, *creat()*, *mkdir()*, *mkdirat()*, *mkfifo()*, and *mkfifoat()*
- *mknod()*, *mknodat()*
- *mq_open()*
- *sem_open()*

Bit positions that are set in *cmask* are cleared in the mode of the created file.

RETURN VALUE

The file permission bits in the value returned by *umask()* shall be the previous value of the file mode creation mask. The state of any other bits in that value is unspecified, except that a subsequent call to *umask()* with the returned value as *cmask* shall leave the state of the mask the same as its state before the first call, including any unspecified use of those bits.

ERRORS

No errors are defined.

EXAMPLES

None.

APPLICATION USAGE

None.

RATIONALE

Unsigned argument and return types for *umask()* were proposed. The return type and the argument were both changed to **mode_t**.

Historical implementations have made use of additional bits in *cmask* for their implementation-defined purposes. The addition of the text that the meaning of other bits of the field is implementation-defined permits these implementations to conform to this volume of POSIX.1-2008.

FUTURE DIRECTIONS

None.

SEE ALSO

creat(), *exec*, *mkdir()*, *mkfifo()*, *mknod()*, *mq_open()*, *open()*, *sem_open()*

XBD <sys/stat.h>, <sys/types.h>

69007 **CHANGE HISTORY**

69008 First released in Issue 1. Derived from Issue 1 of the SVID.

69009 **Issue 6**

69010 In the SYNOPSIS, the optional include of the **<sys/types.h>** header is removed.

69011 The following new requirements on POSIX implementations derive from alignment with the
69012 Single UNIX Specification:

- 69013 • The requirement to include **<sys/types.h>** has been removed. Although **<sys/types.h>** was
69014 required for conforming implementations of previous POSIX specifications, it was not
69015 required for UNIX applications.

69016 IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/143 is applied, adding the *mknod()*,
69017 *mq_open()*, and *sem_open()* functions to the DESCRIPTION and SEE ALSO sections.

69018 **NAME**69019 **uname** — get the name of the current system69020 **SYNOPSIS**

69021 #include <sys/utsname.h>

69022 int uname(struct utsname *name);

69023 **DESCRIPTION**69024 The *uname()* function shall store information identifying the current system in the structure pointed to by *name*.69026 The *uname()* function uses the **utsname** structure defined in <sys/utsname.h>.

69027 The *uname()* function shall return a string naming the current system in the character array *sysname*. Similarly, *nodename* shall contain the name of this node within an implementation-defined communications network. The arrays *release* and *version* shall further identify the operating system. The array *machine* shall contain a name that identifies the hardware that the system is running on.

69032 The format of each member is implementation-defined.

69033 **RETURN VALUE**

69034 Upon successful completion, a non-negative value shall be returned. Otherwise, -1 shall be returned and *errno* set to indicate the error.

69036 **ERRORS**

69037 No errors are defined.

69038 **EXAMPLES**

69039 None.

69040 **APPLICATION USAGE**

69041 The inclusion of the *nodename* member in this structure does not imply that it is sufficient information for interfacing to communications networks.

69043 **RATIONALE**

69044 The values of the structure members are not constrained to have any relation to the version of this volume of POSIX.1-2008 implemented in the operating system. An application should instead depend on `_POSIX_VERSION` and related constants defined in <unistd.h>.

69047 This volume of POSIX.1-2008 does not define the sizes of the members of the structure and permits them to be of different sizes, although most implementations define them all to be the same size: eight bytes plus one byte for the string terminator. That size for *nodename* is not enough for use with many networks.

69051 The *uname()* function originated in System III, System V, and related implementations, and it does not exist in Version 7 or 4.3 BSD. The values it returns are set at system compile time in those historical implementations.

69054 4.3 BSD has *gethostname()* and *gethostid()*, which return a symbolic name and a numeric value, respectively. There are related *sethostname()* and *sethostid()* functions that are used to set the values the other two functions return. The former functions are included in this specification, the latter are not.

69058 **FUTURE DIRECTIONS**

69059 None.

69060 **SEE ALSO**

69061 XBD [<sys/utsname.h>](#)

69062 **CHANGE HISTORY**

69063 First released in Issue 1. Derived from Issue 1 of the SVID.

NAME

ungetc — push byte back into input stream

SYNOPSIS

```
#include <stdio.h>
```

```
int ungetc(int c, FILE *stream);
```

DESCRIPTION

The functionality described on this reference page is aligned with the ISO C standard. Any conflict between the requirements described here and the ISO C standard is unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

The *ungetc()* function shall push the byte specified by *c* (converted to an **unsigned char**) back onto the input stream pointed to by *stream*. The pushed-back bytes shall be returned by subsequent reads on that stream in the reverse order of their pushing. A successful intervening call (with the stream pointed to by *stream*) to a file-positioning function (*fseek()*, *fseeko()*, *fsetpos()*, or *rewind()*) or *fflush()* shall discard any pushed-back bytes for the stream. The external storage corresponding to the stream shall be unchanged.

One byte of push-back shall be provided. If *ungetc()* is called too many times on the same stream without an intervening read or file-positioning operation on that stream, the operation may fail.

If the value of *c* equals that of the macro EOF, the operation shall fail and the input stream shall be left unchanged.

A successful call to *ungetc()* shall clear the end-of-file indicator for the stream. The value of the file-position indicator for the stream after all pushed-back bytes have been read, or discarded by calling *fseek()*, *fseeko()*, *fsetpos()*, or *rewind()* (but not *fflush()*), shall be the same as it was before the bytes were pushed back. The file-position indicator is decremented by each successful call to *ungetc()*; if its value was 0 before a call, its value is unspecified after the call.

RETURN VALUE

Upon successful completion, *ungetc()* shall return the byte pushed back after conversion. Otherwise, it shall return EOF.

ERRORS

No errors are defined.

EXAMPLES

None.

APPLICATION USAGE

None.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

Section 2.5 (on page 495), *fseek()*, *getc()*, *fsetpos()*, *read()*, *rewind()*, *setbuf()*

XBD <stdio.h>

CHANGE HISTORY

First released in Issue 1. Derived from Issue 1 of the SVID.

69106	Issue 7	
69107		POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0687 [87,93], XSH/TC1-2008/0688 +
69108		[87], and XSH/TC1-2008/0689 [14] are applied.

69109 **NAME**

69110 ungetwc — push wide-character code back into the input stream

69111 **SYNOPSIS**

```
69112 #include <stdio.h>
69113 #include <wchar.h>
69114 wint_t ungetwc(wint_t wc, FILE *stream);
```

69115 **DESCRIPTION**

69116 CX The functionality described on this reference page is aligned with the ISO C standard. Any
69117 conflict between the requirements described here and the ISO C standard is unintentional. This
69118 volume of POSIX.1-2008 defers to the ISO C standard.

69119 The *ungetwc()* function shall push the character corresponding to the wide-character code
69120 specified by *wc* back onto the input stream pointed to by *stream*. The pushed-back characters
69121 shall be returned by subsequent reads on that stream in the reverse order of their pushing. A
69122 successful intervening call (with the stream pointed to by *stream*) to a file-positioning function
69123 CX (*fseek()*, *fseeko()*, *fsetpos()*, or *rewind()*) or *fflush()* shall discard any pushed-back characters for
69124 the stream. The external storage corresponding to the stream is unchanged.

69125 At least one character of push-back shall be provided. If *ungetwc()* is called too many times on
69126 the same stream without an intervening read or file-positioning operation on that stream, the
69127 operation may fail.

69128 If the value of *wc* equals that of the macro WEOF, the operation shall fail and the input stream
69129 shall be left unchanged.

69130 A successful call to *ungetwc()* shall clear the end-of-file indicator for the stream. The value of the
69131 file-position indicator for the stream after all pushed-back characters have been read, or
69132 CX discarded by calling *fseek()*, *fseeko()*, *fsetpos()*, or *rewind()* (but not *fflush()*), shall be the same as
69133 it was before the characters were pushed back. The file-position indicator is decremented (by
69134 one or more) by each successful call to *ungetwc()*; if its value was 0 before a call, its value is
69135 unspecified after the call.

69136 **RETURN VALUE**

69137 Upon successful completion, *ungetwc()* shall return the wide-character code corresponding to
69138 the pushed-back character. Otherwise, it shall return WEOF.

69139 **ERRORS**

69140 The *ungetwc()* function may fail if:

69141 CX [EILSEQ] An invalid character sequence is detected, or a wide-character code does not
69142 correspond to a valid character.

69143 **EXAMPLES**

69144 None.

69145 **APPLICATION USAGE**

69146 None.

69147 **RATIONALE**

69148 None.

69149 **FUTURE DIRECTIONS**

69150 None.

SEE ALSO

Section 2.5 (on page 495), *fseek()*, *fsetpos()*, *read()*, *rewind()*, *setbuf()*

XBD **<stdio.h>**, **<wchar.h>**

CHANGE HISTORY

First released in Issue 4. Derived from the MSE working draft.

Issue 5

The Optional Header (OH) marking is removed from **<stdio.h>**.

Issue 6

The [EILSEQ] optional error condition is marked CX.

Issue 7

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0690 [87,93], XSH/TC1-2008/0691 + [87], and XSH/TC1-2008/0692 [14] are applied.

NAME

unlink, unlinkat — remove a directory entry relative to directory file descriptor

SYNOPSIS

```
#include <unistd.h>

int unlink(const char *path);
int unlinkat(int fd, const char *path, int flag);
```

DESCRIPTION

The *unlink()* function shall remove a link to a file. If *path* names a symbolic link, *unlink()* shall remove the symbolic link named by *path* and shall not affect any file or directory named by the contents of the symbolic link. Otherwise, *unlink()* shall remove the link named by the pathname pointed to by *path* and shall decrement the link count of the file referenced by the link.

When the file's link count becomes 0 and no process has the file open, the space occupied by the file shall be freed and the file shall no longer be accessible. If one or more processes have the file open when the last link is removed, the link shall be removed before *unlink()* returns, but the removal of the file contents shall be postponed until all references to the file are closed.

The *path* argument shall not name a directory unless the process has appropriate privileges and the implementation supports using *unlink()* on directories.

Upon successful completion, *unlink()* shall mark for update the last data modification and last file status change timestamps of the parent directory. Also, if the file's link count is not 0, the last file status change timestamp of the file shall be marked for update.

The *unlinkat()* function shall be equivalent to the *unlink()* or *rmdir()* function except in the case where *path* specifies a relative path. In this case the directory entry to be removed is determined relative to the directory associated with the file descriptor *fd* instead of the current working directory. If the file descriptor was opened without *O_SEARCH*, the function shall check whether directory searches are permitted using the current permissions of the directory underlying the file descriptor. If the file descriptor was opened with *O_SEARCH*, the function shall not perform the check.

Values for *flag* are constructed by a bitwise-inclusive OR of flags from the following list, defined in *<fcntl.h>*:

AT_REMOVEDIR

Remove the directory entry specified by *fd* and *path* as a directory, not a normal file.

If *unlinkat()* is passed the special value *AT_FDCWD* in the *fd* parameter, the current working directory shall be used and the behavior shall be identical to a call to *unlink()* or *rmdir()* respectively, depending on whether or not the *AT_REMOVEDIR* bit is set in *flag*.

RETURN VALUE

Upon successful completion, these functions shall return 0. Otherwise, these functions shall return -1 and set *errno* to indicate the error. If -1 is returned, the named file shall not be changed.

ERRORS

These functions shall fail and shall not unlink the file if:

[EACCES]	Search permission is denied for a component of the path prefix, or write permission is denied on the directory containing the directory entry to be removed.
[EBUSY]	The file named by the <i>path</i> argument cannot be unlinked because it is being used by the system or another process and the implementation considers this an error.

69208		[ELOOP]	A loop exists in symbolic links encountered during resolution of the <i>path</i> argument.
69209			
69210		[ENAMETOOLONG]	
69211			The length of a component of a pathname is longer than {NAME_MAX}.
69212		[ENOENT]	A component of <i>path</i> does not name an existing file or <i>path</i> is an empty string.
69213		[ENOTDIR]	A component of the <i>path</i> prefix names an existing file that is neither a directory nor a symbolic link to a directory, or the <i>path</i> argument contains at least one non- <code><slash></code> character and ends with one or more trailing <code><slash></code> characters and the last pathname component names an existing file that is neither a directory nor a symbolic link to a directory.
69214			
69215			
69216			
69217			
69218		[EPERM]	The file named by <i>path</i> is a directory, and either the calling process does not have appropriate privileges, or the implementation prohibits using <i>unlink()</i> on directories.
69219			
69220			
69221	XSI	[EPERM] or [EACCES]	
69222			The S_ISVTX flag is set on the directory containing the file referred to by the <i>path</i> argument and the process does not satisfy the criteria specified in XBD Section 4.2 (on page 107).
69223			
69224			
69225		[EROFS]	The directory entry to be unlinked is part of a read-only file system.
69226			The <i>unlinkat()</i> function shall fail if:
69227		[EACCES]	<i>fd</i> was not opened with O_SEARCH and the permissions of the directory underlying <i>fd</i> do not permit directory searches.
69228			
69229		[EBADF]	The <i>path</i> argument does not specify an absolute path and the <i>fd</i> argument is neither AT_FDCWD nor a valid file descriptor open for reading or searching.
69230			
69231		[ENOTDIR]	The <i>path</i> argument is not an absolute path and <i>fd</i> is a file descriptor associated with a non-directory file.
69232			
69233		[EEXIST] or [ENOTEMPTY]	
69234			The <i>flag</i> parameter has the AT_REMOVEDIR bit set and the <i>path</i> argument names a directory that is not an empty directory, or there are hard links to the directory other than dot or a single entry in dot-dot.
69235			
69236			
69237		[ENOTDIR]	The <i>flag</i> parameter has the AT_REMOVEDIR bit set and <i>path</i> does not name a directory.
69238			
69239			These functions may fail and not unlink the file if:
69240	XSI	[EBUSY]	The file named by <i>path</i> is a named STREAM.
69241		[ELOOP]	More than {SYMLOOP_MAX} symbolic links were encountered during resolution of the <i>path</i> argument.
69242			
69243		[ENAMETOOLONG]	
69244			The length of a pathname exceeds {PATH_MAX}, or pathname resolution of a symbolic link produced an intermediate result with a length that exceeds {PATH_MAX}.
69245			
69246			
69247		[ETXTBSY]	The entry to be unlinked is the last directory entry to a pure procedure (shared text) file that is being executed.
69248			

69249 The *unlinkat()* function may fail if:
 69250 [EINVAL] The value of the *flag* argument is not valid. -

69251 EXAMPLES

69252 Removing a Link to a File

69253 The following example shows how to remove a link to a file named */home/cnd/mod1* by
 69254 removing the entry named */modules/pass1*.

```
69255 #include <unistd.h>
69256 char *path = "/modules/pass1";
69257 int status;
69258 ...
69259 status = unlink(path);
```

69260 Checking for an Error

69261 The following example fragment creates a temporary password lock file named **LOCKFILE**,
 69262 which is defined as */etc/ptmp*, and gets a file descriptor for it. If the file cannot be opened for
 69263 writing, *unlink()* is used to remove the link between the file descriptor and **LOCKFILE**.

```
69264 #include <sys/types.h>
69265 #include <stdio.h>
69266 #include <fcntl.h>
69267 #include <errno.h>
69268 #include <unistd.h>
69269 #include <sys/stat.h>
69270 #define LOCKFILE "/etc/ptmp"
69271 int pfd; /* Integer for file descriptor returned by open call. */
69272 FILE *fpfd; /* File pointer for use in putpwent(). */
69273 ...
69274 /* Open password Lock file. If it exists, this is an error. */
69275 if ((pfd = open(LOCKFILE, O_WRONLY | O_CREAT | O_EXCL, S_IRUSR
69276 | S_IWUSR | S_IRGRP | S_IROTH)) == -1) {
69277     fprintf(stderr, "Cannot open /etc/ptmp. Try again later.\n");
69278     exit(1);
69279 }
69280 /* Lock file created; proceed with fdopen of lock file so that
69281 putpwent() can be used.
69282 */
69283 if ((fpfd = fdopen(pfd, "w")) == NULL) {
69284     close(pfd);
69285     unlink(LOCKFILE);
69286     exit(1);
69287 }
```

Replacing Files

The following example fragment uses *unlink()* to discard links to files, so that they can be replaced with new versions of the files. The first call removes the link to **LOCKFILE** if an error occurs. Successive calls remove the links to **SAVEFILE** and **PASSWDFILE** so that new links can be created, then removes the link to **LOCKFILE** when it is no longer needed.

```
#include <sys/types.h>
#include <stdio.h>
#include <fcntl.h>
#include <errno.h>
#include <unistd.h>
#include <sys/stat.h>

#define LOCKFILE "/etc/ptmp"
#define PASSWDFILE "/etc/passwd"
#define SAVEFILE "/etc/opasswd"
...
/* If no change was made, assume error and leave passwd unchanged. */
if (!valid_change) {
    fprintf(stderr, "Could not change password for user %s\n", user);
    unlink(LOCKFILE);
    exit(1);
}

/* Change permissions on new password file. */
chmod(LOCKFILE, S_IRUSR | S_IRGRP | S_IROTH);

/* Remove saved password file. */
unlink(SAVEFILE);

/* Save current password file. */
link(PASSWDFILE, SAVEFILE);

/* Remove current password file. */
unlink(PASSWDFILE);

/* Save new password file as current password file. */
link(LOCKFILE, PASSWDFILE);

/* Remove lock file. */
unlink(LOCKFILE);

exit(0);
```

APPLICATION USAGE

Applications should use *rmdir()* to remove a directory.

RATIONALE

Unlinking a directory is restricted to the superuser in many historical implementations for reasons given in *link()* (see also *rename()*).

The meaning of [EBUSY] in historical implementations is “mount point busy”. Since this volume of POSIX.1-2008 does not cover the system administration concepts of mounting and unmounting, the description of the error was changed to “resource busy”. (This meaning is used by some device drivers when a second process tries to open an exclusive use device.) The wording is also intended to allow implementations to refuse to remove a directory if it is the root or current working directory of any process.

The standard developers reviewed TR 24715-2006 and noted that LSB-conforming implementations may return [EISDIR] instead of [EPERM] when unlinking a directory. A change to permit this behavior by changing the requirement for [EPERM] to [EPERM] or [EISDIR] was considered, but decided against since it would break existing strictly conforming and conforming applications. Applications written for portability to both POSIX.1-2008 and the LSB should be prepared to handle either error code.

The purpose of the *unlinkat()* function is to remove directory entries in directories other than the current working directory without exposure to race conditions. Any part of the path of a file could be changed in parallel to a call to *unlink()*, resulting in unspecified behavior. By opening a file descriptor for the target directory and using the *unlinkat()* function it can be guaranteed that the removed directory entry is located relative to the desired directory.

FUTURE DIRECTIONS

None.

SEE ALSO

close(), *link()*, *remove()*, *rename()*, *rmdir()*, *symlink()*

XBD Section 4.2 (on page 107), *<fcntl.h>*, *<unistd.h>*

CHANGE HISTORY

First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 5

The [EBUSY] error is added to the optional part of the ERRORS section.

Issue 6

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- In the DESCRIPTION, the effect is specified if *path* specifies a symbolic link.
- The [ELOOP] mandatory error condition is added.
- A second [ENAMETOOLONG] is added as an optional error condition.
- The [ETXTBSY] optional error condition is added.

The following changes were made to align with the IEEE P1003.1a draft standard:

- The [ELOOP] optional error condition is added.

The normative text is updated to avoid use of the term “must” for application requirements.

Issue 7

Austin Group Interpretation 1003.1-2001 #143 is applied.

Austin Group Interpretation 1003.1-2001 #181 is applied, updating the requirements for operations when the S_ISVTX bit is set.

Text arising from the LSB Conflicts TR is added to the RATIONALE about the use of [EPERM] and [EISDIR].

The *unlinkat()* function is added from The Open Group Technical Standard, 2006, Extended API Set Part 2.

Changes are made related to support for finegrained timestamps.

Changes are made to allow a directory to be opened for searching.

69373 The [ENOTDIR] error condition is clarified to cover the condition where the last component of a
69374 pathname exists but is not a directory or a symbolic link to a directory.

69375 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0693 [461], XSH/TC1-2008/0694 [324], +
69376 XSH/TC1-2008/0695 [278], and XSH/TC1-2008/0696 [278] are applied.

69377 **NAME**

69378 unlockpt — unlock a pseudo-terminal master/slave pair

69379 **SYNOPSIS**

```
69380 XSI      #include <stdlib.h>
69381          int unlockpt(int filides);
```

69382 **DESCRIPTION**

69383 The *unlockpt()* function shall unlock the slave pseudo-terminal device associated with the master
69384 to which *filides* refers.

69385 Conforming applications shall ensure that they call *unlockpt()* before opening the slave side of a
69386 pseudo-terminal device.

69387 **RETURN VALUE**

69388 Upon successful completion, *unlockpt()* shall return 0. Otherwise, it shall return -1 and set *errno*
69389 to indicate the error.

69390 **ERRORS**

69391 The *unlockpt()* function may fail if:

69392 [EBADF] The *filides* argument is not a file descriptor open for writing.

69393 [EINVAL] The *filides* argument is not associated with a master pseudo-terminal device.

69394 **EXAMPLES**

69395 None.

69396 **APPLICATION USAGE**

69397 None.

69398 **RATIONALE**

69399 See the RATIONALE section for *posix_openpt()*.

69400 **FUTURE DIRECTIONS**

69401 None.

69402 **SEE ALSO**

69403 *grantpt()*, *open()*, *posix_openpt()*, *ptsname()*

69404 XBD <stdlib.h>

69405 **CHANGE HISTORY**

69406 First released in Issue 4, Version 2.

69407 **Issue 5**

69408 Moved from X/OPEN UNIX extension to BASE.

69409 **Issue 6**

69410 The normative text is updated to avoid use of the term “must” for application requirements.

69411 **Issue 7**

69412 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0697 [96] is applied.

NAME

unsetenv — remove an environment variable

SYNOPSIS

```
CX      #include <stdlib.h>
int unsetenv(const char *name);
```

DESCRIPTION

The *unsetenv()* function shall remove an environment variable from the environment of the calling process. The *name* argument points to a string, which is the name of the variable to be removed. The named argument shall not contain an '=' character. If the named variable does not exist in the current environment, the environment shall be unchanged and the function is considered to have completed successfully.

The *unsetenv()* function shall update the list of pointers to which *environ* points.

The *unsetenv()* function need not be thread-safe.

RETURN VALUE

Upon successful completion, zero shall be returned. Otherwise, -1 shall be returned, *errno* set to indicate the error, and the environment shall be unchanged.

ERRORS

The *unsetenv()* function shall fail if:

[EINVAL]	The <i>name</i> argument points to an empty string, or points to a string containing an '=' character.

EXAMPLES

None.

APPLICATION USAGE

None.

RATIONALE

Refer to the RATIONALE section in *setenv()*.

FUTURE DIRECTIONS

None.

SEE ALSO

getenv(), *setenv()*

XBD *<stdlib.h>*, *<sys/types.h>*

CHANGE HISTORY

First released in Issue 6. Derived from the IEEE P1003.1a draft standard.

Issue 7

Austin Group Interpretation 1003.1-2001 #156 is applied.

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0698 [167] and XSH/TC1-2008/0699 + [185] are applied.

NAME

uselocale — use locale in current thread

SYNOPSIS

```
CX      #include <locale.h>
      locale_t uselocale(locale_t newloc);
```

DESCRIPTION

The *uselocale()* function shall set the current locale for the current thread to the locale represented by *newloc*.

The value for the *newloc* argument shall be one of the following:

1. A value returned by the *newlocale()* or *duplocale()* functions
2. The special locale object descriptor LC_GLOBAL_LOCALE
3. (**locale_t**)0

Once the *uselocale()* function has been called to install a thread-local locale, the behavior of every interface using data from the current locale shall be affected for the calling thread. The current locale for other threads shall remain unchanged.

If the *newloc* argument is (**locale_t**)0, the object returned is the current locale or LC_GLOBAL_LOCALE if there has been no previous call to *uselocale()* for the current thread.

If the *newloc* argument is LC_GLOBAL_LOCALE, the thread shall use the global locale determined by the *setlocale()* function.

RETURN VALUE

Upon successful completion, the *uselocale()* function shall return the locale handle from the previous call for the current thread, or LC_GLOBAL_LOCALE if there was no such previous call. Otherwise, *uselocale()* shall return (**locale_t**)0 and set *errno* to indicate the error.

ERRORS

The *uselocale()* function may fail if:

[EINVAL] *locale* is not a valid locale object.

EXAMPLES

None.

APPLICATION USAGE

Unlike the *setlocale()* function, the *uselocale()* function does not allow replacing some locale categories only. Applications that need to install a locale which differs only in a few categories must use *newlocale()* to change a locale object equivalent to the currently used locale and install it.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

duplocale(), *freelocale()*, *newlocale()*, *setlocale()*

XBD **<locale.h>**

CHANGE HISTORY

69490
69491 First released in Issue 7.

69492 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0700 [290] and XSH/TC1-2008/0701 +
69493 [334] are applied.

NAME

utime — set file access and modification times

SYNOPSIS

```
#include <utime.h>

int utime(const char *path, const struct utimbuf *times);
```

DESCRIPTION

The *utime()* function shall set the access and modification times of the file named by the *path* argument.

If *times* is a null pointer, the access and modification times of the file shall be set to the current time. The effective user ID of the process shall match the owner of the file, or the process has write permission to the file or has appropriate privileges, to use *utime()* in this manner.

If *times* is not a null pointer, *times* shall be interpreted as a pointer to a **utimbuf** structure and the access and modification times shall be set to the values contained in the designated structure. Only a process with the effective user ID equal to the user ID of the file or a process with appropriate privileges may use *utime()* this way.

The **utimbuf** structure is defined in the **<utime.h>** header. The times in the structure **utimbuf** are measured in seconds since the Epoch.

Upon successful completion, the *utime()* function shall mark the last file status change timestamp for update; see **<sys/stat.h>**.

RETURN VALUE

Upon successful completion, 0 shall be returned. Otherwise, -1 shall be returned and *errno* shall be set to indicate the error, and the file times shall not be affected.

ERRORS

The *utime()* function shall fail if:

[EACCES] Search permission is denied by a component of the path prefix; or the *times* argument is a null pointer and the effective user ID of the process does not match the owner of the file, the process does not have write permission for the file, and the process does not have appropriate privileges.

[ELOOP] A loop exists in symbolic links encountered during resolution of the *path* argument.

[ENAMETOOLONG] The length of a component of a pathname is longer than {NAME_MAX}.

[ENOENT] A component of *path* does not name an existing file or *path* is an empty string.

[ENOTDIR] A component of the path prefix names an existing file that is neither a directory nor a symbolic link to a directory, or the *path* argument contains at least one non-**<slash>** character and ends with one or more trailing **<slash>** characters and the last pathname component names an existing file that is neither a directory nor a symbolic link to a directory.

[EPERM] The *times* argument is not a null pointer and the effective user ID of the calling process does not match the owner of the file and the calling process does not have appropriate privileges.

69535 [EROFS] The file system containing the file is read-only.
 69536 The *utime()* function may fail if:
 69537 [ELOOP] More than {SYMLOOP_MAX} symbolic links were encountered during
 69538 resolution of the *path* argument.
 69539 [ENAMETOOLONG]
 69540 The length of a pathname exceeds {PATH_MAX}, or pathname resolution of a
 69541 symbolic link produced an intermediate result with a length that exceeds
 69542 {PATH_MAX}.

69543 EXAMPLES

69544 None.

69545 APPLICATION USAGE

69546 Since the **utimbuf** structure only contains **time_t** variables and is not accurate to fractions of a
 69547 second, applications should use the *utimensat()* function instead of the obsolescent *utime()*
 69548 function.

69549 RATIONALE

69550 The *actime* structure member must be present so that an application may set it, even though an
 69551 implementation may ignore it and not change the last data access timestamp on the file. If an
 69552 application intends to leave one of the times of a file unchanged while changing the other, it
 69553 should use *stat()* or *fstat()* to retrieve the file's *st_atim* and *st_mtim* parameters, set *actime* and
 69554 *modtime* in the buffer, and change one of them before making the *utime()* call.

69555 FUTURE DIRECTIONS

69556 The *utime()* function may be removed in a future version.

69557 SEE ALSO

69558 *fstat()*, *fstatat()*, *futimens()*
 69559 XBD <sys/stat.h>, <utime.h>

69560 CHANGE HISTORY

69561 First released in Issue 1. Derived from Issue 1 of the SVID.

69562 Issue 6

69563 The following new requirements on POSIX implementations derive from alignment with the
 69564 Single UNIX Specification:

- 69565 • The requirement to include <sys/types.h> has been removed. Although <sys/types.h> was
 69566 required for conforming implementations of previous POSIX specifications, it was not
 69567 required for UNIX applications.
- 69568 • The [ELOOP] mandatory error condition is added.
- 69569 • A second [ENAMETOOLONG] is added as an optional error condition.

69570 The following changes were made to align with the IEEE P1003.1a draft standard:

- 69571 • The [ELOOP] optional error condition is added.

69572 The normative text is updated to avoid use of the term “must” for application requirements.

69573 Issue 7

69574 Austin Group Interpretation 1003.1-2001 #143 is applied.

69575 The *utime()* function is marked obsolescent.

69576 Changes are made related to support for finegrained timestamps.

 69577

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0702 [324] is applied.

+ 

69578 **NAME**

69579 utimensat, utimes — set file access and modification times relative to directory file descriptor

69580 **SYNOPSIS**

69581 #include <sys/stat.h>

69582 int utimensat(int *fd*, const char **path*, const struct timespec *times*[2],
69583 int *flag*);

69584 XSI #include <sys/time.h>

69585 int utimes(const char **path*, const struct timeval *times*[2]);69586 **DESCRIPTION**69587 Refer to *futimens()*.

69588 **NAME**

69589 va_arg, va_copy, va_end, va_start — handle variable argument list

69590 **SYNOPSIS**

69591 #include <stdarg.h>

69592 type va_arg(va_list ap, type);

69593 void va_copy(va_list dest, va_list src);

69594 void va_end(va_list ap);

69595 void va_start(va_list ap, argN);

69596 **DESCRIPTION**69597 Refer to XBD [<stdarg.h>](#)

69598 **NAME**69599 `vdprintf`, `vfprintf`, `vprintf`, `vsprintf`, `vsnprintf` — format output of a stdarg argument list69600 **SYNOPSIS**69601 `#include <stdarg.h>`69602 `#include <stdio.h>`

```

69603 CX int vdprintf(int fildes, const char *restrict format, va_list ap);
69604 int vfprintf(FILE *restrict stream, const char *restrict format,
69605 va_list ap);
69606 int vprintf(const char *restrict format, va_list ap);
69607 int vsnprintf(char *restrict s, size_t n, const char *restrict format,
69608 va_list ap);
69609 int vsprintf(char *restrict s, const char *restrict format, va_list ap);

```

69610 **DESCRIPTION**

69611 CX The functionality described on this reference page is aligned with the ISO C standard. Any
 69612 conflict between the requirements described here and the ISO C standard is unintentional. This
 69613 volume of POSIX.1-2008 defers to the ISO C standard.

69614 CX The `vdprintf()`, `vfprintf()`, `vprintf()`, `vsprintf()`, and `vsnprintf()` functions shall be equivalent to the
 69615 CX `dprintf()`, `fprintf()`, `printf()`, `snprintf()`, and `sprintf()` functions respectively, except that instead of
 69616 being called with a variable number of arguments, they are called with an argument list as
 69617 defined by `<stdarg.h>`.

69618 These functions shall not invoke the `va_end` macro. As these functions invoke the `va_arg` macro,
 69619 the value of `ap` after the return is unspecified.

69620 **RETURN VALUE**69621 Refer to `fprintf()`.69622 **ERRORS**69623 Refer to `fprintf()`.69624 **EXAMPLES**

69625 None.

69626 **APPLICATION USAGE**69627 Applications using these functions should call `va_end(ap)` afterwards to clean up.69628 **RATIONALE**

69629 None.

69630 **FUTURE DIRECTIONS**

69631 None.

69632 **SEE ALSO**69633 [Section 2.5](#) (on page 495), `fprintf()`69634 XBD `<stdarg.h>`, `<stdio.h>`69635 **CHANGE HISTORY**

69636 First released in Issue 1. Derived from Issue 1 of the SVID.

69637 **Issue 5**69638 The `vsprintf()` function is added.

69639 **Issue 6**

69640 The *vfprintf()*, *vprintf()*, *vsprintf()*, and *vsprintf()* functions are updated for alignment with the
 69641 ISO/IEC 9899:1999 standard.

69642 **Issue 7**

69643 The *vdprintf()* function is added to complement the *dprintf()* function from The Open Group
 69644 Technical Standard, 2006, Extended API Set Part 1.

69645 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0703 [14] is applied.

+

69646 **NAME**

69647 vfscanf, vscanf, vsscanf — format input of a stdarg argument list

69648 **SYNOPSIS**

69649 #include <stdarg.h>

69650 #include <stdio.h>

69651 int vfscanf(FILE *restrict stream, const char *restrict format,
69652 va_list arg);

69653 int vscanf(const char *restrict format, va_list arg);

69654 int vsscanf(const char *restrict s, const char *restrict format,
69655 va_list arg);69656 **DESCRIPTION**69657 **CX** The functionality described on this reference page is aligned with the ISO C standard. Any
69658 conflict between the requirements described here and the ISO C standard is unintentional. This
69659 volume of POSIX.1-2008 defers to the ISO C standard.69660 The *vscanf()*, *vfscanf()*, and *vsscanf()* functions shall be equivalent to the *scanf()*, *fscanf()*, and
69661 *sscanf()* functions, respectively, except that instead of being called with a variable number of
69662 arguments, they are called with an argument list as defined in the <stdarg.h> header. These
69663 functions shall not invoke the *va_end* macro. As these functions invoke the *va_arg* macro, the
69664 value of *ap* after the return is unspecified.69665 **RETURN VALUE**69666 Refer to *fscanf()*.69667 **ERRORS**69668 Refer to *fscanf()*.69669 **EXAMPLES**

69670 None.

69671 **APPLICATION USAGE**69672 Applications using these functions should call *va_end(ap)* afterwards to clean up.69673 **RATIONALE**

69674 None.

69675 **FUTURE DIRECTIONS**

69676 None.

69677 **SEE ALSO**69678 Section 2.5 (on page 495), *fscanf()*

69679 XBD <stdarg.h>, <stdio.h>

69680 **CHANGE HISTORY**

69681 First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

69682 **Issue 7**

69683 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0704 [14] is applied.

+

69684 **NAME**

69685 vfwprintf, vswprintf, vwprintf — wide-character formatted output of a stdarg argument list

69686 **SYNOPSIS**

```

69687 #include <stdarg.h>
69688 #include <stdio.h>
69689 #include <wchar.h>

69690 int vfwprintf(FILE *restrict stream, const wchar_t *restrict format,
69691             va_list arg);
69692 int vswprintf(wchar_t *restrict ws, size_t n,
69693             const wchar_t *restrict format, va_list arg);
69694 int vwprintf(const wchar_t *restrict format, va_list arg);

```

69695 **DESCRIPTION**

69696 CX The functionality described on this reference page is aligned with the ISO C standard. Any
 69697 conflict between the requirements described here and the ISO C standard is unintentional. This
 69698 volume of POSIX.1-2008 defers to the ISO C standard.

69699 The *vfwprintf()*, *vswprintf()*, and *vwprintf()* functions shall be equivalent to *fwprintf()*, *swprintf()*,
 69700 and *wprintf()* respectively, except that instead of being called with a variable number of
 69701 arguments, they are called with an argument list as defined by **<stdarg.h>**.

69702 These functions shall not invoke the *va_end* macro. However, as these functions do invoke the
 69703 *va_arg* macro, the value of *ap* after the return is unspecified.

69704 **RETURN VALUE**69705 Refer to *fwprintf()*.69706 **ERRORS**69707 Refer to *fwprintf()*.69708 **EXAMPLES**

69709 None.

69710 **APPLICATION USAGE**69711 Applications using these functions should call *va_end(ap)* afterwards to clean up.69712 **RATIONALE**

69713 None.

69714 **FUTURE DIRECTIONS**

69715 None.

69716 **SEE ALSO**69717 [Section 2.5](#) (on page 495), *fwprintf()*69718 XBD **<stdarg.h>**, **<stdio.h>**, **<wchar.h>**69719 **CHANGE HISTORY**

69720 First released in Issue 5. Included for alignment with ISO/IEC 9899:1990/Amendment 1:1995
 69721 (E).

69722 **Issue 6**

69723 The *vfwprintf()*, *vswprintf()*, and *vwprintf()* prototypes are updated for alignment with the
 69724 ISO/IEC 9899:1999 standard.

69725 **Issue 7**
69726

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0705 [14] is applied.

|

NAME

vfwscanf, vswscanf, wscanf — wide-character formatted input of a stdarg argument list

SYNOPSIS

```
#include <stdarg.h>
#include <stdio.h>
#include <wchar.h>

int vfwscanf(FILE *restrict stream, const wchar_t *restrict format,
             va_list arg);
int vswscanf(const wchar_t *restrict ws, const wchar_t *restrict format,
             va_list arg);
int vwscanf(const wchar_t *restrict format, va_list arg);
```

DESCRIPTION

CX The functionality described on this reference page is aligned with the ISO C standard. Any conflict between the requirements described here and the ISO C standard is unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

The *vfwscanf()*, *vswscanf()*, and *vwscanf()* functions shall be equivalent to the *fwscanf()*, *swscanf()*, and *wscanf()* functions, respectively, except that instead of being called with a variable number of arguments, they are called with an argument list as defined in the **<stdarg.h>** header. These functions shall not invoke the *va_end* macro. As these functions invoke the *va_arg* macro, the value of *ap* after the return is unspecified.

RETURN VALUE

Refer to *fwscanf()*.

ERRORS

Refer to *fwscanf()*.

EXAMPLES

None.

APPLICATION USAGE

Applications using these functions should call *va_end(ap)* afterwards to clean up.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

[Section 2.5](#) (on page 495), *fwscanf()*

XBD **<stdarg.h>**, **<stdio.h>**, **<wchar.h>**

CHANGE HISTORY

First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

Issue 7

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0706 [14] is applied.

+

69766 **NAME**

69767 vprintf — format the output of a stdarg argument list

69768 **SYNOPSIS**

69769 #include <stdarg.h>

69770 #include <stdio.h>

69771 int vprintf(const char *restrict *format*, va_list *ap*);69772 **DESCRIPTION**69773 Refer to *vfprintf()*.

69774 **NAME**

69775 vscanf — format input of a stdarg argument list

69776 **SYNOPSIS**

69777 #include <stdarg.h>

69778 #include <stdio.h>

69779 int vscanf(const char *restrict *format*, va_list *arg*);69780 **DESCRIPTION**69781 Refer to *vfscanf()*.

69782 **NAME**

69783 vsprintf, vsprintf — format output of a stdarg argument list

69784 **SYNOPSIS**

69785 #include <stdarg.h>

69786 #include <stdio.h>

```
69787       int vsnprintf(char *restrict s, size_t n,
69788                    const char *restrict format, va_list ap);
69789       int vsprintf(char *restrict s, const char *restrict format,
69790                    va_list ap);
```

69791 **DESCRIPTION**69792 Refer to *fprintf()*.

69793 **NAME**

69794 vsscanf — format input of a stdarg argument list

69795 **SYNOPSIS**

69796 #include <stdarg.h>

69797 #include <stdio.h>

69798 int vsscanf(const char *restrict *s*, const char *restrict *format*,69799 va_list *arg*);69800 **DESCRIPTION**69801 Refer to *vfscanf()*.

69802 **NAME**

69803 vswprintf — wide-character formatted output of a stdarg argument list

69804 **SYNOPSIS**

69805 #include <stdarg.h>

69806 #include <stdio.h>

69807 #include <wchar.h>

69808 int vswprintf(wchar_t *restrict *ws*, size_t *n*,69809 const wchar_t *restrict *format*, va_list *arg*);69810 **DESCRIPTION**69811 Refer to *vfwprintf()*.

69812 **NAME**

69813 vswscanf — wide-character formatted input of a stdarg argument list

69814 **SYNOPSIS**

69815 #include <stdarg.h>

69816 #include <stdio.h>

69817 #include <wchar.h>

```
69818       int vswscanf(const wchar_t *restrict ws, const wchar_t *restrict format,  
69819                   va_list arg);
```

69820 **DESCRIPTION**69821 Refer to *vfwscanf()*.

69822 **NAME**

69823 vwprintf — wide-character formatted output of a stdarg argument list

69824 **SYNOPSIS**

69825 #include <stdarg.h>

69826 #include <stdio.h>

69827 #include <wchar.h>

69828 int vwprintf(const wchar_t *restrict *format*, va_list *arg*);69829 **DESCRIPTION**69830 Refer to *vwprintf()*.

69831 **NAME**

69832 vwscanf — wide-character formatted input of a stdarg argument list

69833 **SYNOPSIS**

69834 #include <stdarg.h>

69835 #include <stdio.h>

69836 #include <wchar.h>

69837 int vwscanf(const wchar_t *restrict *format*, va_list *arg*);69838 **DESCRIPTION**69839 Refer to *vwscanf()*.

69840 **NAME**69841 `wait, waitpid` — wait for a child process to stop or terminate69842 **SYNOPSIS**69843 `#include <sys/wait.h>`69844 `pid_t wait(int *stat_loc);`69845 `pid_t waitpid(pid_t pid, int *stat_loc, int options);`69846 **DESCRIPTION**

69847 The `wait()` and `waitpid()` functions shall obtain status information pertaining to one of the
 69848 caller's child processes. Various options permit status information to be obtained for child
 69849 processes that have terminated or stopped. If status information is available for two or more
 69850 child processes, the order in which their status is reported is unspecified.

69851 The `wait()` function shall suspend execution of the calling thread until status information for one
 69852 of the terminated child processes of the calling process is available, or until delivery of a signal
 69853 whose action is either to execute a signal-catching function or to terminate the process. If more
 69854 than one thread is suspended in `wait()` or `waitpid()` awaiting termination of the same process,
 69855 exactly one thread shall return the process status at the time of the target process termination. If
 69856 status information is available prior to the call to `wait()`, return shall be immediate.

69857 The `waitpid()` function shall be equivalent to `wait()` if the `pid` argument is `(pid_t)-1` and the
 69858 `options` argument is 0. Otherwise, its behavior shall be modified by the values of the `pid` and
 69859 `options` arguments.

69860 The `pid` argument specifies a set of child processes for which `status` is requested. The `waitpid()`
 69861 function shall only return the status of a child process from this set:

- 69862 • If `pid` is equal to `(pid_t)-1`, `status` is requested for any child process. In this respect,
 69863 `waitpid()` is then equivalent to `wait()`.
- 69864 • If `pid` is greater than 0, it specifies the process ID of a single child process for which `status` is
 69865 requested.
- 69866 • If `pid` is 0, `status` is requested for any child process whose process group ID is equal to that
 69867 of the calling process.
- 69868 • If `pid` is less than `(pid_t)-1`, `status` is requested for any child process whose process group
 69869 ID is equal to the absolute value of `pid`.

69870 The `options` argument is constructed from the bitwise-inclusive OR of zero or more of the
 69871 following flags, defined in the `<sys/wait.h>` header:

69872 XSI **WCONTINUED** The `waitpid()` function shall report the status of any continued child process
 69873 specified by `pid` whose status has not been reported since it continued from a
 69874 job control stop.

69875 **WNOHANG** The `waitpid()` function shall not suspend execution of the calling thread if
 69876 `status` is not immediately available for one of the child processes specified by
 69877 `pid`.

69878 **WUNTRACED** The status of any child processes specified by `pid` that are stopped, and whose
 69879 status has not yet been reported since they stopped, shall also be reported to
 69880 the requesting process.

69881 XSI If the calling process has `SA_NOCLDWAIT` set or has `SIGCHLD` set to `SIG_IGN`, and the process
 69882 has no unwaited-for children that were transformed into zombie processes, the calling thread
 69883 shall block until all of the children of the process containing the calling thread terminate, and
 69884 `wait()` and `waitpid()` shall fail and set `errno` to `[ECHILD]`.

If *wait()* or *waitpid()* return because the status of a child process is available, these functions shall return a value equal to the process ID of the child process. In this case, if the value of the argument *stat_loc* is not a null pointer, information shall be stored in the location pointed to by *stat_loc*. The value stored at the location pointed to by *stat_loc* shall be 0 if and only if the status returned is from a terminated child process that terminated by one of the following means:

1. The process returned 0 from *main()*.
2. The process called *_exit()* or *exit()* with a *status* argument of 0.
3. The process was terminated because the last thread in the process terminated.

Regardless of its value, this information may be interpreted using the following macros, which are defined in **<sys/wait.h>** and evaluate to integral expressions; the *stat_val* argument is the integer value pointed to by *stat_loc*.

WIFEXITED(*stat_val*)

Evaluates to a non-zero value if *status* was returned for a child process that terminated normally.

WEXITSTATUS(*stat_val*)

If the value of WIFEXITED(*stat_val*) is non-zero, this macro evaluates to the low-order 8 bits of the *status* argument that the child process passed to *_exit()* or *exit()*, or the value the child process returned from *main()*.

WIFSIGNALED(*stat_val*)

Evaluates to a non-zero value if *status* was returned for a child process that terminated due to the receipt of a signal that was not caught (see **<signal.h>**).

WTERMSIG(*stat_val*)

If the value of WIFSIGNALED(*stat_val*) is non-zero, this macro evaluates to the number of the signal that caused the termination of the child process.

WIFSTOPPED(*stat_val*)

Evaluates to a non-zero value if *status* was returned for a child process that is currently stopped.

WSTOPSIG(*stat_val*)

If the value of WIFSTOPPED(*stat_val*) is non-zero, this macro evaluates to the number of the signal that caused the child process to stop.

WIFCONTINUED(*stat_val*)

Evaluates to a non-zero value if *status* was returned for a child process that has continued from a job control stop.

It is unspecified whether the *status* value returned by calls to *wait()* or *waitpid()* for processes created by *posix_spawn()* or *posix_spawnnp()* can indicate a WIFSTOPPED(*stat_val*) before subsequent calls to *wait()* or *waitpid()* indicate WIFEXITED(*stat_val*) as the result of an error detected before the new process image starts executing.

It is unspecified whether the *status* value returned by calls to *wait()* or *waitpid()* for processes created by *posix_spawn()* or *posix_spawnnp()* can indicate a WIFSIGNALED(*stat_val*) if a signal is sent to the parent's process group after *posix_spawn()* or *posix_spawnnp()* is called.

If the information pointed to by *stat_loc* was stored by a call to *waitpid()* that specified the WUNTRACED flag and did not specify the WCONTINUED flag, exactly one of the macros WIFEXITED(**stat_loc*), WIFSIGNALED(**stat_loc*), and WIFSTOPPED(**stat_loc*) shall evaluate to a non-zero value.

69929 If the information pointed to by *stat_loc* was stored by a call to *waitpid()* that specified the
 69930 XSI WUNTRACED and WCONTINUED flags, exactly one of the macros WIFEXITED(**stat_loc*),
 69931 XSI WIFSIGNALED(**stat_loc*), WIFSTOPPED(**stat_loc*), and WIFCONTINUED(**stat_loc*) shall
 69932 evaluate to a non-zero value.

69933 If the information pointed to by *stat_loc* was stored by a call to *waitpid()* that did not specify the
 69934 XSI WUNTRACED or WCONTINUED flags, or by a call to the *wait()* function, exactly one of the
 69935 macros WIFEXITED(**stat_loc*) and WIFSIGNALED(**stat_loc*) shall evaluate to a non-zero value.

69936 If the information pointed to by *stat_loc* was stored by a call to *waitpid()* that did not specify the
 69937 XSI WUNTRACED flag and specified the WCONTINUED flag, or by a call to the *wait()* function,
 69938 XSI exactly one of the macros WIFEXITED(**stat_loc*), WIFSIGNALED(**stat_loc*), and
 69939 WIFCONTINUED(**stat_loc*) shall evaluate to a non-zero value.

69940 If _POSIX_REALTIME_SIGNALS is defined, and the implementation queues the SIGCHLD
 69941 signal, then if *wait()* or *waitpid()* returns because the status of a child process is available, any
 69942 pending SIGCHLD signal associated with the process ID of the child process shall be discarded.
 69943 Any other pending SIGCHLD signals shall remain pending.

69944 Otherwise, if SIGCHLD is blocked, if *wait()* or *waitpid()* return because the status of a child
 69945 process is available, any pending SIGCHLD signal shall be cleared unless the status of another
 69946 child process is available.

69947 For all other conditions, it is unspecified whether child *status* will be available when a SIGCHLD
 69948 signal is delivered.

69949 There may be additional implementation-defined circumstances under which *wait()* or *waitpid()*
 69950 report *status*. This shall not occur unless the calling process or one of its child processes
 69951 explicitly makes use of a non-standard extension. In these cases the interpretation of the
 69952 reported *status* is implementation-defined.

69953 If a parent process terminates without waiting for all of its child processes to terminate, the
 69954 remaining child processes shall be assigned a new parent process ID corresponding to an
 69955 implementation-defined system process.

69956 RETURN VALUE

69957 If *wait()* or *waitpid()* returns because the status of a child process is available, these functions
 69958 shall return a value equal to the process ID of the child process for which *status* is reported. If
 69959 *wait()* or *waitpid()* returns due to the delivery of a signal to the calling process, -1 shall be
 69960 returned and *errno* set to [EINTR]. If *waitpid()* was invoked with WNOHANG set in *options*, it
 69961 has at least one child process specified by *pid* for which *status* is not available, and *status* is not
 69962 available for any process specified by *pid*, 0 is returned. Otherwise, -1 shall be returned, and
 69963 *errno* set to indicate the error.

69964 ERRORS

69965 The *wait()* function shall fail if:

69966 [ECHILD] The calling process has no existing unwaited-for child processes.

69967 [EINTR] The function was interrupted by a signal. The value of the location pointed to
 69968 by *stat_loc* is undefined.

69969 The *waitpid()* function shall fail if:

69970 [ECHILD] The process specified by *pid* does not exist or is not a child of the calling
 69971 process, or the process group specified by *pid* does not exist or does not have
 69972 any member process that is a child of the calling process.

69973 [EINTR] The function was interrupted by a signal. The value of the location pointed to
 69974 by *stat_loc* is undefined.

69975 [EINVAL] The *options* argument is not valid.

69976 EXAMPLES

69977 Waiting for a Child Process and then Checking its Status

69978 The following example demonstrates the use of *waitpid()*, *fork()*, and the macros used to
 69979 interpret the status value returned by *waitpid()* (and *wait()*). The code segment creates a child
 69980 process which does some unspecified work. Meanwhile the parent loops performing calls to
 69981 *waitpid()* to monitor the status of the child. The loop terminates when child termination is
 69982 detected.

```

69983 #include <stdio.h>
69984 #include <stdlib.h>
69985 #include <unistd.h>
69986 #include <sys/wait.h>
69987 ...
69988 pid_t child_pid, wpid;
69989 int status;
69990
69991 child_pid = fork();
69992 if (child_pid == -1) {          /* fork() failed */
69993     perror("fork");
69994     exit(EXIT_FAILURE);
69995 }
69996
69997 if (child_pid == 0) {          /* This is the child */
69998     /* Child does some work and then terminates */
69999     ...
70000 } else {                      /* This is the parent */
70001     do {
70002         wpid = waitpid(child_pid, &status, WUNTRACED
70003 #ifdef WCONTINUED             /* Not all implementations support this */
70004         | WCONTINUED
70005 #endif
70006         );
70007         if (wpid == -1) {
70008             perror("waitpid");
70009             exit(EXIT_FAILURE);
70010         }
70011         if (WIFEXITED(status)) {
70012             printf("child exited, status=%d\n", WEXITSTATUS(status));
70013         } else if (WIFSIGNALED(status)) {
70014             printf("child killed (signal %d)\n", WTERMSIG(status));
70015         } else if (WIFSTOPPED(status)) {
70016             printf("child stopped (signal %d)\n", WSTOPSIG(status));
70017 #ifdef WIFCONTINUED           /* Not all implementations support this */
70018         } else if (WIFCONTINUED(status)) {
70019             printf("child continued\n");

```

```

70018     #endif
70019         } else { /* Non-standard case -- may never happen */
70020             printf("Unexpected status (0x%x)\n", status);
70021         }
70022     } while (!WIFEXITED(status) && !WIFSIGNALED(status));
70023 }

```

Waiting for a Child Process in a Signal Handler for SIGCHLD

The following example demonstrates how to use *waitpid()* in a signal handler for SIGCHLD without passing -1 as the *pid* argument. (See the APPLICATION USAGE section below for the reasons why passing a *pid* of -1 is not recommended.) The method used here relies on the standard behavior of *waitpid()* when SIGCHLD is blocked. On historical non-conforming systems, the status of some child processes might not be reported.

```

70030 #include <stdlib.h>
70031 #include <stdio.h>
70032 #include <signal.h>
70033 #include <sys/types.h>
70034 #include <sys/wait.h>
70035 #include <unistd.h>
70036
70037 #define CHILDREN 10
70038
70039 static void
70040 handle_sigchld(int signum, siginfo_t *sinfo, void *unused)
70041 {
70042     int sav_errno = errno;
70043     int status;
70044
70045     /*
70046      * Obtain status information for the child which
70047      * caused the SIGCHLD signal and write its exit code
70048      * to stdout.
70049      */
70050     if (sinfo->si_code != CLD_EXITED)
70051     {
70052         static char msg[] = "wrong si_code\n";
70053         write(2, msg, sizeof msg - 1);
70054     }
70055     else if (waitpid(sinfo->si_pid, &status, 0) == -1)
70056     {
70057         static char msg[] = "waitpid() failed\n";
70058         write(2, msg, sizeof msg - 1);
70059     }
70060     else if (!WIFEXITED(status))
70061     {
70062         static char msg[] = "WIFEXITED was false\n";
70063         write(2, msg, sizeof msg - 1);
70064     }
70065     else
70066     {
70067         int code = WEXITSTATUS(status);
70068         char buf[2];

```

```

70066         buf[0] = '0' + code;
70067         buf[1] = '\n';
70068         write(1, buf, 2);
70069     }
70070     errno = sav_errno;
70071 }
70072
70073 int
70074 main(void)
70075 {
70076     int i;
70077     pid_t pid;
70078     struct sigaction sa;
70079
70080     sa.sa_flags = SA_SIGINFO;
70081     sa.sa_sigaction = handle_sigchld;
70082     sigemptyset(&sa.sa_mask);
70083     if (sigaction(SIGCHLD, &sa, NULL) == -1)
70084     {
70085         perror("sigaction");
70086         exit(EXIT_FAILURE);
70087     }
70088
70089     for (i = 0; i < CHILDREN; i++)
70090     {
70091         switch (pid = fork())
70092         {
70093             case -1:
70094                 perror("fork");
70095                 exit(EXIT_FAILURE);
70096             case 0:
70097                 sleep(2);
70098                 _exit(i);
70099             }
70100     }
70101
70102     /* Wait for all the SIGCHLD signals, then terminate on SIGALRM */
70103     alarm(3);
70104     for (;;)
70105         pause();
70106
70107     return 0; /* NOTREACHED */
70108 }

```

APPLICATION USAGE

Calls to *wait()* will collect information about any child process. This may result in interactions with other interfaces that may be waiting for their own children (such as by use of *system()*). For this and other reasons it is recommended that portable applications not use *wait()*, but instead use *waitpid()*. For these same reasons, the use of *waitpid()* with a *pid* argument of *-1*, and the use of *waitid()* with the *idtype* argument set to *P_ALL*, are also not recommended for portable applications.

RATIONALE

A call to the *wait()* or *waitpid()* function only returns *status* on an immediate child process of the calling process; that is, a child that was produced by a single *fork()* call (perhaps followed by an *exec* or other function calls) from the parent. If a child produces grandchildren by further use of *fork()*, none of those grandchildren nor any of their descendants affect the behavior of a *wait()* from the original parent process. Nothing in this volume of POSIX.1-2008 prevents an implementation from providing extensions that permit a process to get *status* from a grandchild or any other process, but a process that does not use such extensions must be guaranteed to see *status* from only its direct children.

The *waitpid()* function is provided for three reasons:

1. To support job control
2. To permit a non-blocking version of the *wait()* function
3. To permit a library routine, such as *system()* or *pclose()*, to wait for its children without interfering with other terminated children for which the process has not waited

The first two of these facilities are based on the *wait3()* function provided by 4.3 BSD. The function uses the *options* argument, which is equivalent to an argument to *wait3()*. The WUNTRACED flag is used only in conjunction with job control on systems supporting job control. Its name comes from 4.3 BSD and refers to the fact that there are two types of stopped processes in that implementation: processes being traced via the *ptrace()* debugging facility and (untraced) processes stopped by job control signals. Since *ptrace()* is not part of this volume of POSIX.1-2008, only the second type is relevant. The name WUNTRACED was retained because its usage is the same, even though the name is not intuitively meaningful in this context.

The third reason for the *waitpid()* function is to permit independent sections of a process to spawn and wait for children without interfering with each other. For example, the following problem occurs in developing a portable shell, or command interpreter:

```
stream = popen("/bin/true");
(void) system("sleep 100");
(void) pclose(stream);
```

On all historical implementations, the final *pclose()* fails to reap the *wait()* *status* of the *popen()*.

The status values are retrieved by macros, rather than given as specific bit encodings as they are in most historical implementations (and thus expected by existing programs). This was necessary to eliminate a limitation on the number of signals an implementation can support that was inherent in the traditional encodings. This volume of POSIX.1-2008 does require that a *status* value of zero corresponds to a process calling *_exit(0)*, as this is the most common encoding expected by existing programs. Some of the macro names were adopted from 4.3 BSD.

These macros syntactically operate on an arbitrary integer value. The behavior is undefined unless that value is one stored by a successful call to *wait()* or *waitpid()* in the location pointed to by the *stat_loc* argument. An early proposal attempted to make this clearer by specifying each argument as **stat_loc* rather than *stat_val*. However, that did not follow the conventions of other specifications in this volume of POSIX.1-2008 or traditional usage. It also could have implied that the argument to the macro must literally be **stat_loc*; in fact, that value can be stored or passed as an argument to other functions before being interpreted by these macros.

The extension that affects *wait()* and *waitpid()* and is common in historical implementations is the *ptrace()* function. It is called by a child process and causes that child to stop and return a *status* that appears identical to the *status* indicated by WIFSTOPPED. The *status* of *ptrace()* children is traditionally returned regardless of the WUNTRACED flag (or by the *wait()*)

function). Most applications do not need to concern themselves with such extensions because they have control over what extensions they or their children use. However, applications, such as command interpreters, that invoke arbitrary processes may see this behavior when those arbitrary processes misuse such extensions.

Implementations that support **core** file creation or other implementation-defined actions on termination of some processes traditionally provide a bit in the *status* returned by *wait()* to indicate that such actions have occurred.

Allowing the *wait()* family of functions to discard a pending SIGCHLD signal that is associated with a successfully waited-for child process puts them into the *sigwait()* and *sigwaitinfo()* category with respect to SIGCHLD.

This definition allows implementations to treat a pending SIGCHLD signal as accepted by the process in *wait()*, with the same meaning of “accepted” as when that word is applied to the *sigwait()* family of functions.

Allowing the *wait()* family of functions to behave this way permits an implementation to be able to deal precisely with SIGCHLD signals.

In particular, an implementation that does accept (discard) the SIGCHLD signal can make the following guarantees regardless of the queuing depth of signals in general (the list of waitable children can hold the SIGCHLD queue):

1. If a SIGCHLD signal handler is established via *sigaction()* without the SA_RESETHAND flag, SIGCHLD signals can be accurately counted; that is, exactly one SIGCHLD signal will be delivered to or accepted by the process for every child process that terminates.
2. A single *wait()* issued from a SIGCHLD signal handler can be guaranteed to return immediately with status information for a child process.
3. When SA_SIGINFO is requested, the SIGCHLD signal handler can be guaranteed to receive a non-null pointer to a **siginfo_t** structure that describes a child process for which a wait via *waitpid()* or *waitid()* will not block or fail.
4. The *system()* function will not cause the SIGCHLD handler of a process to be called as a result of the *fork()/exec* executed within *system()* because *system()* will accept the SIGCHLD signal when it performs a *waitpid()* for its child process. This is a desirable behavior of *system()* so that it can be used in a library without causing side-effects to the application linked with the library.

An implementation that does not permit the *wait()* family of functions to accept (discard) a pending SIGCHLD signal associated with a successfully waited-for child, cannot make the guarantees described above for the following reasons:

Guarantee #1

Although it might be assumed that reliable queuing of all SIGCHLD signals generated by the system can make this guarantee, the counter-example is the case of a process that blocks SIGCHLD and performs an indefinite loop of *fork()/wait()* operations. If the implementation supports queued signals, then eventually the system will run out of memory for the queue. The guarantee cannot be made because there must be some limit to the depth of queuing.

Guarantees #2 and #3

These cannot be guaranteed unless the *wait()* family of functions accepts the SIGCHLD signal. Otherwise, a *fork()/wait()* executed while SIGCHLD is blocked (as in the *system()* function) will result in an invocation of the handler when SIGCHLD is unblocked, after the process has disappeared.

Guarantee #4

Although possible to make this guarantee, *system()* would have to set the SIGCHLD handler to SIG_DFL so that the SIGCHLD signal generated by its *fork()* would be discarded (the SIGCHLD default action is to be ignored), then restore it to its previous setting. This would have the undesirable side-effect of discarding all SIGCHLD signals pending to the process.

FUTURE DIRECTIONS

None.

SEE ALSO

exec, *exit()*, *fork()*, *system()*, *waitid()*

XBD Section 4.11 (on page 110), [`<signal.h>`](#), [`<sys/wait.h>`](#)

CHANGE HISTORY

First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 5

The DESCRIPTION is updated for alignment with the POSIX Threads Extension.

Issue 6

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The requirement to include `<sys/types.h>` has been removed. Although `<sys/types.h>` was required for conforming implementations of previous POSIX specifications, it was not required for UNIX applications.

The following changes were made to align with the IEEE P1003.1a draft standard:

- The processing of the SIGCHLD signal and the [ECHILD] error is clarified.

The semantics of *WIFSTOPPED(stat_val)*, *WIFEXITED(stat_val)*, and *WIFSIGNALED(stat_val)* are defined with respect to *posix_spawn()* or *posix_spawnnp()* for alignment with IEEE Std 1003.1d-1999.

The DESCRIPTION is updated for alignment with the ISO/IEC 9899:1999 standard.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/145 is applied, adding the example to the EXAMPLES section.

Issue 7

SD5-XSH-ERN-202 is applied.

APPLICATION USAGE is added, recommending that the *wait()* function not be used.

An additional example for *waitpid()* is added.

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0707 [421], XSH/TC1-2008/0708 [166], + XSH/TC1-2008/0709 [166], and XSH/TC1-2008/0710 [69] are applied.

NAME

waitid — wait for a child process to change state

SYNOPSIS

```
#include <sys/wait.h>
```

```
int waitid(idtype_t idtype, id_t id, siginfo_t *infop, int options);
```

DESCRIPTION

The *waitid()* function shall suspend the calling thread until one child of the process containing the calling thread changes state. It records the current state of a child in the structure pointed to by *infop*. The fields of the structure pointed to by *infop* are filled in as described for the SIGCHLD signal in **<signal.h>**. If a child process changed state prior to the call to *waitid()*, *waitid()* shall return immediately. If more than one thread is suspended in *wait()*, *waitid()*, or *waitpid()* waiting for termination of the same process, exactly one thread shall return the process status at the time of the target process termination.

The *idtype* and *id* arguments are used to specify which children *waitid()* waits for.

If *idtype* is P_PID, *waitid()* shall wait for the child with a process ID equal to (**pid_t**)*id*.

If *idtype* is P_PGID, *waitid()* shall wait for any child with a process group ID equal to (**pid_t**)*id*.

If *idtype* is P_ALL, *waitid()* shall wait for any children and *id* is ignored.

The *options* argument is used to specify which state changes *waitid()* shall wait for. It is formed by OR'ing together the following flags:

WCONTINUED	Status shall be returned for any continued child process whose status either has not been reported since it continued from a job control stop or has been reported only by calls to <i>waitid()</i> with the WNOWAIT flag set.	

WEXITED	Wait for processes that have exited.	

WNOHANG	Do not hang if no status is available; return immediately.	

WNOWAIT	Keep the process whose status is returned in <i>infop</i> in a waitable state. This shall not affect the state of the process; the process may be waited for again after this call completes.	

WSTOPPED	Status shall be returned for any child that has stopped upon receipt of a signal, and whose status either has not been reported since it stopped or has been reported only by calls to <i>waitid()</i> with the WNOWAIT flag set.	

Applications shall specify at least one of the flags WEXITED, WSTOPPED, or WCONTINUED to be OR'ed in with the *options* argument.

The application shall ensure that the *infop* argument points to a **siginfo_t** structure. If *waitid()* returns because a child process was found that satisfied the conditions indicated by the arguments *idtype* and *options*, then the structure pointed to by *infop* shall be filled in by the system with the status of the process; the *si_signo* member shall be set equal to SIGCHLD. If *waitid()* returns because WNOHANG was specified and status is not available for any process specified by *idtype* and *id*, then the *si_signo* and *si_pid* members of the structure pointed to by *infop* shall be set to zero and the values of other members of the structure are unspecified.

RETURN VALUE

If WNOHANG was specified and status is not available for any process specified by *idtype* and *id*, 0 shall be returned. If *waitid()* returns due to the change of state of one of its children, 0 shall be returned. Otherwise, -1 shall be returned and *errno* set to indicate the error.

ERRORS

The *waitid()* function shall fail if:

- [ECHILD] The calling process has no existing unwaited-for child processes.
- [EINTR] The *waitid()* function was interrupted by a signal.
- [EINVAL] An invalid value was specified for *options*, or *idtype* and *id* specify an invalid set of processes.

EXAMPLES

None.

APPLICATION USAGE

Calls to *waitid()* with *idtype* equal to P_ALL will collect information about any child process. This may result in interactions with other interfaces that may be waiting for their own children (such as by use of *system()*). For this reason it is recommended that portable applications not use *waitid()* with *idtype* of P_ALL. See also APPLICATION USAGE for *wait()*.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

exec, *exit()*, *wait()*

XBD <signal.h>, <sys/wait.h>

CHANGE HISTORY

First released in Issue 4, Version 2.

Issue 5

Moved from X/OPEN UNIX extension to BASE.

The DESCRIPTION is updated for alignment with the POSIX Threads Extension.

Issue 6

The normative text is updated to avoid use of the term “must” for application requirements.

Issue 7

Austin Group Interpretation 1003.1-2001 #060 is applied, updating the DESCRIPTION.

The *waitid()* function is moved from the XSI option to the Base.

APPLICATION USAGE is added, recommending that the *waitid()* function not be used with *idtype* equal to P_ALL.

The description of the WNOHANG flag is updated.

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0711 [154], XSH/TC1-2008/0712 [154], + and XSH/TC1-2008/0713 [153] are applied.

70316 **NAME**

70317 waitpid — wait for a child process to stop or terminate

70318 **SYNOPSIS**

70319 #include <sys/wait.h>

70320 pid_t waitpid(pid_t *pid*, int **stat_loc*, int *options*);

70321 **DESCRIPTION**

70322 Refer to *wait()*.

70323 **NAME**

70324 wcpcpy — copy a wide-character string, returning a pointer to its end

70325 **SYNOPSIS**

```
70326 CX       #include <wchar.h>  
70327       wchar_t *wcpcpy(wchar_t *restrict ws1, const wchar_t *restrict ws2);
```

70328 **DESCRIPTION**70329 Refer to *wcscpy()*.

70330 **NAME**

70331 wcpncpy — copy a fixed-size wide-character string, returning a pointer to its end

70332 **SYNOPSIS**

```
70333 CX      #include <wchar.h>
70334          wchar_t *wcpncpy(wchar_t restrict *ws1, const wchar_t *restrict ws2,
70335                          size_t n);
```

70336 **DESCRIPTION**70337 Refer to *wcsncpy()*.

70338 **NAME**70339 `wrtomb` — convert a wide-character code to a character (restartable)70340 **SYNOPSIS**70341 `#include <wchar.h>`70342 `size_t wrtomb(char *restrict s, wchar_t wc, mbstate_t *restrict ps);`70343 **DESCRIPTION**

70344 CX The functionality described on this reference page is aligned with the ISO C standard. Any
 70345 conflict between the requirements described here and the ISO C standard is unintentional. This
 70346 volume of POSIX.1-2008 defers to the ISO C standard.

70347 If *s* is a null pointer, the `wrtomb()` function shall be equivalent to the call:70348 `wrtomb(buf, L'\0', ps)`70349 where *buf* is an internal buffer.

70350 If *s* is not a null pointer, the `wrtomb()` function shall determine the number of bytes needed to
 70351 represent the character that corresponds to the wide character given by *wc* (including any shift
 70352 sequences), and store the resulting bytes in the array whose first element is pointed to by *s*. At
 70353 most {MB_CUR_MAX} bytes are stored. If *wc* is a null wide character, a null byte shall be stored,
 70354 preceded by any shift sequence needed to restore the initial shift state. The resulting state
 70355 described shall be the initial conversion state.

70356 If *ps* is a null pointer, the `wrtomb()` function shall use its own internal **mbstate_t** object, which is
 70357 initialized at program start-up to the initial conversion state. Otherwise, the **mbstate_t** object
 70358 pointed to by *ps* shall be used to completely describe the current conversion state of the
 70359 associated character sequence. The implementation shall behave as if no function defined in this
 70360 volume of POSIX.1-2008 calls `wrtomb()`.

70361 CX The `wrtomb()` function need not be thread-safe if called with a NULL *ps* argument.70362 The behavior of this function shall be affected by the `LC_CTYPE` category of the current locale.70363 The `wrtomb()` function shall not change the setting of *errno* if successful.70364 **RETURN VALUE**

70365 The `wrtomb()` function shall return the number of bytes stored in the array object (including any
 70366 shift sequences). When *wc* is not a valid wide character, an encoding error shall occur. In this
 70367 case, the function shall store the value of the macro [EILSEQ] in *errno* and shall return (**size_t**)−1;
 70368 the conversion state shall be undefined.

70369 **ERRORS**70370 The `wrtomb()` function shall fail if:

70371 [EILSEQ] An invalid wide-character code is detected.

70372 The `wrtomb()` function may fail if:70373 CX [EINVAL] *ps* points to an object that contains an invalid conversion state.

70374 **EXAMPLES**

70375 None.

70376 **APPLICATION USAGE**

70377 None.

70378 **RATIONALE**

70379 None.

70380 **FUTURE DIRECTIONS**

70381 None.

70382 **SEE ALSO**70383 *mbsinit()*, *wcsrtombs()*

70384 XBD <wchar.h>

70385 **CHANGE HISTORY**70386 First released in Issue 5. Included for alignment with ISO/IEC 9899:1990/Amendment 1:1995
70387 (E).70388 **Issue 6**

70389 In the DESCRIPTION, a note on using this function in a threaded application is added.

70390 Extensions beyond the ISO C standard are marked.

70391 The normative text is updated to avoid use of the term “must” for application requirements.

70392 The *wcr tomb()* prototype is updated for alignment with the ISO/IEC 9899:1999 standard.70393 **Issue 7**70394 Austin Group Interpretation 1003.1-2001 #148 is applied, clarifying that the *wcr tomb()* function
70395 need not be thread-safe if called with a NULL *ps* argument.

70396 Austin Group Interpretation 1003.1-2001 #170 is applied.

70397 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0714 [88] and XSH/TC1-2008/0715 +
70398 [105] are applied.

NAME

`wcscasecmp`, `wcscasecmp_l`, `wcsncasecmp`, `wcsncasecmp_l` — case-insensitive wide-character string comparison

SYNOPSIS

```
CX #include <wchar.h>

int wcscasecmp(const wchar_t *ws1, const wchar_t *ws2);
int wcscasecmp_l(const wchar_t *ws1, const wchar_t *ws2,
    locale_t locale);
int wcsncasecmp(const wchar_t *ws1, const wchar_t *ws2, size_t n);
int wcsncasecmp_l(const wchar_t *ws1, const wchar_t *ws2,
    size_t n, locale_t locale);
```

DESCRIPTION

The `wcscasecmp()` and `wcsncasecmp()` functions are the wide-character equivalent of the `strcasecmp()` and `strncasecmp()` functions, respectively.

The `wcscasecmp()` and `wcscasecmp_l()` functions shall compare, while ignoring differences in case, the wide-character string pointed to by `ws1` to the wide-character string pointed to by `ws2`.

The `wcsncasecmp()` and `wcsncasecmp_l()` functions shall compare, while ignoring differences in case, not more than `n` wide-characters from the wide-character string pointed to by `ws1` to the wide-character string pointed to by `ws2`.

The `wcscasecmp()` and `wcsncasecmp()` functions use the current locale to determine the case of the wide characters. +

The `wcscasecmp_l()` and `wcsncasecmp_l()` functions use the locale represented by `locale` to determine the case of the wide characters. +

When the `LC_CTIME` category of the locale being used is from the POSIX locale, these functions shall behave as if the wide-character strings had been converted to lowercase and then a comparison of wide-character codes performed. Otherwise, the results are unspecified. |

The information for `wcscasecmp_l()` and `wcsncasecmp_l()` about the case of the characters comes from the locale represented by `locale`.

The behavior is undefined if the `locale` argument to `wcscasecmp_l()` or `wcsncasecmp_l()` is the special locale object `LC_GLOBAL_LOCALE` or is not a valid locale object handle. +

RETURN VALUE

Upon completion, the `wcscasecmp()` and `wcscasecmp_l()` functions shall return an integer greater than, equal to, or less than 0 if the wide-character string pointed to by `ws1` is, ignoring case, greater than, equal to, or less than the wide-character string pointed to by `ws2`, respectively.

Upon completion, the `wcsncasecmp()` and `wcsncasecmp_l()` functions shall return an integer greater than, equal to, or less than 0 if the possibly null wide-character terminated string pointed to by `ws1` is, ignoring case, greater than, equal to, or less than the possibly null wide-character terminated string pointed to by `ws2`, respectively.

No return values are reserved to indicate an error.

ERRORS

No errors are defined. |

70440 EXAMPLES

70441 None.

70442 APPLICATION USAGE

70443 None.

70444 RATIONALE

70445 None.

70446 FUTURE DIRECTIONS

70447 None.

70448 SEE ALSO

70449 [*strcasecmp\(\)*](#), [*wscmp\(\)*](#), [*wcsncmp\(\)*](#)

70450 XBD [**<wchar.h>**](#)

70451 CHANGE HISTORY

70452 First released in Issue 7.

70453 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0716 [294], XSH/TC1-2008/0717 [283], +
70454 and XSH/TC1-2008/0718 [283] are applied.

70455 **NAME**

70456 wscat — concatenate two wide-character strings

70457 **SYNOPSIS**

70458 #include <wchar.h>

70459 wchar_t *wscat(wchar_t *restrict *ws1*, const wchar_t *restrict *ws2*);70460 **DESCRIPTION**

70461 CX The functionality described on this reference page is aligned with the ISO C standard. Any
70462 conflict between the requirements described here and the ISO C standard is unintentional. This
70463 volume of POSIX.1-2008 defers to the ISO C standard.

70464 The *wscat()* function shall append a copy of the wide-character string pointed to by *ws2*
70465 (including the terminating null wide-character code) to the end of the wide-character string
70466 pointed to by *ws1*. The initial wide-character code of *ws2* shall overwrite the null wide-character
70467 code at the end of *ws1*. If copying takes place between objects that overlap, the behavior is
70468 undefined.

70469 **RETURN VALUE**70470 The *wscat()* function shall return *ws1*; no return value is reserved to indicate an error.70471 **ERRORS**

70472 No errors are defined.

70473 **EXAMPLES**

70474 None.

70475 **APPLICATION USAGE**

70476 None.

70477 **RATIONALE**

70478 None.

70479 **FUTURE DIRECTIONS**

70480 None.

70481 **SEE ALSO**70482 [*wcsncat\(\)*](#)70483 XBD [*<wchar.h>*](#)70484 **CHANGE HISTORY**

70485 First released in Issue 4. Derived from the MSE working draft.

70486 **Issue 6**

70487 The Open Group Corrigendum U040/2 is applied. In the RETURN VALUE section, *s1* is
70488 changed to *ws1*.

70489 The *wscat()* prototype is updated for alignment with the ISO/IEC 9899:1999 standard.

70490 **NAME**

70491 wcschr — wide-character string scanning operation

70492 **SYNOPSIS**

70493 #include <wchar.h>

70494 wchar_t *wcschr(const wchar_t *ws, wchar_t wc);

70495 **DESCRIPTION**

70496 **CX** The functionality described on this reference page is aligned with the ISO C standard. Any
70497 conflict between the requirements described here and the ISO C standard is unintentional. This
70498 volume of POSIX.1-2008 defers to the ISO C standard.

70499 The *wcschr()* function shall locate the first occurrence of *wc* in the wide-character string pointed
70500 to by *ws*. The application shall ensure that the value of *wc* is a character representable as a type
70501 **wchar_t** and a wide-character code corresponding to a valid character in the current locale. The
70502 terminating null wide-character code is considered to be part of the wide-character string.

70503 **RETURN VALUE**

70504 Upon completion, *wcschr()* shall return a pointer to the wide-character code, or a null pointer if
70505 the wide-character code is not found.

70506 **ERRORS**

70507 No errors are defined.

70508 **EXAMPLES**

70509 None.

70510 **APPLICATION USAGE**

70511 None.

70512 **RATIONALE**

70513 None.

70514 **FUTURE DIRECTIONS**

70515 None.

70516 **SEE ALSO**70517 *wcsrchr()*

70518 XBD <wchar.h>

70519 **CHANGE HISTORY**

70520 First released in Issue 4. Derived from the MSE working draft.

70521 **Issue 6**

70522 The normative text is updated to avoid use of the term “must” for application requirements.

70523 **NAME**70524 `wscmp` — compare two wide-character strings70525 **SYNOPSIS**70526 `#include <wchar.h>`70527 `int wscmp(const wchar_t *ws1, const wchar_t *ws2);`70528 **DESCRIPTION**

70529 CX The functionality described on this reference page is aligned with the ISO C standard. Any
70530 conflict between the requirements described here and the ISO C standard is unintentional. This
70531 volume of POSIX.1-2008 defers to the ISO C standard.

70532 The `wscmp()` function shall compare the wide-character string pointed to by `ws1` to the wide-
70533 character string pointed to by `ws2`.

70534 The sign of a non-zero return value shall be determined by the sign of the difference between the
70535 values of the first pair of wide-character codes that differ in the objects being compared.

70536 **RETURN VALUE**

70537 Upon completion, `wscmp()` shall return an integer greater than, equal to, or less than 0, if the
70538 wide-character string pointed to by `ws1` is greater than, equal to, or less than the wide-character
70539 string pointed to by `ws2`, respectively.

70540 **ERRORS**

70541 No errors are defined.

70542 **EXAMPLES**

70543 None.

70544 **APPLICATION USAGE**

70545 None.

70546 **RATIONALE**

70547 None.

70548 **FUTURE DIRECTIONS**

70549 None.

70550 **SEE ALSO**70551 [`wscasecmp\(\)`](#), [`wcsncmp\(\)`](#)70552 XBD [`<wchar.h>`](#)70553 **CHANGE HISTORY**

70554 First released in Issue 4. Derived from the MSE working draft.

NAME

wscoll, wscoll_l — wide-character string comparison using collating information

SYNOPSIS

```
#include <wchar.h>
```

```
int wscoll(const wchar_t *ws1, const wchar_t *ws2);
```

```
CX int wscoll_l(const wchar_t *ws1, const wchar_t *ws2,  
    locale_t locale);
```

DESCRIPTION

CX For *wscoll()*: The functionality described on this reference page is aligned with the ISO C standard. Any conflict between the requirements described here and the ISO C standard is unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

CX The *wscoll()* and *wscoll_l()* functions shall compare the wide-character string pointed to by *ws1* to the wide-character string pointed to by *ws2*, both interpreted as appropriate to the LC_COLLATE category of the current locale, or the locale represented by *locale*, respectively.

CX The *wscoll()* and *wscoll_l()* functions shall not change the setting of *errno* if successful.

CX An application wishing to check for error situations should set *errno* to 0 before calling *wscoll()* or *wscoll_l()*. If *errno* is non-zero on return, an error has occurred.

CX The behavior is undefined if the *locale* argument to *wscoll_l()* is the special locale object LC_GLOBAL_LOCALE or is not a valid locale object handle.

RETURN VALUE

CX Upon successful completion, *wscoll()* and *wscoll_l()* shall return an integer greater than, equal to, or less than 0, according to whether the wide-character string pointed to by *ws1* is greater than, equal to, or less than the wide-character string pointed to by *ws2*, when both are interpreted as appropriate to the current locale, or to the locale represented by *locale*, respectively. On error, *wscoll()* and *wscoll_l()* shall set *errno*, but no return value is reserved to indicate an error.

ERRORS

These functions may fail if:

CX [EINVAL] The *ws1* or *ws2* arguments contain wide-character codes outside the domain of the collating sequence.

EXAMPLES

None.

APPLICATION USAGE

The *wcsxfrm()* and *wscmp()* functions should be used for sorting large lists.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

wscmp(), *wcsxfrm()*

XBD <wchar.h>

70596 **CHANGE HISTORY**

70597 First released in Issue 4. Derived from the MSE working draft.

70598 **Issue 5**

70599 Moved from ENHANCED I18N to BASE and the [ENOSYS] error is removed.

70600 The DESCRIPTION is updated to indicate that *errno* is not changed if the function is successful.

70601 **Issue 7**

70602 The *wcscoll_l()* function is added from The Open Group Technical Standard, 2006, Extended API
70603 Set Part 4.

70604 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0719 [302], XSH/TC1-2008/0720 [283], +
70605 and XSH/TC1-2008/0721 [283] are applied.

NAME

wcpcpy, wcscopy — copy a wide-character string, returning a pointer to its end

SYNOPSIS

```
#include <wchar.h>
```

```
CX      wchar_t *wcpcpy(wchar_t *restrict ws1, const wchar_t *restrict ws2);
        wchar_t *wcscopy(wchar_t *restrict ws1, const wchar_t *restrict ws2);
```

DESCRIPTION

CX For *wcscopy()*: The functionality described on this reference page is aligned with the ISO C standard. Any conflict between the requirements described here and the ISO C standard is unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

CX The *wcpcpy()* and *wcscopy()* functions shall copy the wide-character string pointed to by *ws2* (including the terminating null wide-character code) into the array pointed to by *ws1*.

The application shall ensure that there is room for at least *wcslen(ws2)+1* wide characters in the *ws1* array, and that the *ws2* and *ws1* arrays do not overlap.

If copying takes place between objects that overlap, the behavior is undefined.

RETURN VALUE

CX The *wcpcpy()* function shall return a pointer to the terminating null wide-character code copied into the *ws1* buffer.

The *wcscopy()* function shall return *ws1*.

No return values are reserved to indicate an error.

ERRORS

No errors are defined.

EXAMPLES

None.

APPLICATION USAGE

None.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

[*strcpy\(\)*](#), [*wcsdup\(\)*](#), [*wcsncpy\(\)*](#)

XBD [**<wchar.h>**](#)

CHANGE HISTORY

First released in Issue 4. Derived from the MSE working draft.

Issue 6

The *wcscopy()* prototype is updated for alignment with the ISO/IEC 9899:1999 standard.

Issue 7

The *wcpcpy()* function is added from The Open Group Technical Standard, 2006, Extended API Set Part 1.

70646 **NAME**

70647 wcscspn — get the length of a complementary wide substring

70648 **SYNOPSIS**

70649 #include <wchar.h>

70650 size_t wcscspn(const wchar_t *ws1, const wchar_t *ws2);

70651 **DESCRIPTION**

70652 CX The functionality described on this reference page is aligned with the ISO C standard. Any
70653 conflict between the requirements described here and the ISO C standard is unintentional. This
70654 volume of POSIX.1-2008 defers to the ISO C standard.

70655 The *wcscspn()* function shall compute the length (in wide characters) of the maximum initial
70656 segment of the wide-character string pointed to by *ws1* which consists entirely of wide-character
70657 codes *not* from the wide-character string pointed to by *ws2*.

70658 **RETURN VALUE**

70659 The *wcscspn()* function shall return the length of the initial substring of *ws1*; no return value is
70660 reserved to indicate an error.

70661 **ERRORS**

70662 No errors are defined.

70663 **EXAMPLES**

70664 None.

70665 **APPLICATION USAGE**

70666 None.

70667 **RATIONALE**

70668 None.

70669 **FUTURE DIRECTIONS**

70670 None.

70671 **SEE ALSO**70672 [*wcssp\(\)*](#)70673 XBD [**<wchar.h>**](#)70674 **CHANGE HISTORY**

70675 First released in Issue 4. Derived from the MSE working draft.

70676 **Issue 5**

70677 The RETURN VALUE section is updated to indicate that *wcscspn()* returns the length of *ws1*,
70678 rather than *ws1* itself.

NAME

wcsdup — duplicate a wide-character string

SYNOPSIS

```
CX      #include <wchar.h>
      wchar_t *wcsdup(const wchar_t *string);
```

DESCRIPTION

The *wcsdup()* function is the wide-character equivalent of the *strdup()* function.

The *wcsdup()* function shall return a pointer to a new wide-character string, allocated as if by a call to *malloc()*, which is the duplicate of the wide-character string *string*. The returned pointer can be passed to *free()*. A null pointer is returned if the new wide-character string cannot be created.

RETURN VALUE

Upon successful completion, the *wcsdup()* function shall return a pointer to the newly allocated wide-character string. Otherwise, it shall return a null pointer and set *errno* to indicate the error.

ERRORS

The *wcsdup()* function shall fail if:

[ENOMEM] Memory large enough for the duplicate string could not be allocated.

EXAMPLES

None.

APPLICATION USAGE

For functions that allocate memory as if by *malloc()*, the application should release such memory when it is no longer required by a call to *free()*. For *wcsdup()*, this is the return value.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

free(), *strdup()*, *wscpy()*

XBD <wchar.h>

CHANGE HISTORY

First released in Issue 7.

NAME

wcsftime — convert date and time to a wide-character string

SYNOPSIS

```
#include <wchar.h>
```

```
size_t wcsftime(wchar_t *restrict wcs, size_t maxsize,
                const wchar_t *restrict format, const struct tm *restrict timeptr);
```

DESCRIPTION

CX The functionality described on this reference page is aligned with the ISO C standard. Any conflict between the requirements described here and the ISO C standard is unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

The *wcsftime()* function shall be equivalent to the *strftime()* function, except that:

- The argument *wcs* points to the initial element of an array of wide characters into which the generated output is to be placed.
- The argument *maxsize* indicates the maximum number of wide characters to be placed in the output array.
- The argument *format* is a wide-character string and the conversion specifications are replaced by corresponding sequences of wide characters.
- The return value indicates the number of wide characters placed in the output array.

If copying takes place between objects that overlap, the behavior is undefined.

RETURN VALUE

If the total number of resulting wide-character codes including the terminating null wide-character code is no more than *maxsize*, *wcsftime()* shall return the number of wide-character codes placed into the array pointed to by *wcs*, not including the terminating null wide-character code. Otherwise, zero is returned and the contents of the array are unspecified.

ERRORS

No errors are defined.

EXAMPLES

None.

APPLICATION USAGE

None.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

[*strftime\(\)*](#)

XBD [*<wchar.h>*](#)

CHANGE HISTORY

First released in Issue 4.

70749 Issue 5

70750 Moved from ENHANCED I18N to BASE and the [ENOSYS] error is removed.

70751 Aligned with ISO/IEC 9899:1990/Amendment 1:1995 (E). Specifically, the type of the *format*
70752 argument is changed from **const char *** to **const wchar_t ***.

70753 Issue 6

70754 The *wcsftime()* prototype is updated for alignment with the ISO/IEC 9899:1999 standard.

70755 **NAME**

70756 wcslen, wcsnlen — get length of a fixed-sized wide-character string

70757 **SYNOPSIS**

70758 #include <wchar.h>

70759 size_t wcslen(const wchar_t *ws);

70760 CX size_t wcsnlen(const wchar_t *ws, size_t maxlen);

70761 **DESCRIPTION**

70762 CX For *wcslen()*: The functionality described on this reference page is aligned with the ISO C
 70763 standard. Any conflict between the requirements described here and the ISO C standard is
 70764 unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

70765 The *wcslen()* function shall compute the number of wide-character codes in the wide-character
 70766 string to which *ws* points, not including the terminating null wide-character code.

70767 CX The *wcsnlen()* function shall compute the smaller of the number of wide characters in the string
 70768 to which *ws* points, not including the terminating null wide-character code, and the value of
 70769 *maxlen*. The *wcsnlen()* function shall never examine more than the first *maxlen* characters of the
 70770 wide-character string pointed to by *ws*.

70771 **RETURN VALUE**70772 The *wcslen()* function shall return the length of *ws*.

70773 CX The *wcsnlen()* function shall return an integer containing the smaller of either the length of the
 70774 wide-character string pointed to by *ws* or *maxlen*.

70775 No return values are reserved to indicate an error.

70776 **ERRORS**

70777 No errors are defined.

70778 **EXAMPLES**

70779 None.

70780 **APPLICATION USAGE**

70781 None.

70782 **RATIONALE**

70783 None.

70784 **FUTURE DIRECTIONS**

70785 None.

70786 **SEE ALSO**70787 [*strlen\(\)*](#)70788 XBD [*<wchar.h>*](#)70789 **CHANGE HISTORY**

70790 First released in Issue 4. Derived from the MSE working draft.

70791 **Issue 7**

70792 The *wcsnlen()* function is added from The Open Group Technical Standard, 2006, Extended API
 70793 Set Part 1.

70794 **NAME**

70795 wcsncasecmp, wcsncasecmp_l — case-insensitive wide-character string comparison

70796 **SYNOPSIS**

```
70797 CX      #include <wchar.h>
70798          int wcsncasecmp(const wchar_t *ws1, const wchar_t *ws2, size_t n);
70799          int wcsncasecmp_l(const wchar_t *ws1, const wchar_t *ws2,
70800                          size_t n, locale_t locale);
```

70801 **DESCRIPTION**70802 Refer to *wcscasecmp()*.

70803 **NAME**70804 `wcsncat` — concatenate a wide-character string with part of another70805 **SYNOPSIS**70806 `#include <wchar.h>`70807 `wchar_t *wcsncat(wchar_t *restrict ws1, const wchar_t *restrict ws2,`
70808 `size_t n);`70809 **DESCRIPTION**70810 **CX** The functionality described on this reference page is aligned with the ISO C standard. Any
70811 conflict between the requirements described here and the ISO C standard is unintentional. This
70812 volume of POSIX.1-2008 defers to the ISO C standard.70813 The `wcsncat()` function shall append not more than *n* wide-character codes (a null wide-
70814 character code and wide-character codes that follow it are not appended) from the array pointed
70815 to by *ws2* to the end of the wide-character string pointed to by *ws1*. The initial wide-character
70816 code of *ws2* shall overwrite the null wide-character code at the end of *ws1*. A terminating null
70817 wide-character code shall always be appended to the result. If copying takes place between
70818 objects that overlap, the behavior is undefined.70819 **RETURN VALUE**70820 The `wcsncat()` function shall return *ws1*; no return value is reserved to indicate an error.70821 **ERRORS**

70822 No errors are defined.

70823 **EXAMPLES**

70824 None.

70825 **APPLICATION USAGE**

70826 None.

70827 **RATIONALE**

70828 None.

70829 **FUTURE DIRECTIONS**

70830 None.

70831 **SEE ALSO**70832 [*wcscat\(\)*](#)70833 XBD [*<wchar.h>*](#)70834 **CHANGE HISTORY**

70835 First released in Issue 4. Derived from the MSE working draft.

70836 **Issue 6**70837 The `wcsncat()` prototype is updated for alignment with the ISO/IEC 9899:1999 standard.

NAME

wcsncmp — compare part of two wide-character strings

SYNOPSIS

```
#include <wchar.h>
```

```
int wcsncmp(const wchar_t *ws1, const wchar_t *ws2, size_t n);
```

DESCRIPTION

CX The functionality described on this reference page is aligned with the ISO C standard. Any conflict between the requirements described here and the ISO C standard is unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

The *wcsncmp()* function shall compare not more than *n* wide-character codes (wide-character codes that follow a null wide-character code are not compared) from the array pointed to by *ws1* to the array pointed to by *ws2*.

The sign of a non-zero return value shall be determined by the sign of the difference between the values of the first pair of wide-character codes that differ in the objects being compared.

RETURN VALUE

Upon successful completion, *wcsncmp()* shall return an integer greater than, equal to, or less than 0, if the possibly null-terminated array pointed to by *ws1* is greater than, equal to, or less than the possibly null-terminated array pointed to by *ws2*, respectively.

ERRORS

No errors are defined.

EXAMPLES

None.

APPLICATION USAGE

None.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

[*wscasecmp\(\)*](#), [*wscmp\(\)*](#)

XBD [*<wchar.h>*](#)

CHANGE HISTORY

First released in Issue 4. Derived from the MSE working draft.

70871 **NAME**

70872 wcpncpy, wcsncpy — copy a fixed-size wide-character string, returning a pointer to its end

70873 **SYNOPSIS**

70874 #include <wchar.h>

70875 CX wchar_t *wcpncpy(wchar_t restrict *ws1, const wchar_t *restrict ws2,
70876 size_t n);70877 wchar_t *wcsncpy(wchar_t *restrict ws1, const wchar_t *restrict ws2,
70878 size_t n);70879 **DESCRIPTION**70880 CX For *wcsncpy()*: The functionality described on this reference page is aligned with the ISO C
70881 standard. Any conflict between the requirements described here and the ISO C standard is
70882 unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.70883 CX The *wcpncpy()* and *wcsncpy()* functions shall copy not more than *n* wide-character codes (wide-
70884 character codes that follow a null wide-character code are not copied) from the array pointed to
70885 by *ws2* to the array pointed to by *ws1*. If copying takes place between objects that overlap, the
70886 behavior is undefined.70887 If the array pointed to by *ws2* is a wide-character string that is shorter than *n* wide-character
70888 codes, null wide-character codes shall be appended to the copy in the array pointed to by *ws1*,
70889 until *n* wide-character codes in all are written.70890 **RETURN VALUE**70891 CX If any null wide-character codes were written into the destination, the *wcpncpy()* function shall
70892 return the address of the first such null wide-character code. Otherwise, it shall return *&ws1[n]*.70893 The *wcsncpy()* function shall return *ws1*.

70894 No return values are reserved to indicate an error.

70895 **ERRORS**

70896 No errors are defined.

70897 **EXAMPLES**

70898 None.

70899 **APPLICATION USAGE**70900 If there is no null wide-character code in the first *n* wide-character codes of the array pointed to
70901 by *ws2*, the result is not null-terminated.70902 **RATIONALE**

70903 None.

70904 **FUTURE DIRECTIONS**

70905 None.

70906 **SEE ALSO**70907 *strncpy()*, *wcscpy()*

70908 XBD <wchar.h>

70909 **CHANGE HISTORY**

70910 First released in Issue 4. Derived from the MSE working draft.

Issue 6

70911 The *wcsncpy()* prototype is updated for alignment with the ISO/IEC 9899:1999 standard.

Issue 7

70913 The *wcpncpy()* function is added from The Open Group Technical Standard, 2006, Extended API
70914 Set Part 1.
70915

70916 **NAME**

70917 wcsnlen — get length of a fixed-sized wide-character string

70918 **SYNOPSIS**

```
70919 CX       #include <wchar.h>  
70920       size_t wcsnlen(const wchar_t *ws, size_t maxlen);
```

70921 **DESCRIPTION**70922 Refer to *wcslen()*.

70923 **NAME**

70924 wcsnrtombs — convert wide-character string to multi-byte string

70925 **SYNOPSIS**

```
70926 CX      #include <wchar.h>
70927          size_t wcsnrtombs(char *restrict dst, const wchar_t **restrict src,
70928                          size_t nwc, size_t len, mbstate_t *restrict ps);
```

70929 **DESCRIPTION**70930 Refer to *wcsrtombs()*.

70931 **NAME**

70932 wcpbrk — scan a wide-character string for a wide-character code

70933 **SYNOPSIS**

70934 #include <wchar.h>

70935 wchar_t *wcpbrk(const wchar_t *ws1, const wchar_t *ws2);

70936 **DESCRIPTION**

70937 CX The functionality described on this reference page is aligned with the ISO C standard. Any
70938 conflict between the requirements described here and the ISO C standard is unintentional. This
70939 volume of POSIX.1-2008 defers to the ISO C standard.

70940 The *wcpbrk()* function shall locate the first occurrence in the wide-character string pointed to by
70941 *ws1* of any wide-character code from the wide-character string pointed to by *ws2*.

70942 **RETURN VALUE**

70943 Upon successful completion, *wcpbrk()* shall return a pointer to the wide-character code or a null
70944 pointer if no wide-character code from *ws2* occurs in *ws1*.

70945 **ERRORS**

70946 No errors are defined.

70947 **EXAMPLES**

70948 None.

70949 **APPLICATION USAGE**

70950 None.

70951 **RATIONALE**

70952 None.

70953 **FUTURE DIRECTIONS**

70954 None.

70955 **SEE ALSO**70956 *wcschr()*, *wcsrchr()*

70957 XBD <wchar.h>

70958 **CHANGE HISTORY**

70959 First released in Issue 4. Derived from the MSE working draft.

70960 **NAME**

70961 wcsrchr — wide-character string scanning operation

70962 **SYNOPSIS**

70963 #include <wchar.h>

70964 wchar_t *wcsrchr(const wchar_t *ws, wchar_t wc);

70965 **DESCRIPTION**

70966 **CX** The functionality described on this reference page is aligned with the ISO C standard. Any
70967 conflict between the requirements described here and the ISO C standard is unintentional. This
70968 volume of POSIX.1-2008 defers to the ISO C standard.

70969 The *wcsrchr()* function shall locate the last occurrence of *wc* in the wide-character string pointed
70970 to by *ws*. The application shall ensure that the value of *wc* is a character representable as a type
70971 **wchar_t** and a wide-character code corresponding to a valid character in the current locale. The
70972 terminating null wide-character code shall be considered to be part of the wide-character string.

70973 **RETURN VALUE**

70974 Upon successful completion, *wcsrchr()* shall return a pointer to the wide-character code or a null
70975 pointer if *wc* does not occur in the wide-character string.

70976 **ERRORS**

70977 No errors are defined.

70978 **EXAMPLES**

70979 None.

70980 **APPLICATION USAGE**

70981 None.

70982 **RATIONALE**

70983 None.

70984 **FUTURE DIRECTIONS**

70985 None.

70986 **SEE ALSO**70987 *wcschr()*

70988 XBD <wchar.h>

70989 **CHANGE HISTORY**

70990 First released in Issue 4. Derived from the MSE working draft.

70991 **Issue 6**

70992 The normative text is updated to avoid use of the term “must” for application requirements.

70993 NAME

70994 wcsnrtombs, wcsrtombs — convert a wide-character string to a character string (restartable)

70995 SYNOPSIS

70996 #include <wchar.h>

```
70997 CX size_t wcsnrtombs(char *restrict dst, const wchar_t **restrict src,
70998 size_t nwc, size_t len, mbstate_t *restrict ps);
70999 size_t wcsrtombs(char *restrict dst, const wchar_t **restrict src,
71000 size_t len, mbstate_t *restrict ps);
```

71001 DESCRIPTION

71002 CX For *wcsrtombs()*: The functionality described on this reference page is aligned with the ISO C
 71003 standard. Any conflict between the requirements described here and the ISO C standard is
 71004 unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

71005 The *wcsrtombs()* function shall convert a sequence of wide characters from the array indirectly
 71006 pointed to by *src* into a sequence of corresponding characters, beginning in the conversion state
 71007 described by the object pointed to by *ps*. If *dst* is not a null pointer, the converted characters
 71008 shall then be stored into the array pointed to by *dst*. Conversion continues up to and including a
 71009 terminating null wide character, which shall also be stored. Conversion shall stop earlier in the
 71010 following cases:

- 71011 • When a code is reached that does not correspond to a valid character
- 71012 • When the next character would exceed the limit of *len* total bytes to be stored in the array
- 71013 pointed to by *dst* (and *dst* is not a null pointer)

71014 Each conversion shall take place as if by a call to the *wcrtomb()* function.

71015 If *dst* is not a null pointer, the pointer object pointed to by *src* shall be assigned either a null
 71016 pointer (if conversion stopped due to reaching a terminating null wide character) or the address
 71017 just past the last wide character converted (if any). If conversion stopped due to reaching a
 71018 terminating null wide character, the resulting state described shall be the initial conversion state.

71019 If *ps* is a null pointer, the *wcsrtombs()* function shall use its own internal **mbstate_t** object, which
 71020 is initialized at program start-up to the initial conversion state. Otherwise, the **mbstate_t** object
 71021 pointed to by *ps* shall be used to completely describe the current conversion state of the
 71022 associated character sequence.

71023 CX The *wcsnrtombs()* and *wcsrtombs()* functions need not be thread-safe if called with a NULL *ps* |
 71024 argument.

71025 The *wcsnrtombs()* function shall be equivalent to the *wcsrtombs()* function, except that the
 71026 conversion is limited to the first *nwc* wide characters.

71027 The *wcsrtombs()* function shall not change the setting of *errno* if successful. +

71028 The behavior of these functions shall be affected by the *LC_CTYPE* category of the current locale.

71029 The implementation shall behave as if no function defined in System Interfaces volume of
 71030 POSIX.1-2008 calls these functions.

71031 RETURN VALUE

71032 If conversion stops because a code is reached that does not correspond to a valid character, an
 71033 encoding error occurs. In this case, these functions shall store the value of the macro [EILSEQ] in
 71034 *errno* and return (**size_t**)–1; the conversion state is undefined. Otherwise, these functions shall
 71035 return the number of bytes in the resulting character sequence, not including the terminating
 71036 null (if any).

71037 **ERRORS**

71038 These functions shall fail if:

71039 [EILSEQ] A wide-character code does not correspond to a valid character.

71040 These functions may fail if:

71041 CX [EINVAL] *ps* points to an object that contains an invalid conversion state.71042 **EXAMPLES**

71043 None.

71044 **APPLICATION USAGE**

71045 None.

71046 **RATIONALE**

71047 None.

71048 **FUTURE DIRECTIONS**

71049 None.

71050 **SEE ALSO**71051 *mbsinit()*, *wcrtomb()*

71052 XBD <wchar.h>

71053 **CHANGE HISTORY**71054 First released in Issue 5. Included for alignment with ISO/IEC 9899:1990/Amendment 1:1995
71055 (E).71056 **Issue 6**

71057 In the DESCRIPTION, a note on using this function in a threaded application is added.

71058 Extensions beyond the ISO C standard are marked.

71059 The normative text is updated to avoid use of the term “must” for application requirements.

71060 The *wcsrtombs()* prototype is updated for alignment with the ISO/IEC 9899:1999 standard.71061 **Issue 7**71062 Austin Group Interpretation 1003.1-2001 #148 is applied, clarifying that the *wcsrtombs()* function
71063 need not be thread-safe if called with a NULL *ps* argument.

71064 Austin Group Interpretation 1003.1-2001 #170 is applied.

71065 The *wcnsrtombs()* function is added from The Open Group Technical Standard, 2006, Extended
71066 API Set Part 1.

71067 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0722 [109,105] is applied.

+

71068 **NAME**

71069 wcsspnp — get the length of a wide substring

71070 **SYNOPSIS**

71071 #include <wchar.h>

71072 size_t wcsspnp(const wchar_t *ws1, const wchar_t *ws2);

71073 **DESCRIPTION**

71074 **CX** The functionality described on this reference page is aligned with the ISO C standard. Any
71075 conflict between the requirements described here and the ISO C standard is unintentional. This
71076 volume of POSIX.1-2008 defers to the ISO C standard.

71077 The *wcsspnp()* function shall compute the length (in wide characters) of the maximum initial
71078 segment of the wide-character string pointed to by *ws1* which consists entirely of wide-character
71079 codes from the wide-character string pointed to by *ws2*.

71080 **RETURN VALUE**

71081 The *wcsspnp()* function shall return the length of the initial substring of *ws1*; no return value is
71082 reserved to indicate an error.

71083 **ERRORS**

71084 No errors are defined.

71085 **EXAMPLES**

71086 None.

71087 **APPLICATION USAGE**

71088 None.

71089 **RATIONALE**

71090 None.

71091 **FUTURE DIRECTIONS**

71092 None.

71093 **SEE ALSO**71094 *wcscspnp()*

71095 XBD <wchar.h>

71096 **CHANGE HISTORY**

71097 First released in Issue 4. Derived from the MSE working draft.

71098 **Issue 5**

71099 The RETURN VALUE section is updated to indicate that *wcsspnp()* returns the length of *ws1*
71100 rather than *ws1* itself.

71101 **NAME**

71102 wcsstr — find a wide-character substring

71103 **SYNOPSIS**

71104 #include <wchar.h>

71105 wchar_t *wcsstr(const wchar_t *restrict ws1,
71106 const wchar_t *restrict ws2);71107 **DESCRIPTION**71108 CX The functionality described on this reference page is aligned with the ISO C standard. Any
71109 conflict between the requirements described here and the ISO C standard is unintentional. This
71110 volume of POSIX.1-2008 defers to the ISO C standard.71111 The *wcsstr()* function shall locate the first occurrence in the wide-character string pointed to by
71112 *ws1* of the sequence of wide characters (excluding the terminating null wide character) in the
71113 wide-character string pointed to by *ws2*.71114 **RETURN VALUE**71115 Upon successful completion, *wcsstr()* shall return a pointer to the located wide-character string,
71116 or a null pointer if the wide-character string is not found.71117 If *ws2* points to a wide-character string with zero length, the function shall return *ws1*.71118 **ERRORS**

71119 No errors are defined.

71120 **EXAMPLES**

71121 None.

71122 **APPLICATION USAGE**

71123 None.

71124 **RATIONALE**

71125 None.

71126 **FUTURE DIRECTIONS**

71127 None.

71128 **SEE ALSO**71129 [*wcschr\(\)*](#)71130 XBD [*<wchar.h>*](#)71131 **CHANGE HISTORY**71132 First released in Issue 5. Included for alignment with ISO/IEC 9899:1990/Amendment 1:1995
71133 (E).71134 **Issue 6**71135 The *wcsstr()* prototype is updated for alignment with the ISO/IEC 9899:1999 standard.

71136 NAME

71137 wcstod, wcstof, wcstold — convert a wide-character string to a double-precision number

71138 SYNOPSIS

71139 #include <wchar.h>

71140 double wcstod(const wchar_t *restrict nptr, wchar_t **restrict endptr);

71141 float wcstof(const wchar_t *restrict nptr, wchar_t **restrict endptr);

71142 long double wcstold(const wchar_t *restrict nptr,

71143 wchar_t **restrict endptr);

71144 DESCRIPTION

71145 CX The functionality described on this reference page is aligned with the ISO C standard. Any
 71146 conflict between the requirements described here and the ISO C standard is unintentional. This
 71147 volume of POSIX.1-2008 defers to the ISO C standard.

71148 These functions shall convert the initial portion of the wide-character string pointed to by *nptr* to
 71149 **double**, **float**, and **long double** representation, respectively. First, they shall decompose the
 71150 input wide-character string into three parts:

- 71151 1. An initial, possibly empty, sequence of white-space wide-character codes (as specified by
 71152 *iswspace()*)
- 71153 2. A subject sequence interpreted as a floating-point constant or representing infinity or
 71154 NaN
- 71155 3. A final wide-character string of one or more unrecognized wide-character codes,
 71156 including the terminating null wide-character code of the input wide-character string

71157 Then they shall attempt to convert the subject sequence to a floating-point number, and return
 71158 the result.

71159 The expected form of the subject sequence is an optional '+' or '-' sign, then one of the
 71160 following:

- 71161 • A non-empty sequence of decimal digits optionally containing a radix character; then an
 71162 optional exponent part consisting of the wide character 'e' or the wide character 'E',
 71163 optionally followed by a '+' or '-' wide character, and then followed by one or more
 71164 decimal digits
- 71165 • A 0x or 0X, then a non-empty sequence of hexadecimal digits optionally containing a radix
 71166 character; then an optional binary exponent part consisting of the wide character 'p' or
 71167 the wide character 'P', optionally followed by a '+' or '-' wide character, and then
 71168 followed by one or more decimal digits
- 71169 • One of INF or INFINITY, or any other wide string equivalent except for case
- 71170 • One of NAN or NAN(*n-wchar-sequence_{opt}*), or any other wide string ignoring case in the
 71171 NAN part, where:

71172 n-wchar-sequence:

71173 digit

71174 nondigit

71175 n-wchar-sequence digit

71176 n-wchar-sequence nondigit

71177 The subject sequence is defined as the longest initial subsequence of the input wide string,
 71178 starting with the first non-white-space wide character, that is of the expected form. The subject
 71179 sequence contains no wide characters if the input wide string is not of the expected form.

If the subject sequence has the expected form for a floating-point number, the sequence of wide characters starting with the first digit or the radix character (whichever occurs first) shall be interpreted as a floating constant according to the rules of the C language, except that the radix character shall be used in place of a period, and that if neither an exponent part nor a radix character appears in a decimal floating-point number, or if a binary exponent part does not appear in a hexadecimal floating-point number, an exponent part of the appropriate type with value zero shall be assumed to follow the last digit in the string. If the subject sequence begins with a minus-sign, the sequence shall be interpreted as negated. A wide-character sequence INF or INFINITY shall be interpreted as an infinity, if representable in the return type, else as if it were a floating constant that is too large for the range of the return type. A wide-character sequence NAN or NAN(*n-wchar-sequence_{opt}*) shall be interpreted as a quiet NaN, if supported in the return type, else as if it were a subject sequence part that does not have the expected form; the meaning of the *n-wchar* sequences is implementation-defined. A pointer to the final wide string shall be stored in the object pointed to by *endptr*, provided that *endptr* is not a null pointer.

If the subject sequence has the hexadecimal form and FLT_RADIX is a power of 2, the conversion shall be rounded in an implementation-defined manner.

CX The radix character shall be as defined in the current locale (category *LC_NUMERIC*). In the POSIX locale, or in a locale where the radix character is not defined, the radix character shall default to a <period> ('.').

CX In other than the C or POSIX locales, other implementation-defined subject sequences may be accepted.

If the subject sequence is empty or does not have the expected form, no conversion shall be performed; the value of *nptr* shall be stored in the object pointed to by *endptr*, provided that *endptr* is not a null pointer.

These functions shall not change the setting of *errno* if successful.

Since 0 is returned on error and is also a valid return on success, an application wishing to check for error situations should set *errno* to 0, then call *wcstod()*, *wcstof()*, or *wcstold()*, then check *errno*.

71208 RETURN VALUE

Upon successful completion, these functions shall return the converted value. If no conversion could be performed, 0 shall be returned and *errno* may be set to [EINVAL].

If the correct value is outside the range of representable values, $\pm$ HUGE_VAL, $\pm$ HUGE_VALF, or $\pm$ HUGE_VALL shall be returned (according to the sign of the value), and *errno* shall be set to [ERANGE].

If the correct value would cause underflow, a value whose magnitude is no greater than the smallest normalized positive number in the return type shall be returned and *errno* set to [ERANGE].

71217 ERRORS

The *wcstod()* function shall fail if:

[ERANGE] The value to be returned would cause overflow or underflow.

The *wcstod()* function may fail if:

CX [EINVAL] No conversion could be performed.

EXAMPLES

None.

APPLICATION USAGE

If the subject sequence has the hexadecimal form and FLT_RADIX is not a power of 2, and the result is not exactly representable, the result should be one of the two numbers in the appropriate internal format that are adjacent to the hexadecimal floating source value, with the extra stipulation that the error should have a correct sign for the current rounding direction.

If the subject sequence has the decimal form and at most DECIMAL_DIG (defined in `<float.h>`) significant digits, the result should be correctly rounded. If the subject sequence *D* has the decimal form and more than DECIMAL_DIG significant digits, consider the two bounding, adjacent decimal strings *L* and *U*, both having DECIMAL_DIG significant digits, such that the values of *L*, *D*, and *U* satisfy "*L* <= *D* <= *U*". The result should be one of the (equal or adjacent) values that would be obtained by correctly rounding *L* and *U* according to the current rounding direction, with the extra stipulation that the error with respect to *D* should have a correct sign for the current rounding direction.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

[*fscanf\(\)*](#), [*iswspace\(\)*](#), [*localeconv\(\)*](#), [*setlocale\(\)*](#), [*wcstol\(\)*](#)

XBD [Chapter 7](#) (on page 135), `<float.h>`, `<wchar.h>`

CHANGE HISTORY

First released in Issue 4. Derived from the MSE working draft.

Issue 5

The DESCRIPTION is updated to indicate that *errno* is not changed if the function is successful.

Issue 6

Extensions beyond the ISO C standard are marked.

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- In the RETURN VALUE and ERRORS sections, the [EINVAL] optional error condition is added if no conversion could be performed.

The following changes are made for alignment with the ISO/IEC 9899:1999 standard:

- The *wcstod()* prototype is updated.
- The *wcstof()* and *wcstold()* functions are added.
- If the correct value for *wcstod()* would cause underflow, the return value changed from 0 (as specified in Issue 5) to the smallest normalized positive number.
- The DESCRIPTION, RETURN VALUE, and APPLICATION USAGE sections are extensively updated.

ISO/IEC 9899:1999 standard, Technical Corrigendum 1 is incorporated.

IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/66 is applied, correcting the second paragraph in the RETURN VALUE section.

Issue 7

71265 Austin Group Interpretation 1003.1-2001 #015 is applied.

71266 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0723 [302] and XSH/TC1-2008/0724 +
71267 [105] are applied.

71268 **NAME**

71269 wcstoimax, wcstoumax — convert a wide-character string to an integer type

71270 **SYNOPSIS**

```
71271       #include <stddef.h>
71272       #include <inttypes.h>

71273       intmax_t wcstoimax(const wchar_t *restrict nptr,
71274                          wchar_t **restrict endptr, int base);
71275       uintmax_t wcstoumax(const wchar_t *restrict nptr,
71276                          wchar_t **restrict endptr, int base);
```

71277 **DESCRIPTION**

71278 **CX** The functionality described on this reference page is aligned with the ISO C standard. Any
 71279 conflict between the requirements described here and the ISO C standard is unintentional. This
 71280 volume of POSIX.1-2008 defers to the ISO C standard.

71281 These functions shall be equivalent to the *wcstol()*, *wcstoll()*, *wcstoul()*, and *wcstoull()* functions,
 71282 respectively, except that the initial portion of the wide string shall be converted to **intmax_t** and
 71283 **uintmax_t** representation, respectively.

71284 **RETURN VALUE**

71285 These functions shall return the converted value, if any.

71286 If no conversion could be performed, zero shall be returned. If the correct value is outside the
 71287 range of representable values, {INTMAX_MAX}, {INTMAX_MIN}, or {UINTMAX_MAX} shall
 71288 be returned (according to the return type and sign of the value, if any), and *errno* shall be set to
 71289 [ERANGE].

71290 **ERRORS**

71291 These functions shall fail if:

71292 [EINVAL] The value of *base* is not supported.

71293 [ERANGE] The value to be returned is not representable.

71294 These functions may fail if:

71295 [EINVAL] No conversion could be performed.

71296 **EXAMPLES**

71297 None.

71298 **APPLICATION USAGE**

71299 None.

71300 **RATIONALE**

71301 None.

71302 **FUTURE DIRECTIONS**

71303 None.

71304 **SEE ALSO**71305 *wcstol()*, *wcstoul()*

71306 XBD <inttypes.h>, <stddef.h>

71307 **CHANGE HISTORY**

71308 First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

71309 **NAME**

71310 wcstok — split a wide-character string into tokens

71311 **SYNOPSIS**

71312 #include <wchar.h>

```
71313       wchar_t *wcstok(wchar_t *restrict ws1, const wchar_t *restrict ws2,
71314                      wchar_t **restrict ptr);
```

71315 **DESCRIPTION**

71316 CX The functionality described on this reference page is aligned with the ISO C standard. Any
 71317 conflict between the requirements described here and the ISO C standard is unintentional. This
 71318 volume of POSIX.1-2008 defers to the ISO C standard.

71319 A sequence of calls to *wcstok()* shall break the wide-character string pointed to by *ws1* into a
 71320 sequence of tokens, each of which shall be delimited by a wide-character code from the wide-
 71321 character string pointed to by *ws2*. The *ptr* argument points to a caller-provided **wchar_t** pointer
 71322 into which the *wcstok()* function shall store information necessary for it to continue scanning the
 71323 same wide-character string.

71324 The first call in the sequence has *ws1* as its first argument, and is followed by calls with a null
 71325 pointer as their first argument. The separator string pointed to by *ws2* may be different from call
 71326 to call.

71327 The first call in the sequence shall search the wide-character string pointed to by *ws1* for the first
 71328 wide-character code that is *not* contained in the current separator string pointed to by *ws2*. If no
 71329 such wide-character code is found, then there are no tokens in the wide-character string pointed
 71330 to by *ws1* and *wcstok()* shall return a null pointer. If such a wide-character code is found, it shall
 71331 be the start of the first token.

71332 The *wcstok()* function shall then search from there for a wide-character code that *is* contained in
 71333 the current separator string. If no such wide-character code is found, the current token extends
 71334 to the end of the wide-character string pointed to by *ws1*, and subsequent searches for a token
 71335 shall return a null pointer. If such a wide-character code is found, it shall be overwritten by a
 71336 null wide character, which terminates the current token. The *wcstok()* function shall save a
 71337 pointer to the following wide-character code, from which the next search for a token shall start.

71338 Each subsequent call, with a null pointer as the value of the first argument, shall start searching
 71339 from the saved pointer and behave as described above.

71340 The implementation shall behave as if no function calls *wcstok()*.

71341 **RETURN VALUE**

71342 Upon successful completion, the *wcstok()* function shall return a pointer to the first wide-
 71343 character code of a token. Otherwise, if there is no token, *wcstok()* shall return a null pointer.

71344 **ERRORS**

71345 No errors are defined.

71346 **EXAMPLES**

71347 None.

71348 **APPLICATION USAGE**

71349 None.

71350 **RATIONALE**

71351 None.

71352 **FUTURE DIRECTIONS**

71353 None.

71354 **SEE ALSO**71355 XBD [<wchar.h>](#)71356 **CHANGE HISTORY**

71357 First released in Issue 4.

71358 **Issue 5**

71359 Aligned with ISO/IEC 9899:1990/Amendment 1:1995 (E). Specifically, a third argument is
71360 added to the definition of *wcstok()* in the SYNOPSIS.

71361 **Issue 6**71362 The *wcstok()* prototype is updated for alignment with the ISO/IEC 9899:1999 standard.

NAME

wcstol, wcstoll — convert a wide-character string to a long integer

SYNOPSIS

```
#include <wchar.h>

long wcstol(const wchar_t *restrict nptr, wchar_t **restrict endptr,
            int base);
long long wcstoll(const wchar_t *restrict nptr,
                  wchar_t **restrict endptr, int base);
```

DESCRIPTION

The functionality described on this reference page is aligned with the ISO C standard. Any conflict between the requirements described here and the ISO C standard is unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

These functions shall convert the initial portion of the wide-character string pointed to by *nptr* to **long** and **long long**, respectively. First, they shall decompose the input string into three parts:

1. An initial, possibly empty, sequence of white-space wide-character codes (as specified by *isspace()*)
2. A subject sequence interpreted as an integer represented in some radix determined by the value of *base*
3. A final wide-character string of one or more unrecognized wide-character codes, including the terminating null wide-character code of the input wide-character string

Then they shall attempt to convert the subject sequence to an integer, and return the result.

If *base* is 0, the expected form of the subject sequence is that of a decimal constant, octal constant, or hexadecimal constant, any of which may be preceded by a '+' or '-' sign. A decimal constant begins with a non-zero digit, and consists of a sequence of decimal digits. An octal constant consists of the prefix '0' optionally followed by a sequence of the digits '0' to '7' only. A hexadecimal constant consists of the prefix 0x or 0X followed by a sequence of the decimal digits and letters 'a' (or 'A') to 'f' (or 'F') with values 10 to 15 respectively.

If the value of *base* is between 2 and 36, the expected form of the subject sequence is a sequence of letters and digits representing an integer with the radix specified by *base*, optionally preceded by a '+' or '-' sign, but not including an integer suffix. The letters from 'a' (or 'A') to 'z' (or 'Z') inclusive are ascribed the values 10 to 35; only letters whose ascribed values are less than that of *base* shall be permitted. If the value of *base* is 16, the wide-character code representations of 0x or 0X may optionally precede the sequence of letters and digits, following the sign if present.

The subject sequence is defined as the longest initial subsequence of the input wide-character string, starting with the first non-white-space wide-character code that is of the expected form. The subject sequence contains no wide-character codes if the input wide-character string is empty or consists entirely of white-space wide-character code, or if the first non-white-space wide-character code is other than a sign or a permissible letter or digit.

If the subject sequence has the expected form and *base* is 0, the sequence of wide-character codes starting with the first digit shall be interpreted as an integer constant. If the subject sequence has the expected form and the value of *base* is between 2 and 36, it shall be used as the base for conversion, ascribing to each letter its value as given above. If the subject sequence begins with a minus-sign, the value resulting from the conversion shall be negated. A pointer to the final wide-character string shall be stored in the object pointed to by *endptr*, provided that *endptr* is not a null pointer.

71409 CX In other than the C or POSIX locales, other implementation-defined subject sequences may be
71410 accepted.

71411 If the subject sequence is empty or does not have the expected form, no conversion shall be
71412 performed; the value of *nptr* shall be stored in the object pointed to by *endptr*, provided that
71413 *endptr* is not a null pointer.

71414 These functions shall not change the setting of *errno* if successful. -

71415 Since 0, {LONG_MIN} or {LLONG_MIN} and {LONG_MAX} or {LLONG_MAX} are returned on
71416 error and are also valid returns on success, an application wishing to check for error situations
71417 should set *errno* to 0, then call *wcstol()* or *wcstoll()*, then check *errno*. -

71418 RETURN VALUE

71419 Upon successful completion, these functions shall return the converted value, if any. If no
71420 CX conversion could be performed, 0 shall be returned and *errno* may be set to indicate the error. If
71421 the correct value is outside the range of representable values, {LONG_MIN}, {LONG_MAX},
71422 {LLONG_MIN}, or {LLONG_MAX} shall be returned (according to the sign of the value), and
71423 *errno* set to [ERANGE].

71424 ERRORS

71425 These functions shall fail if:

71426 CX [EINVAL] The value of *base* is not supported.

71427 [ERANGE] The value to be returned is not representable.

71428 These functions may fail if:

71429 CX [EINVAL] No conversion could be performed.

71430 EXAMPLES

71431 None.

71432 APPLICATION USAGE

71433 None.

71434 RATIONALE

71435 None.

71436 FUTURE DIRECTIONS

71437 None.

71438 SEE ALSO

71439 *fscanf()*, *iswalph()*, *wcstod()*

71440 XBD <wchar.h>

71441 CHANGE HISTORY

71442 First released in Issue 4. Derived from the MSE working draft.

71443 Issue 5

71444 The DESCRIPTION is updated to indicate that *errno* is not changed if the function is successful.

71445 Issue 6

71446 Extensions beyond the ISO C standard are marked.

71447 The following new requirements on POSIX implementations derive from alignment with the
71448 Single UNIX Specification:

- 71449 • In the RETURN VALUE and ERRORS sections, the [EINVAL] optional error condition is
71450 added if no conversion could be performed.

71451 The following changes are made for alignment with the ISO/IEC 9899:1999 standard:

- 71452 • The *wcstol()* prototype is updated.
- 71453 • The *wcstoll()* function is added.

71454 **Issue 7**

71455 SD5-XSH-ERN-56 is applied, removing the reference to **unsigned long** and **unsigned long long**
71456 from the DESCRIPTION.

71457 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0725 [105] is applied.

+

71458 **NAME**

71459 wcstold — convert a wide-character string to a double-precision number

71460 **SYNOPSIS**

71461 #include <wchar.h>

71462 long double wcstold(const wchar_t *restrict *nptr*,
71463 wchar_t **restrict *endptr*);71464 **DESCRIPTION**71465 Refer to *wcstod()*.

71466 **NAME**

71467 wcstoll — convert a wide-character string to a long integer

71468 **SYNOPSIS**

71469 #include <wchar.h>

71470 long long wcstoll(const wchar_t *restrict *nptr*,
71471 wchar_t **restrict *endptr*, int *base*);71472 **DESCRIPTION**71473 Refer to *wcstol()*.

71474 **NAME**

71475 wcstombs — convert a wide-character string to a character string

71476 **SYNOPSIS**

71477 #include <stdlib.h>

71478 size_t wcstombs(char *restrict *s*, const wchar_t *restrict *pwcs*,
71479 size_t *n*);71480 **DESCRIPTION**71481 CX The functionality described on this reference page is aligned with the ISO C standard. Any
71482 conflict between the requirements described here and the ISO C standard is unintentional. This
71483 volume of POSIX.1-2008 defers to the ISO C standard.71484 The *wcstombs()* function shall convert the sequence of wide-character codes that are in the array
71485 pointed to by *pwcs* into a sequence of characters that begins in the initial shift state and store
71486 these characters into the array pointed to by *s*, stopping if a character would exceed the limit of *n*
71487 total bytes or if a null byte is stored. Each wide-character code shall be converted as if by a call to
71488 *wctomb()*, except that the shift state of *wctomb()* shall not be affected.71489 The behavior of this function shall be affected by the *LC_CTYPE* category of the current locale.71490 No more than *n* bytes shall be modified in the array pointed to by *s*. If copying takes place
71491 CX between objects that overlap, the behavior is undefined. If *s* is a null pointer, *wcstombs()* shall
71492 return the length required to convert the entire array regardless of the value of *n*, but no values
71493 are stored.71494 **RETURN VALUE**71495 If a wide-character code is encountered that does not correspond to a valid character (of one or
71496 more bytes each), *wcstombs()* shall return (**size_t**)−1. Otherwise, *wcstombs()* shall return the
71497 number of bytes stored in the character array, not including any terminating null byte. The array
71498 shall not be null-terminated if the value returned is *n*.71499 **ERRORS**71500 The *wcstombs()* function shall fail if:

71501 CX [EILSEQ] A wide-character code does not correspond to a valid character.

71502 **EXAMPLES**

71503 None.

71504 **APPLICATION USAGE**

71505 None.

71506 **RATIONALE**

71507 None.

71508 **FUTURE DIRECTIONS**

71509 None.

71510 **SEE ALSO**71511 *mblen()*, *mbtowc()*, *mbstowcs()*, *wctomb()*

71512 XBD <stdlib.h>

71513 **CHANGE HISTORY**

71514 First released in Issue 4. Derived from the ISO C standard.

71515 **Issue 6**

71516 The following new requirements on POSIX implementations derive from alignment with the
71517 Single UNIX Specification:

- 71518 • The DESCRIPTION states the effect of when *s* is a null pointer.
- 71519 • The [EILSEQ] error condition is added.

71520 The *wcstombs()* prototype is updated for alignment with the ISO/IEC 9899:1999 standard.

71521 **Issue 7**

71522 Austin Group Interpretations 1003.1-2001 #156 and #170 are applied.

71523 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0726 [109] is applied.

+

71524 **NAME**

71525 wcstoul, wcstoull — convert a wide-character string to an unsigned long

71526 **SYNOPSIS**

```
71527     #include <wchar.h>

71528     unsigned long wcstoul(const wchar_t *restrict nptr,
71529         wchar_t **restrict endptr, int base);
71530     unsigned long long wcstoull(const wchar_t *restrict nptr,
71531         wchar_t **restrict endptr, int base);
```

71532 **DESCRIPTION**

71533 CX The functionality described on this reference page is aligned with the ISO C standard. Any
 71534 conflict between the requirements described here and the ISO C standard is unintentional. This
 71535 volume of POSIX.1-2008 defers to the ISO C standard.

71536 The *wcstoul()* and *wcstoull()* functions shall convert the initial portion of the wide-character
 71537 string pointed to by *nptr* to **unsigned long** and **unsigned long long** representation, respectively.
 71538 First, they shall decompose the input wide-character string into three parts:

- 71539 1. An initial, possibly empty, sequence of white-space wide-character codes (as specified by
 71540 *iswspace()*)
- 71541 2. A subject sequence interpreted as an integer represented in some radix determined by the
 71542 value of *base*
- 71543 3. A final wide-character string of one or more unrecognized wide-character codes,
 71544 including the terminating null wide-character code of the input wide-character string

71545 Then they shall attempt to convert the subject sequence to an unsigned integer, and return the
 71546 result.

71547 If *base* is 0, the expected form of the subject sequence is that of a decimal constant, octal constant,
 71548 or hexadecimal constant, any of which may be preceded by a '+' or '-' sign. A decimal
 71549 constant begins with a non-zero digit, and consists of a sequence of decimal digits. An octal
 71550 constant consists of the prefix '0' optionally followed by a sequence of the digits '0' to '7'
 71551 only. A hexadecimal constant consists of the prefix 0x or 0X followed by a sequence of the
 71552 decimal digits and letters 'a' (or 'A') to 'f' (or 'F') with values 10 to 15 respectively.

71553 If the value of *base* is between 2 and 36, the expected form of the subject sequence is a sequence
 71554 of letters and digits representing an integer with the radix specified by *base*, optionally preceded
 71555 by a '+' or '-' sign, but not including an integer suffix. The letters from 'a' (or 'A') to 'z'
 71556 (or 'Z') inclusive are ascribed the values 10 to 35; only letters whose ascribed values are less
 71557 than that of *base* shall be permitted. If the value of *base* is 16, the wide-character codes 0x or 0X
 71558 may optionally precede the sequence of letters and digits, following the sign if present.

71559 The subject sequence is defined as the longest initial subsequence of the input wide-character
 71560 string, starting with the first wide-character code that is not white space and is of the expected
 71561 form. The subject sequence contains no wide-character codes if the input wide-character string is
 71562 empty or consists entirely of white-space wide-character codes, or if the first wide-character
 71563 code that is not white space is other than a sign or a permissible letter or digit.

71564 If the subject sequence has the expected form and *base* is 0, the sequence of wide-character codes
 71565 starting with the first digit shall be interpreted as an integer constant. If the subject sequence has
 71566 the expected form and the value of *base* is between 2 and 36, it shall be used as the base for
 71567 conversion, ascribing to each letter its value as given above. If the subject sequence begins with a
 71568 minus-sign, the value resulting from the conversion shall be negated. A pointer to the final
 71569 wide-character string shall be stored in the object pointed to by *endptr*, provided that *endptr* is

71570 not a null pointer.

71571 CX In other than the C or POSIX locales, other implementation-defined subject sequences may be
71572 accepted.

71573 If the subject sequence is empty or does not have the expected form, no conversion shall be
71574 performed; the value of *nptr* shall be stored in the object pointed to by *endptr*, provided that
71575 *endptr* is not a null pointer.

71576 These functions shall not change the setting of *errno* if successful. |

71577 Since 0, {ULONG_MAX}, and {ULLONG_MAX} are returned on error and 0 is also a valid return
71578 on success, an application wishing to check for error situations should set *errno* to 0, then call
71579 *wcstoul()* or *wcstoull()*, then check *errno*. -

71580 **RETURN VALUE**

71581 Upon successful completion, the *wcstoul()* and *wcstoull()* functions shall return the converted
71582 CX value, if any. If no conversion could be performed, 0 shall be returned and *errno* may be set to
71583 indicate the error. If the correct value is outside the range of representable values,
71584 {ULONG_MAX} or {ULLONG_MAX} respectively shall be returned and *errno* set to [ERANGE].

71585 **ERRORS**

71586 These functions shall fail if:

71587 CX [EINVAL] The value of *base* is not supported.

71588 [ERANGE] The value to be returned is not representable.

71589 These functions may fail if:

71590 CX [EINVAL] No conversion could be performed.

71591 **EXAMPLES**

71592 None.

71593 **APPLICATION USAGE**

71594 None.

71595 **RATIONALE**

71596 None.

71597 **FUTURE DIRECTIONS**

71598 None.

71599 **SEE ALSO**

71600 *fscanf()*, *iswalpha()*, *wcstod()*, *wcstol()*

71601 XBD <wchar.h>

71602 **CHANGE HISTORY**

71603 First released in Issue 4. Derived from the MSE working draft.

71604 **Issue 5**

71605 The DESCRIPTION is updated to indicate that *errno* is not changed if the function is successful.

71606 **Issue 6**

71607 Extensions beyond the ISO C standard are marked.

71608 The following new requirements on POSIX implementations derive from alignment with the
71609 Single UNIX Specification:

- 71610 • The [EINVAL] error condition is added for when the value of *base* is not supported.

71611 In the RETURN VALUE and ERRORS sections, the [EINVAL] optional error condition is
71612 added if no conversion could be performed.

71613 The following changes are made for alignment with the ISO/IEC 9899:1999 standard:

- 71614 • The *wcstoul()* prototype is updated.

- 71615 • The *wcstoull()* function is added.

71616 **Issue 7**

71617 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0727 [105] is applied.

+

71618 **NAME**

71619 wcstoumax — convert a wide-character string to an integer type

71620 **SYNOPSIS**

71621 #include <stddef.h>

71622 #include <inttypes.h>

71623 uintmax_t wcstoumax(const wchar_t *restrict *nptr*,71624 wchar_t **restrict *endptr*, int *base*);71625 **DESCRIPTION**71626 Refer to *wcstoimax()*.

71627 **NAME**

71628 wcswidth — number of column positions of a wide-character string

71629 **SYNOPSIS**

71630 XSI #include <wchar.h>

71631 int wcswidth(const wchar_t *pwcs, size_t n);

71632 **DESCRIPTION**

71633 The *wcswidth()* function shall determine the number of column positions required for *n* wide-
71634 character codes (or fewer than *n* wide-character codes if a null wide-character code is
71635 encountered before *n* wide-character codes are exhausted) in the string pointed to by *pwcs*.

71636 **RETURN VALUE**

71637 The *wcswidth()* function either shall return 0 (if *pwcs* points to a null wide-character code), or
71638 return the number of column positions to be occupied by the wide-character string pointed to by
71639 *pwcs*, or return -1 (if any of the first *n* wide-character codes in the wide-character string pointed
71640 to by *pwcs* is not a printable wide-character code).

71641 **ERRORS**

71642 No errors are defined.

71643 **EXAMPLES**

71644 None.

71645 **APPLICATION USAGE**

71646 This function was removed from the final ISO/IEC 9899:1990/Amendment 1:1995 (E), and the
71647 return value for a non-printable wide character is not specified.

71648 **RATIONALE**

71649 None.

71650 **FUTURE DIRECTIONS**

71651 None.

71652 **SEE ALSO**71653 *wcwidth()*71654 XBD [Section 3.103](#) (on page 50), [<wchar.h>](#)71655 **CHANGE HISTORY**

71656 First released in Issue 4. Derived from the MSE working draft.

71657 **Issue 6**

71658 The Open Group Corrigendum U021/11 is applied. The function is marked as an extension.

71659 **NAME**71660 `wcsxfrm`, `wcsxfrm_l` — wide-character string transformation71661 **SYNOPSIS**71662 `#include <wchar.h>`71663 `size_t wcsxfrm(wchar_t *restrict ws1, const wchar_t *restrict ws2,`
71664 `size_t n);`71665 CX `size_t wcsxfrm_l(wchar_t *restrict ws1, const wchar_t *restrict ws2,`
71666 `size_t n, locale_t locale);`71667 **DESCRIPTION**71668 CX For `wcsxfrm()`: The functionality described on this reference page is aligned with the ISO C
71669 standard. Any conflict between the requirements described here and the ISO C standard is
71670 unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.71671 CX The `wcsxfrm()` and `wcsxfrm_l()` functions shall transform the wide-character string pointed to
71672 by `ws2` and place the resulting wide-character string into the array pointed to by `ws1`. The
71673 transformation shall be such that if `wcscmp()` is applied to two transformed wide strings, it shall
71674 CX return a value greater than, equal to, or less than 0, corresponding to the result of `wcscoll()` and
71675 `wcscoll_l()` applied to the same two original wide-character strings, and the same `LC_COLLATE`
71676 CX category of the current locale or the locale object `locale`, respectively. No more than `n` wide-
71677 character codes shall be placed into the resulting array pointed to by `ws1`, including the
71678 terminating null wide-character code. If `n` is 0, `ws1` is permitted to be a null pointer. If copying
71679 takes place between objects that overlap, the behavior is undefined.71680 CX The `wcsxfrm()` and `wcsxfrm_l()` functions shall not change the setting of `errno` if successful.71681 Since no return value is reserved to indicate an error, an application wishing to check for error
71682 situations should set `errno` to 0, then call `wcsxfrm()` or `wcsxfrm_l()`, then check `errno`.71683 The behavior is undefined if the `locale` argument to `wcsxfrm_l()` is the special locale object +
71684 `LC_GLOBAL_LOCALE` or is not a valid locale object handle.71685 **RETURN VALUE**71686 CX The `wcsxfrm()` and `wcsxfrm_l()` functions shall return the length of the transformed wide-
71687 character string (not including the terminating null wide-character code). If the value returned is
71688 `n` or more, the contents of the array pointed to by `ws1` are unspecified.71689 CX On error, the `wcsxfrm()` and `wcsxfrm_l()` functions may set `errno`, but no return value is reserved
71690 to indicate an error.71691 **ERRORS**

71692 These functions may fail if:

71693 CX [EINVAL] The wide-character string pointed to by `ws2` contains wide-character codes
71694 outside the domain of the collating sequence.

71695 **EXAMPLES**

71696 None.

71697 **APPLICATION USAGE**

71698 The transformation function is such that two transformed wide-character strings can be ordered
 71699 by *wscmp()* as appropriate to collating sequence information in the current locale (category
 71700 *LC_COLLATE*).

71701 The fact that when *n* is 0 *ws1* is permitted to be a null pointer is useful to determine the size of
 71702 the *ws1* array prior to making the transformation.

71703 **RATIONALE**

71704 None.

71705 **FUTURE DIRECTIONS**

71706 None.

71707 **SEE ALSO**71708 *wscmp()*, *wscoll()*

71709 XBD <wchar.h>

71710 **CHANGE HISTORY**

71711 First released in Issue 4. Derived from the MSE working draft.

71712 **Issue 5**

71713 Moved from ENHANCED I18N to BASE and the [ENOSYS] error is removed.

71714 The DESCRIPTION is updated to indicate that *errno* is not changed if the function is successful.71715 **Issue 6**

71716 In earlier versions, this function was required to return -1 on error.

71717 Extensions beyond the ISO C standard are marked.

71718 The following new requirements on POSIX implementations derive from alignment with the
 71719 Single UNIX Specification:

- 71720 • In the RETURN VALUE and ERRORS sections, the [EINVAL] optional error condition is
 71721 added if no conversion could be performed.

71722 The *wcsxfrm()* prototype is updated for alignment with the ISO/IEC 9899:1999 standard.71723 **Issue 7**

71724 The *wcsxfrm_l()* function is added from The Open Group Technical Standard, 2006, Extended
 71725 API Set Part 4.

71726 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0728 [302], XSH/TC1-2008/0729 [283], +
 71727 XSH/TC1-2008/0730 [283], and XSH/TC1-2008/0731 [302] are applied.

71728 **NAME**

71729 wctob — wide-character to single-byte conversion

71730 **SYNOPSIS**

71731 #include <stdio.h>

71732 #include <wchar.h>

71733 int wctob(wint_t c);

71734 **DESCRIPTION**

71735 CX The functionality described on this reference page is aligned with the ISO C standard. Any
 71736 conflict between the requirements described here and the ISO C standard is unintentional. This
 71737 volume of POSIX.1-2008 defers to the ISO C standard.

71738 The *wctob()* function shall determine whether *c* corresponds to a member of the extended
 71739 character set whose character representation is a single byte when in the initial shift state.

71740 The behavior of this function shall be affected by the *LC_CTYPE* category of the current locale.

71741 **RETURN VALUE**

71742 The *wctob()* function shall return EOF if *c* does not correspond to a character with length one in
 71743 the initial shift state. Otherwise, it shall return the single-byte representation of that character as
 71744 an **unsigned char** converted to **int**.

71745 **ERRORS**

71746 No errors are defined.

71747 **EXAMPLES**

71748 None.

71749 **APPLICATION USAGE**

71750 None.

71751 **RATIONALE**

71752 None.

71753 **FUTURE DIRECTIONS**

71754 None.

71755 **SEE ALSO**71756 *btowc()*

71757 XBD <stdio.h>, <wchar.h>

71758 **CHANGE HISTORY**

71759 First released in Issue 5. Included for alignment with ISO/IEC 9899:1990/Amendment 1:1995
 71760 (E).

71761 **NAME**

71762 wctomb — convert a wide-character code to a character

71763 **SYNOPSIS**

71764 #include <stdlib.h>

71765 int wctomb(char *s, wchar_t wchar);

71766 **DESCRIPTION**

71767 CX The functionality described on this reference page is aligned with the ISO C standard. Any
 71768 conflict between the requirements described here and the ISO C standard is unintentional. This
 71769 volume of POSIX.1-2008 defers to the ISO C standard.

71770 The *wctomb()* function shall determine the number of bytes needed to represent the character
 71771 corresponding to the wide-character code whose value is *wchar* (including any change in the
 71772 shift state). It shall store the character representation (possibly multiple bytes and any special
 71773 bytes to change shift state) in the array object pointed to by *s* (if *s* is not a null pointer). At most
 71774 {MB_CUR_MAX} bytes shall be stored. If *wchar* is 0, a null byte shall be stored, preceded by any
 71775 shift sequence needed to restore the initial shift state, and *wctomb()* shall be left in the initial shift
 71776 state.

71777 CX The behavior of this function is affected by the *LC_CTYPE* category of the current locale. For a
 71778 state-dependent encoding, this function shall be placed into its initial state by a call for which its
 71779 character pointer argument, *s*, is a null pointer. Subsequent calls with *s* as other than a null
 71780 pointer shall cause the internal state of the function to be altered as necessary. A call with *s* as a
 71781 null pointer shall cause this function to return a non-zero value if encodings have state
 71782 dependency, and 0 otherwise. Changing the *LC_CTYPE* category causes the shift state of this
 71783 function to be unspecified.

71784 The *wctomb()* function need not be thread-safe.

71785 The implementation shall behave as if no function defined in this volume of POSIX.1-2008 calls
 71786 *wctomb()*.

71787 **RETURN VALUE**

71788 If *s* is a null pointer, *wctomb()* shall return a non-zero or 0 value, if character encodings,
 71789 respectively, do or do not have state-dependent encodings. If *s* is not a null pointer, *wctomb()*
 71790 shall return -1 if the value of *wchar* does not correspond to a valid character, or return the
 71791 number of bytes that constitute the character corresponding to the value of *wchar*.

71792 In no case shall the value returned be greater than the value of the {MB_CUR_MAX} macro.

71793 **ERRORS**71794 The *wctomb()* function shall fail if:

71795 CX [EILSEQ] An invalid wide-character code is detected.

71796 **EXAMPLES**

71797 None.

71798 **APPLICATION USAGE**

71799 None.

71800 **RATIONALE**

71801 None.

71802 FUTURE DIRECTIONS

71803 None.

71804 SEE ALSO

71805 [*mblen\(\)*](#), [*mbtowc\(\)*](#), [*mbstowcs\(\)*](#), [*wcstombs\(\)*](#)

71806 XBD [*<stdlib.h>*](#)

71807 CHANGE HISTORY

71808 First released in Issue 4. Derived from the ANSI C standard.

71809 Issue 6

71810 Extensions beyond the ISO C standard are marked.

71811 A note indicating that this function need not be reentrant is added to the DESCRIPTION.

71812 Issue 7

71813 Austin Group Interpretations 1003.1-2001 #156 and #170 are applied.

71814 **NAME**

71815 wctrans, wctrans_l — define character mapping

71816 **SYNOPSIS**

71817 #include <wctype.h>

71818 wctrans_t wctrans(const char *charclass);

71819 CX wctrans_t wctrans_l(const char *charclass, locale_t locale);

71820 **DESCRIPTION**

71821 CX For *wctrans()*: The functionality described on this reference page is aligned with the ISO C
 71822 standard. Any conflict between the requirements described here and the ISO C standard is
 71823 unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

71824 CX The *wctrans()* and *wctrans_l()* functions are defined for valid character mapping names
 71825 identified in the current locale. The *charclass* is a string identifying a generic character mapping
 71826 name for which codeset-specific information is required. The following character mapping
 71827 names are defined in all locales: **tolower** and **toupper**.

71828 These functions shall return a value of type **wctrans_t**, which can be used as the second
 71829 CX argument to subsequent calls of *towctrans()* and *towctrans_l()*.

71830 CX The *wctrans()* and *wctrans_l()* functions shall determine values of **wctrans_t** according to the
 71831 CX rules of the coded character set defined by character mapping information in the current locale
 71832 or in the locale represented by *locale*, respectively (category *LC_CTYPE*).

71833 The values returned by *wctrans()* shall be valid until a call to *setlocale()* that modifies the
 71834 category *LC_CTYPE*.

71835 CX The values returned by *wctrans_l()* shall be valid only in calls to *towctrans_l()* with a locale
 71836 represented by *locale* with the same *LC_CTYPE* category value.

71837 The behavior is undefined if the *locale* argument to *wctrans_l()* is the special locale object +
 71838 *LC_GLOBAL_LOCALE* or is not a valid locale object handle.

71839 **RETURN VALUE**

71840 CX The *wctrans()* and *wctrans_l()* functions shall return 0 and may set *errno* to indicate the error if
 71841 the given character mapping name is not valid for the current locale (category *LC_CTYPE*);
 71842 otherwise, they shall return a non-zero object of type **wctrans_t** that can be used in calls to
 71843 CX *towctrans()* and *towctrans_l()*.

71844 **ERRORS**

71845 These functions may fail if:

71846 CX [EINVAL] The character mapping name pointed to by *charclass* is not valid in the current
 71847 locale.

71848 **EXAMPLES**

71849 None.

71850 **APPLICATION USAGE**

71851 None.

71852 **RATIONALE**

71853 None.

71854 **FUTURE DIRECTIONS**

71855 None.

71856 **SEE ALSO**71857 *towctrans()*71858 XBD <**wctype.h**>71859 **CHANGE HISTORY**

71860 First released in Issue 5. Derived from ISO/IEC 9899:1990/Amendment 1:1995 (E).

71861 **Issue 7**71862 The *wctrans_l()* function is added from The Open Group Technical Standard, 2006, Extended
71863 API Set Part 4.71864 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0732 [302], XSH/TC1-2008/0733 [289], +
71865 XSH/TC1-2008/0734 [283], and XSH/TC1-2008/0735 [283] are applied.

71866 **NAME**

71867 wctype, wctype_l — define character class

71868 **SYNOPSIS**

71869 #include <wctype.h>

71870 wctype_t wctype(const char *property);

71871 CX wctype_t wctype_l(const char *property, locale_t locale);

71872 **DESCRIPTION**

71873 CX For *wctype()*: The functionality described on this reference page is aligned with the ISO C
 71874 standard. Any conflict between the requirements described here and the ISO C standard is
 71875 unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

71876 CX The *wctype()* and *wctype_l()* functions are defined for valid character class names as defined in
 71877 CX the current locale or in the locale represented by *locale*, respectively.

71878 The *property* argument is a string identifying a generic character class for which codeset-specific
 71879 type information is required. The following character class names shall be defined in all locales:

71880	alnum	digit	punct
71881	alpha	graph	space
71882	blank	lower	upper
71883	cntrl	print	xdigit

71884 Additional character class names defined in the locale definition file (category *LC_CTYPE*) can
 71885 also be specified.

71886 These functions shall return a value of type **wctype_t**, which can be used as the second
 71887 CX argument to subsequent calls of *iswctype()* and *iswctype_l()*.

71888 CX The *wctype()* and *wctype_l()* functions shall determine values of **wctype_t** according to the
 71889 CX rules of the coded character set defined by character type information in the current locale or in
 71890 the locale represented by *locale*, respectively (category *LC_CTYPE*).

71891 The values returned by *wctype()* shall be valid until a call to *setlocale()* that modifies the category
 71892 *LC_CTYPE*.

71893 CX The values returned by *wctype_l()* shall be valid only in calls to *iswctype_l()* with a locale
 71894 represented by *locale* with the same *LC_CTYPE* category value.

71895 The behavior is undefined if the *locale* argument to *wctype_l()* is the special locale object +
 71896 *LC_GLOBAL_LOCALE* or is not a valid locale object handle.

71897 **RETURN VALUE**

71898 CX The *wctype()* and *wctype_l()* functions shall return 0 if the given character class name is not
 71899 valid for the current locale (category *LC_CTYPE*); otherwise, they shall return an object of type
 71900 CX **wctype_t** that can be used in calls to *iswctype()* and *iswctype_l()*.

71901 **ERRORS**

71902 No errors are defined.

71903 **EXAMPLES**

71904 None.

71905 **APPLICATION USAGE**

71906 None.

71907 **RATIONALE**

71908 None.

71909 **FUTURE DIRECTIONS**

71910 None.

71911 **SEE ALSO**71912 *iswctype()*71913 XBD **<wctype.h>**71914 **CHANGE HISTORY**

71915 First released in Issue 4.

71916 **Issue 5**71917 The following change has been made in this version for alignment with
71918 ISO/IEC 9899:1990/Amendment 1:1995 (E):

- 71919
- The SYNOPSIS has been changed to indicate that this function and associated data types
71920 are now made visible by inclusion of the **<wctype.h>** header rather than **<wchar.h>**.

71921 **Issue 7**71922 The *wctype_l()* function is added from The Open Group Technical Standard, 2006, Extended API
71923 Set Part 4.71924 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0736 [302], XSH/TC1-2008/0737 [283], +
71925 and XSH/TC1-2008/0738 [283] are applied.

71926 **NAME**

71927 wctype — number of column positions of a wide-character code

71928 **SYNOPSIS**

```
71929 XSI      #include <wctype.h>
71930          int wctype(wctype_t wc);
```

71931 **DESCRIPTION**

71932 The *wctype()* function shall determine the number of column positions required for the wide
71933 character *wc*. The application shall ensure that the value of *wc* is a character representable as a
71934 *wctype_t*, and is a wide-character code corresponding to a valid character in the current locale.

71935 **RETURN VALUE**

71936 The *wctype()* function shall either return 0 (if *wc* is a null wide-character code), or return the
71937 number of column positions to be occupied by the wide-character code *wc*, or return -1 (if *wc*
71938 does not correspond to a printable wide-character code).

71939 **ERRORS**

71940 No errors are defined.

71941 **EXAMPLES**

71942 None.

71943 **APPLICATION USAGE**

71944 This function was removed from the final ISO/IEC 9899:1990/Amendment 1:1995 (E), and the
71945 return value for a non-printable wide character is not specified.

71946 **RATIONALE**

71947 None.

71948 **FUTURE DIRECTIONS**

71949 None.

71950 **SEE ALSO**71951 *wcsnwidth()*

71952 XBD <wctype.h>

71953 **CHANGE HISTORY**

71954 First released as a World-wide Portability Interface in Issue 4. Derived from the MSE working
71955 draft.

71956 **Issue 6**

71957 The Open Group Corrigendum U021/12 is applied. This function is marked as an extension.

71958 The normative text is updated to avoid use of the term “must” for application requirements.

NAME

wmemchr — find a wide character in memory

SYNOPSIS

```
#include <wchar.h>
```

```
wchar_t *wmemchr(const wchar_t *ws, wchar_t wc, size_t n);
```

DESCRIPTION

CX The functionality described on this reference page is aligned with the ISO C standard. Any conflict between the requirements described here and the ISO C standard is unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

The *wmemchr()* function shall locate the first occurrence of *wc* in the initial *n* wide characters of the object pointed to by *ws*. This function shall not be affected by locale and all **wchar_t** values shall be treated identically. The null wide character and **wchar_t** values not corresponding to valid characters shall not be treated specially.

If *n* is zero, the application shall ensure that *ws* is a valid pointer and the function behaves as if no valid occurrence of *wc* is found.

RETURN VALUE

The *wmemchr()* function shall return a pointer to the located wide character, or a null pointer if the wide character does not occur in the object.

ERRORS

No errors are defined.

EXAMPLES

None.

APPLICATION USAGE

None.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

wmemcmp(), *wmemcpy()*, *wmemmove()*, *wmemset()*

XBD **<wchar.h>**

CHANGE HISTORY

First released in Issue 5. Included for alignment with ISO/IEC 9899:1990/Amendment 1:1995 (E).

Issue 6

The normative text is updated to avoid use of the term “must” for application requirements.

71995 **NAME**

71996 wmemcmp — compare wide characters in memory

71997 **SYNOPSIS**

71998 #include <wchar.h>

71999 int wmemcmp(const wchar_t *ws1, const wchar_t *ws2, size_t n);

72000 **DESCRIPTION**

72001 CX The functionality described on this reference page is aligned with the ISO C standard. Any
 72002 conflict between the requirements described here and the ISO C standard is unintentional. This
 72003 volume of POSIX.1-2008 defers to the ISO C standard.

72004 The *wmemcmp()* function shall compare the first *n* wide characters of the object pointed to by
 72005 *ws1* to the first *n* wide characters of the object pointed to by *ws2*. This function shall not be
 72006 affected by locale and all **wchar_t** values shall be treated identically. The null wide character and
 72007 **wchar_t** values not corresponding to valid characters shall not be treated specially.

72008 If *n* is zero, the application shall ensure that *ws1* and *ws2* are valid pointers, and the function
 72009 shall behave as if the two objects compare equal.

72010 **RETURN VALUE**

72011 The *wmemcmp()* function shall return an integer greater than, equal to, or less than zero,
 72012 respectively, as the object pointed to by *ws1* is greater than, equal to, or less than the object
 72013 pointed to by *ws2*.

72014 **ERRORS**

72015 No errors are defined.

72016 **EXAMPLES**

72017 None.

72018 **APPLICATION USAGE**

72019 None.

72020 **RATIONALE**

72021 None.

72022 **FUTURE DIRECTIONS**

72023 None.

72024 **SEE ALSO**72025 *wmemchr()*, *wmemcpy()*, *wmemmove()*, *wmemset()*72026 XBD <**wchar.h**>72027 **CHANGE HISTORY**

72028 First released in Issue 5. Included for alignment with ISO/IEC 9899:1990/Amendment 1:1995
 72029 (E).

72030 **Issue 6**

72031 The normative text is updated to avoid use of the term “must” for application requirements.

72032 **NAME**

72033 wmemcpy — copy wide characters in memory

72034 **SYNOPSIS**

72035 #include <wchar.h>

72036 wchar_t *wmemcpy(wchar_t *restrict *ws1*, const wchar_t *restrict *ws2*,
72037 size_t *n*);72038 **DESCRIPTION**72039 **CX** The functionality described on this reference page is aligned with the ISO C standard. Any
72040 conflict between the requirements described here and the ISO C standard is unintentional. This
72041 volume of POSIX.1-2008 defers to the ISO C standard.72042 The *wmemcpy()* function shall copy *n* wide characters from the object pointed to by *ws2* to the
72043 object pointed to by *ws1*. This function shall not be affected by locale and all **wchar_t** values
72044 shall be treated identically. The null wide character and **wchar_t** values not corresponding to
72045 valid characters shall not be treated specially.72046 If *n* is zero, the application shall ensure that *ws1* and *ws2* are valid pointers, and the function
72047 shall copy zero wide characters.72048 **RETURN VALUE**72049 The *wmemcpy()* function shall return the value of *ws1*.72050 **ERRORS**

72051 No errors are defined.

72052 **EXAMPLES**

72053 None.

72054 **APPLICATION USAGE**

72055 None.

72056 **RATIONALE**

72057 None.

72058 **FUTURE DIRECTIONS**

72059 None.

72060 **SEE ALSO**72061 *wmemchr()*, *wmemcmp()*, *wmemmove()*, *wmemset()*

72062 XBD <wchar.h>

72063 **CHANGE HISTORY**72064 First released in Issue 5. Included for alignment with ISO/IEC 9899:1990/Amendment 1:1995
72065 (E).72066 **Issue 6**

72067 The normative text is updated to avoid use of the term “must” for application requirements.

72068 The *wmemcpy()* prototype is updated for alignment with the ISO/IEC 9899:1999 standard.

72069 **NAME**72070 `wmemmove` — copy wide characters in memory with overlapping areas72071 **SYNOPSIS**72072 `#include <wchar.h>`72073 `wchar_t *wmemmove(wchar_t *ws1, const wchar_t *ws2, size_t n);`72074 **DESCRIPTION**

72075 **CX** The functionality described on this reference page is aligned with the ISO C standard. Any
 72076 conflict between the requirements described here and the ISO C standard is unintentional. This
 72077 volume of POSIX.1-2008 defers to the ISO C standard.

72078 The `wmemmove()` function shall copy *n* wide characters from the object pointed to by *ws2* to the
 72079 object pointed to by *ws1*. Copying shall take place as if the *n* wide characters from the object
 72080 pointed to by *ws2* are first copied into a temporary array of *n* wide characters that does not
 72081 overlap the objects pointed to by *ws1* or *ws2*, and then the *n* wide characters from the temporary
 72082 array are copied into the object pointed to by *ws1*.

72083 This function shall not be affected by locale and all **wchar_t** values shall be treated identically.
 72084 The null wide character and **wchar_t** values not corresponding to valid characters shall not be
 72085 treated specially.

72086 If *n* is zero, the application shall ensure that *ws1* and *ws2* are valid pointers, and the function
 72087 shall copy zero wide characters.

72088 **RETURN VALUE**72089 The `wmemmove()` function shall return the value of *ws1*.72090 **ERRORS**

72091 No errors are defined

72092 **EXAMPLES**

72093 None.

72094 **APPLICATION USAGE**

72095 None.

72096 **RATIONALE**

72097 None.

72098 **FUTURE DIRECTIONS**

72099 None.

72100 **SEE ALSO**72101 [`wmemchr\(\)`](#), [`wmemcmp\(\)`](#), [`wmemcpy\(\)`](#), [`wmemset\(\)`](#)72102 XBD [`<wchar.h>`](#)72103 **CHANGE HISTORY**

72104 First released in Issue 5. Included for alignment with ISO/IEC 9899:1990/Amendment 1:1995
 72105 (E).

72106 **Issue 6**

72107 The normative text is updated to avoid use of the term “must” for application requirements.

72108 **NAME**

72109 wmemset — set wide characters in memory

72110 **SYNOPSIS**

72111 #include <wchar.h>

72112 wchar_t *wmemset(wchar_t *ws, wchar_t wc, size_t n);

72113 **DESCRIPTION**

72114 CX The functionality described on this reference page is aligned with the ISO C standard. Any
 72115 conflict between the requirements described here and the ISO C standard is unintentional. This
 72116 volume of POSIX.1-2008 defers to the ISO C standard.

72117 The *wmemset()* function shall copy the value of *wc* into each of the first *n* wide characters of the
 72118 object pointed to by *ws*. This function shall not be affected by locale and all **wchar_t** values shall
 72119 be treated identically. The null wide character and **wchar_t** values not corresponding to valid
 72120 characters shall not be treated specially.

72121 If *n* is zero, the application shall ensure that *ws* is a valid pointer, and the function shall copy
 72122 zero wide characters.

72123 **RETURN VALUE**72124 The *wmemset()* functions shall return the value of *ws*.72125 **ERRORS**

72126 No errors are defined.

72127 **EXAMPLES**

72128 None.

72129 **APPLICATION USAGE**

72130 None.

72131 **RATIONALE**

72132 None.

72133 **FUTURE DIRECTIONS**

72134 None.

72135 **SEE ALSO**72136 *wmemchr()*, *wmemcmp()*, *wmemcpy()*, *wmemmove()*

72137 XBD <wchar.h>

72138 **CHANGE HISTORY**

72139 First released in Issue 5. Included for alignment with ISO/IEC 9899:1990/Amendment 1:1995
 72140 (E).

72141 **Issue 6**

72142 The normative text is updated to avoid use of the term “must” for application requirements.

NAME

wordexp, wordfree — perform word expansions

SYNOPSIS

```
#include <wordexp.h>

int wordexp(const char *restrict words, wordexp_t *restrict pwordexp,
            int flags);
void wordfree(wordexp_t *pwordexp);
```

DESCRIPTION

The *wordexp()* function shall perform word expansions as described in XCU [Section 2.6](#) (on page 2327), subject to quoting as described in XCU [Section 2.2](#) (on page 2320), and place the list of expanded words into the structure pointed to by *pwordexp*.

The *words* argument is a pointer to a string containing one or more words to be expanded. The expansions shall be the same as would be performed by the command line interpreter if *words* were the part of a command line representing the arguments to a utility. Therefore, the application shall ensure that *words* does not contain an unquoted <newline> character or any of the unquoted shell special characters ' | ', ' & ', ' ; ', ' < ', ' > ' except in the context of command substitution as specified in XCU [Section 2.6.3](#) (on page 2331). It also shall not contain unquoted parentheses or braces, except in the context of command or variable substitution. The application shall ensure that every member of *words* which it expects to have expanded by *wordexp()* does not contain an unquoted initial comment character. The application shall also ensure that any words which it intends to be ignored (because they begin or continue a comment) are deleted from *words*. If the argument *words* contains an unquoted comment character (<number-sign>) that is the beginning of a token, *wordexp()* shall either treat the comment character as a regular character, or interpret it as a comment indicator and ignore the remainder of *words*.

The structure type **wordexp_t** is defined in the **<wordexp.h>** header and includes at least the following members:

Member Type	Member Name	Description
size_t	<i>we_wordc</i>	Count of words matched by <i>words</i> .
char **	<i>we_wordv</i>	Pointer to list of expanded words.
size_t	<i>we_offs</i>	Slots to reserve at the beginning of <i>pwordexp->we_wordv</i> .

The *wordexp()* function shall store the number of generated words into *pwordexp->we_wordc* and a pointer to a list of pointers to words in *pwordexp->we_wordv*. Each individual field created during field splitting (see XCU [Section 2.6.5](#), on page 2333) or pathname expansion (see XCU [Section 2.6.6](#), on page 2333) shall be a separate word in the *pwordexp->we_wordv* list. The words shall be in order as described in XCU [Section 2.6](#) (on page 2327). The first pointer after the last word pointer shall be a null pointer. The expansion of special parameters described in XCU [Section 2.5.2](#) (on page 2324) is unspecified.

It is the caller's responsibility to allocate the storage pointed to by *pwordexp*. The *wordexp()* function shall allocate other space as needed, including memory pointed to by *pwordexp->we_wordv*. The *wordfree()* function frees any memory associated with *pwordexp* from a previous call to *wordexp()*.

The *flags* argument is used to control the behavior of *wordexp()*. The value of *flags* is the bitwise-inclusive OR of zero or more of the following constants, which are defined in **<wordexp.h>**:

72188	WRDE_APPEND	Append words generated to the ones from a previous call to <i>wordexp()</i> .
72189	WRDE_DOOFFS	Make use of <i>pwordexp->we_offs</i> . If this flag is set, <i>pwordexp->we_offs</i> is used to specify how many null pointers to add to the beginning of <i>pwordexp->we_wordv</i> . In other words, <i>pwordexp->we_wordv</i> shall point to <i>pwordexp->we_offs</i> null pointers, followed by <i>pwordexp->we_wordc</i> word pointers, followed by a null pointer.
72190		
72191		
72192		
72193		
72194	WRDE_NOCMD	If the implementation supports the utilities defined in the Shell and Utilities volume of POSIX.1-2008, fail if command substitution, as specified in XCU Section 2.6.3 (on page 2331), is requested.
72195		
72196		
72197	WRDE_REUSE	The <i>pwordexp</i> argument was passed to a previous successful call to <i>wordexp()</i> , and has not been passed to <i>wordfree()</i> . The result shall be the same as if the application had called <i>wordfree()</i> and then called <i>wordexp()</i> without WRDE_REUSE.
72198		
72199		
72200		
72201	WRDE_SHOWERR	Do not redirect <i>stderr</i> to /dev/null .
72202	WRDE_UNDEF	Report error on an attempt to expand an undefined shell variable.
72203	The WRDE_APPEND flag can be used to append a new set of words to those generated by a	
72204	previous call to <i>wordexp()</i> . The following rules apply to applications when two or more calls to	
72205	<i>wordexp()</i> are made with the same value of <i>pwordexp</i> and without intervening calls to <i>wordfree()</i> :	
72206	1. The first such call shall not set WRDE_APPEND. All subsequent calls shall set it.	
72207	2. All of the calls shall set WRDE_DOOFFS, or all shall not set it.	
72208	3. After the second and each subsequent call, <i>pwordexp->we_wordv</i> shall point to a list	
72209	containing the following:	
72210	a. Zero or more null pointers, as specified by WRDE_DOOFFS and	
72211	<i>pwordexp->we_offs</i>	
72212	b. Pointers to the words that were in the <i>pwordexp->we_wordv</i> list before the call, in	
72213	the same order as before	
72214	c. Pointers to the new words generated by the latest call, in the specified order	
72215	4. The count returned in <i>pwordexp->we_wordc</i> shall be the total number of words from all of	
72216	the calls.	
72217	5. The application can change any of the fields after a call to <i>wordexp()</i> , but if it does it shall	
72218	reset them to the original value before a subsequent call, using the same <i>pwordexp</i> value,	
72219	to <i>wordfree()</i> or <i>wordexp()</i> with the WRDE_APPEND or WRDE_REUSE flag.	
72220	If the implementation supports the utilities defined in the Shell and Utilities volume of	
72221	POSIX.1-2008, and <i>words</i> contains an unquoted character—<newline>, ' ', ' & ', ' ; ', ' < ', ' > ',	
72222	' (', ') ', ' { ', ' } '—in an inappropriate context, <i>wordexp()</i> shall fail, and the number of	
72223	expanded words shall be 0.	
72224	Unless WRDE_SHOWERR is set in <i>flags</i> , <i>wordexp()</i> shall redirect <i>stderr</i> to /dev/null for any	
72225	utilities executed as a result of command substitution while expanding <i>words</i> . If	
72226	WRDE_SHOWERR is set, <i>wordexp()</i> may write messages to <i>stderr</i> if syntax errors are detected	
72227	while expanding <i>words</i> ; however, it is unspecified whether any write errors encountered while	
72228	outputting such messages will affect the <i>stderr</i> error indicator or the value of <i>errno</i> .	
72229	The application shall ensure that if WRDE_DOOFFS is set, then <i>pwordexp->we_offs</i> has the same	
72230	value for each <i>wordexp()</i> call and <i>wordfree()</i> call using a given <i>pwordexp</i> .	

72231 The following constants are defined as error return values:

72232	WRDE_BADCHAR	One of the unquoted characters—<newline>, ' ', ' & ', ' ; ', ' < ', ' > ', ' (', ') ', ' { ', ' } '—appears in <i>words</i> in an inappropriate context.
72233		
72234	WRDE_BADVAL	Reference to undefined shell variable when WRDE_UNDEF is set in <i>flags</i> .
72235	WRDE_CMDSUB	Command substitution requested when WRDE_NOCMD was set in <i>flags</i> .
72236	WRDE_NOSPACE	Attempt to allocate memory failed.
72237	WRDE_SYNTAX	Shell syntax error, such as unbalanced parentheses or unterminated string.
72238		

72239 RETURN VALUE

72240 Upon successful completion, *wordexp()* shall return 0. Otherwise, a non-zero value, as described
 72241 in <**wordexp.h**>, shall be returned to indicate an error. If *wordexp()* returns the value
 72242 WRDE_NOSPACE, then *pwordexp->we_wordc* and *pwordexp->we_wordv* shall be updated to
 72243 reflect any words that were successfully expanded. In other cases, they shall not be modified.

72244 The *wordfree()* function shall not return a value.

72245 ERRORS

72246 No errors are defined.

72247 EXAMPLES

72248 None.

72249 APPLICATION USAGE

72250 The *wordexp()* function is intended to be used by an application that wants to do all of the shell's
 72251 expansions on a word or words obtained from a user. For example, if the application prompts
 72252 for a pathname (or list of pathnames) and then uses *wordexp()* to process the input, the user
 72253 could respond with anything that would be valid as input to the shell.

72254 The WRDE_NOCMD flag is provided for applications that, for security or other reasons, want to
 72255 prevent a user from executing shell commands. Disallowing unquoted shell special characters
 72256 also prevents unwanted side-effects, such as executing a command or writing a file.

72257 POSIX.1-2008 does not require the *wordexp()* function to be thread-safe if passed an expression
 72258 referencing an environment variable while any other thread is concurrently modifying any
 72259 environment variable; see *exec* (on page 777).

72260 Even though the WRDE_SHOWERR flag allows the implementation to write messages to *stderr* +
 72261 during command substitution or syntax errors, this standard does not provide any way to detect +
 72262 write failures during the output of such messages.

72263 RATIONALE

72264 This function was included as an alternative to *glob()*. There had been continuing controversy
 72265 over exactly what features should be included in *glob()*. It is hoped that by providing *wordexp()*
 72266 (which provides all of the shell word expansions, but which may be slow to execute) and *glob()*
 72267 (which is faster, but which only performs pathname expansion, without tilde or parameter
 72268 expansion) this will satisfy the majority of applications.

72269 While *wordexp()* could be implemented entirely as a library routine, it is expected that most
 72270 implementations run a shell in a subprocess to do the expansion.

72271 Two different approaches have been proposed for how the required information might be
 72272 presented to the shell and the results returned. They are presented here as examples.

72273 One proposal is to extend the *echo* utility by adding a **-q** option. This option would cause *echo* to

add a <backslash> before each <backslash> and <blank> that occurs within an argument. The *wordexp()* function could then invoke the shell as follows:

```
(void) strcpy(buffer, "echo -q");
(void) strcat(buffer, words);
if ((flags & WRDE_SHOWERR) == 0)
    (void) strcat(buffer, "2>/dev/null");
f = popen(buffer, "r");
```

The *wordexp()* function would read the resulting output, remove unquoted <backslash> characters, and break into words at unquoted <blank> characters. If the WRDE_NOCMD flag was set, *wordexp()* would have to scan *words* before starting the subshell to make sure that there would be no command substitution. In any case, it would have to scan *words* for unquoted special characters.

Another proposal is to add the following options to *sh*:

-w *wordlist*

This option provides a wordlist expansion service to applications. The words in *wordlist* shall be expanded and the following written to standard output:

1. The count of the number of words after expansion, in decimal, followed by a null byte
2. The number of bytes needed to represent the expanded words (not including null separators), in decimal, followed by a null byte
3. The expanded words, each terminated by a null byte

If an error is encountered during word expansion, *sh* exits with a non-zero status after writing the former to report any words successfully expanded

-P Run in “protected” mode. If specified with the **-w** option, no command substitution shall be performed.

With these options, *wordexp()* could be implemented fairly simply by creating a subprocess using *fork()* and executing *sh* using the line:

```
execl(<shell path>, "sh", "-P", "-w", words, (char *)0);
```

after directing standard error to **/dev/null**.

It seemed objectionable for a library routine to write messages to standard error, unless explicitly requested, so *wordexp()* is required to redirect standard error to **/dev/null** to ensure that no messages are generated, even for commands executed for command substitution. The WRDE_SHOWERR flag can be specified to request that error messages be written.

The WRDE_REUSE flag allows the implementation to avoid the expense of freeing and reallocating memory, if that is possible. A minimal implementation can call *wordfree()* when WRDE_REUSE is set.

FUTURE DIRECTIONS

None.

SEE ALSO

exec, *fnmatch()*, *glob()*

XBD **<wordexp.h>**

XCU Chapter 2 (on page 2319)

72316 **CHANGE HISTORY**

72317 First released in Issue 4. Derived from the ISO POSIX-2 standard.

72318 **Issue 5**

72319 Moved from POSIX2 C-language Binding to BASE.

72320 **Issue 6**

72321 The normative text is updated to avoid use of the term “must” for application requirements.

72322 The **restrict** keyword is added to the *wordexp()* prototype for alignment with the
72323 ISO/IEC 9899:1999 standard.

72324 **Issue 7**

72325 Austin Group Interpretation 1003.1-2001 #148 is applied, adding APPLICATION USAGE.

72326 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0739 [460], XSH/TC1-2008/0740 [291], +
72327 and XSH/TC1-2008/0741 [460] are applied.

72328 **NAME**

72329 `wprintf` — print formatted wide-character output

72330 **SYNOPSIS**

72331 `#include <stdio.h>`

72332 `#include <wchar.h>`

72333 `int wprintf(const wchar_t *restrict format, ...);`

72334 **DESCRIPTION**

72335 Refer to *fwprintf()*.

72336 **NAME**72337 `pwrite, write` — write on a file72338 **SYNOPSIS**

```
72339 #include <unistd.h>
72340 ssize_t pwrite(int fildes, const void *buf, size_t nbyte,
72341               off_t offset);
72342 ssize_t write(int fildes, const void *buf, size_t nbyte);
```

72343 **DESCRIPTION**

72344 The `write()` function shall attempt to write *nbyte* bytes from the buffer pointed to by *buf* to the
 72345 file associated with the open file descriptor, *fildes*.

72346 Before any action described below is taken, and if *nbyte* is zero and the file is a regular file, the
 72347 `write()` function may detect and return errors as described below. In the absence of errors, or if
 72348 error detection is not performed, the `write()` function shall return zero and have no other results.
 72349 If *nbyte* is zero and the file is not a regular file, the results are unspecified.

72350 On a regular file or other file capable of seeking, the actual writing of data shall proceed from
 72351 the position in the file indicated by the file offset associated with *fildes*. Before successful return
 72352 from `write()`, the file offset shall be incremented by the number of bytes actually written. On a
 72353 regular file, if the position of the last byte written is greater than or equal to the length of the file,
 72354 the length of the file shall be set to this position plus one.

72355 On a file not capable of seeking, writing shall always take place starting at the current position.
 72356 The value of a file offset associated with such a device is undefined.

72357 If the `O_APPEND` flag of the file status flags is set, the file offset shall be set to the end of the file
 72358 prior to each write and no intervening file modification operation shall occur between changing
 72359 the file offset and the write operation.

72360 XSI If a `write()` requests that more bytes be written than there is room for (for example, the file size
 72361 limit of the process or the physical end of a medium), only as many bytes as there is room for
 72362 shall be written. For example, suppose there is space for 20 bytes more in a file before reaching a
 72363 limit. A write of 512 bytes will return 20. The next write of a non-zero number of bytes would
 72364 give a failure return (except as noted below).

72365 XSI If the request would cause the file size to exceed the soft file size limit for the process and there
 72366 is no room for any bytes to be written, the request shall fail and the implementation shall
 72367 generate the `SIGXFSZ` signal for the thread.

72368 If `write()` is interrupted by a signal before it writes any data, it shall return `-1` with *errno* set to
 72369 `[EINTR]`.

72370 If `write()` is interrupted by a signal after it successfully writes some data, it shall return the
 72371 number of bytes written.

72372 If the value of *nbyte* is greater than `{SSIZE_MAX}`, the result is implementation-defined.

72373 After a `write()` to a regular file has successfully returned:

- 72374 • Any successful `read()` from each byte position in the file that was modified by that write
 72375 shall return the data specified by the `write()` for that position until such byte positions are
 72376 again modified.
- 72377 • Any subsequent successful `write()` to the same byte position in the file shall overwrite that
 72378 file data.

72379 Write requests to a pipe or FIFO shall be handled in the same way as a regular file with the
72380 following exceptions:

- 72381 • There is no file offset associated with a pipe, hence each write request shall append to the
72382 end of the pipe.
- 72383 • Write requests of {PIPE_BUF} bytes or less shall not be interleaved with data from other
72384 processes doing writes on the same pipe. Writes of greater than {PIPE_BUF} bytes may
72385 have data interleaved, on arbitrary boundaries, with writes by other processes, whether or
72386 not the O_NONBLOCK flag of the file status flags is set.
- 72387 • If the O_NONBLOCK flag is clear, a write request may cause the thread to block, but on
72388 normal completion it shall return *nbyte*.
- 72389 • If the O_NONBLOCK flag is set, *write()* requests shall be handled differently, in the
72390 following ways:
 - 72391 — The *write()* function shall not block the thread.
 - 72392 — A write request for {PIPE_BUF} or fewer bytes shall have the following effect: if there
72393 is sufficient space available in the pipe, *write()* shall transfer all the data and return
72394 the number of bytes requested. Otherwise, *write()* shall transfer no data and return
72395 -1 with *errno* set to [EAGAIN].
 - 72396 — A write request for more than {PIPE_BUF} bytes shall cause one of the following:
 - 72397 — When at least one byte can be written, transfer what it can and return the
72398 number of bytes written. When all data previously written to the pipe is read, it
72399 shall transfer at least {PIPE_BUF} bytes.
 - 72400 — When no data can be written, transfer no data, and return -1 with *errno* set to
72401 [EAGAIN].

72402 When attempting to write to a file descriptor (other than a pipe or FIFO) that supports non-
72403 blocking writes and cannot accept the data immediately:

- 72404 • If the O_NONBLOCK flag is clear, *write()* shall block the calling thread until the data can
72405 be accepted.
- 72406 • If the O_NONBLOCK flag is set, *write()* shall not block the thread. If some data can be
72407 written without blocking the thread, *write()* shall write what it can and return the number
72408 of bytes written. Otherwise, it shall return -1 and set *errno* to [EAGAIN].

72409 Upon successful completion, where *nbyte* is greater than 0, *write()* shall mark for update the last
72410 data modification and last file status change timestamps of the file, and if the file is a regular file,
72411 the S_ISUID and S_ISGID bits of the file mode may be cleared.

72412 For regular files, no data transfer shall occur past the offset maximum established in the open
72413 file description associated with *filides*.

72414 If *filides* refers to a socket, *write()* shall be equivalent to *send()* with no flags set.

72415 SIO If the O_DSYNC bit has been set, write I/O operations on the file descriptor shall complete as
72416 defined by synchronized I/O data integrity completion.

72417 If the O_SYNC bit has been set, write I/O operations on the file descriptor shall complete as
72418 defined by synchronized I/O file integrity completion.

72419 SHM If *filides* refers to a shared memory object, the result of the *write()* function is unspecified.

72420	TYM	If <i>fildes</i> refers to a typed memory object, the result of the <i>write()</i> function is unspecified.
72421	OB XSR	If <i>fildes</i> refers to a STREAM, the operation of <i>write()</i> shall be determined by the values of the minimum and maximum <i>nbyte</i> range (packet size) accepted by the STREAM. These values are determined by the topmost STREAM module. If <i>nbyte</i> falls within the packet size range, <i>nbyte</i> bytes shall be written. If <i>nbyte</i> does not fall within the range and the minimum packet size value is 0, <i>write()</i> shall break the buffer into maximum packet size segments prior to sending the data downstream (the last segment may contain less than the maximum packet size). If <i>nbyte</i> does not fall within the range and the minimum value is non-zero, <i>write()</i> shall fail with <i>errno</i> set to [ERANGE]. Writing a zero-length buffer (<i>nbyte</i> is 0) to a STREAMS device sends 0 bytes with 0 returned. However, writing a zero-length buffer to a STREAMS-based pipe or FIFO sends no message and 0 is returned. The process may issue <code>I_SWROPT ioctl()</code> to enable zero-length messages to be sent across the pipe or FIFO. When writing to a STREAM, data messages are created with a priority band of 0. When writing to a STREAM that is not a pipe or FIFO: • If <code>O_NONBLOCK</code> is clear, and the STREAM cannot accept data (the STREAM write queue is full due to internal flow control conditions), <i>write()</i> shall block until data can be accepted. • If <code>O_NONBLOCK</code> is set and the STREAM cannot accept data, <i>write()</i> shall return -1 and set <i>errno</i> to [EAGAIN]. • If <code>O_NONBLOCK</code> is set and part of the buffer has been written while a condition in which the STREAM cannot accept additional data occurs, <i>write()</i> shall terminate and return the number of bytes written. In addition, <i>write()</i> shall fail if the STREAM head has processed an asynchronous error before the call. In this case, the value of <i>errno</i> does not reflect the result of <i>write()</i>, but reflects the prior error. The <i>pwrite()</i> function shall be equivalent to <i>write()</i>, except that it writes into a given position and does not change the file offset (regardless of whether <code>O_APPEND</code> is set). The first three arguments to <i>pwrite()</i> are the same as <i>write()</i> with the addition of a fourth argument <i>offset</i> for the desired position inside the file. An attempt to perform a <i>pwrite()</i> on a file that is incapable of seeking shall result in an error.
72422		
72423		
72424		
72425		
72426		
72427		
72428		
72429		
72430		
72432		
72433		
72434		
72435		
72436		
72437		
72438		
72439		
72440		
72441		
72442		
72443		
72444		
72445		The <i>pwrite()</i> function shall be equivalent to <i>write()</i> , except that it writes into a given position and does not change the file offset (regardless of whether <code>O_APPEND</code> is set). The first three arguments to <i>pwrite()</i> are the same as <i>write()</i> with the addition of a fourth argument <i>offset</i> for the desired position inside the file. An attempt to perform a <i>pwrite()</i> on a file that is incapable of seeking shall result in an error.
72446		
72447		
72448		
72449		
72450	RETURN VALUE	Upon successful completion, these functions shall return the number of bytes actually written to the file associated with <i>fildes</i> . This number shall never be greater than <i>nbyte</i> . Otherwise, -1 shall be returned and <i>errno</i> set to indicate the error.
72451		
72452		
72453		
72454	ERRORS	These functions shall fail if: [EAGAIN] The file is neither a pipe, nor a FIFO, nor a socket, the <code>O_NONBLOCK</code> flag is set for the file descriptor, and the thread would be delayed in the <i>write()</i> operation. [EBADF] The <i>fildes</i> argument is not a valid file descriptor open for writing. [EFBIG] An attempt was made to write a file that exceeds the implementation-defined maximum file size <u>or the file size limit of the process</u>, and there was no room for any bytes to be written.
72455		
72456		
72457		
72458		
72459		
72460		
72461	XSI	
72462		

72463		[EFBIG]	The file is a regular file, <i>nbyte</i> is greater than 0, and the starting position is greater than or equal to the offset maximum established in the open file description associated with <i>fildest</i> .	
72464				
72465				
72466		[EINTR]	The write operation was terminated due to the receipt of a signal, and no data was transferred.	
72467				
72468		[EIO]	The process is a member of a background process group attempting to write to its controlling terminal, TOSTOP is set, the calling thread is not blocking SIGTTOU, the process is not ignoring SIGTTOU, and the process group of the process is orphaned. This error may also be returned under implementation-defined conditions.	
72469				
72470				
72471				
72472				
72473		[ENOSPC]	There was no free space remaining on the device containing the file.	-
72474	OB XSR	[ERANGE]	The transfer request size was outside the range supported by the STREAMS file associated with <i>fildest</i> .	
72475				
72476			The <i>pwrite()</i> function shall fail if:	+
72477		[EINVAL]	The file is a regular file or block special file, and the <i>offset</i> argument is negative. The file pointer shall remain unchanged.	+
72478				+
72479		[ESPIPE]	The file is a pipe, FIFO, or socket.	+
72480			The <i>write()</i> function shall fail if:	
72481		[EAGAIN]	The file is a pipe or FIFO, the O_NONBLOCK flag is set for the file descriptor, and the thread would be delayed in the write operation.	+
72482				
72483		[EAGAIN] or [EWOULDBLOCK]		
72484			The file is a socket, the O_NONBLOCK flag is set for the file descriptor, and the thread would be delayed in the write operation.	
72485				
72486		[ECONNRESET]	A write was attempted on a socket that is not connected.	
72487		[EPIPE]	An attempt is made to write to a pipe or FIFO that is not open for reading by any process, or that only has one end open. A SIGPIPE signal shall also be sent to the thread.	+
72488				+
72489				+
72490		[EPIPE]	A write was attempted on a socket that is shut down for writing, or is no longer connected. In the latter case, if the socket is of type SOCK_STREAM, a SIGPIPE signal shall also be sent to the thread.	
72491				
72492				
72493			These functions may fail if:	
72494	OB XSR	[EINVAL]	The STREAM or multiplexer referenced by <i>fildest</i> is linked (directly or indirectly) downstream from a multiplexer.	
72495				
72496		[EIO]	A physical I/O error has occurred.	
72497		[ENOBUFFS]	Insufficient resources were available in the system to perform the operation.	
72498		[ENXIO]	A request was made of a nonexistent device, or the request was outside the capabilities of the device.	
72499				
72500	OB XSR	[ENXIO]	A hangup occurred on the STREAM being written to.	
72501	OB XSR		A write to a STREAMS file may fail if an error message has been received at the STREAM head. In this case, <i>errno</i> is set to the value included in the error message.	
72502				

The `write()` function may fail if:

[EACCES] A write was attempted on a socket and the calling process does not have appropriate privileges.

[ENETDOWN] A write was attempted on a socket and the local network interface used to reach the destination is down.

[ENETUNREACH]

A write was attempted on a socket and no route to the network is present.

EXAMPLES

Writing from a Buffer

The following example writes data from the buffer pointed to by `buf` to the file associated with the file descriptor `fd`.

```
#include <sys/types.h>
#include <string.h>
...
char buf[20];
size_t nbytes;
ssize_t bytes_written;
int fd;
...
strcpy(buf, "This is a test\n");
nbytes = strlen(buf);

bytes_written = write(fd, buf, nbytes);
...
```

APPLICATION USAGE

None.

RATIONALE

See also the RATIONALE section in `read()`.

An attempt to write to a pipe or FIFO has several major characteristics:

- *Atomic/non-atomic*: A write is atomic if the whole amount written in one operation is not interleaved with data from any other process. This is useful when there are multiple writers sending data to a single reader. Applications need to know how large a write request can be expected to be performed atomically. This maximum is called `{PIPE_BUF}`. This volume of POSIX.1-2008 does not say whether write requests for more than `{PIPE_BUF}` bytes are atomic, but requires that writes of `{PIPE_BUF}` or fewer bytes shall be atomic.
- *Blocking/immediate*: Blocking is only possible with `O_NONBLOCK` clear. If there is enough space for all the data requested to be written immediately, the implementation should do so. Otherwise, the calling thread may block; that is, pause until enough space is available for writing. The effective size of a pipe or FIFO (the maximum amount that can be written in one operation without blocking) may vary dynamically, depending on the implementation, so it is not possible to specify a fixed value for it.
- *Complete/partial/deferred*: A write request:

```
int fildes;
size_t nbyte;
```

```

72547     ssize_t ret;
72548     char *buf;

72549     ret = write(fildes, buf, nbyte);

```

72550 may return:

72551 Complete *ret=nbyte*

72552 Partial *ret<nbyte*

72553 This shall never happen if *nbyte*≤{PIPE_BUF}. If it does happen (with
72554 *nbyte*>{PIPE_BUF}), this volume of POSIX.1-2008 does not guarantee
72555 atomicity, even if *ret*≤{PIPE_BUF}, because atomicity is guaranteed according
72556 to the amount *requested*, not the amount *written*.

72557 Deferred: *ret=-1, errno=[EAGAIN]*

72558 This error indicates that a later request may succeed. It does not indicate that
72559 it *shall* succeed, even if *nbyte*≤{PIPE_BUF}, because if no process reads from
72560 the pipe or FIFO, the write never succeeds. An application could usefully
72561 count the number of times [EAGAIN] is caused by a particular value of
72562 *nbyte*>{PIPE_BUF} and perhaps do later writes with a smaller value, on the
72563 assumption that the effective size of the pipe may have decreased.

72564 Partial and deferred writes are only possible with O_NONBLOCK set.

72565 The relations of these properties are shown in the following tables:

72566 **Write to a Pipe or FIFO with O_NONBLOCK clear**

72567 Immediately Writable:	None	Some	<i>nbyte</i>
72568 <i>nbyte</i> ≤{PIPE_BUF}	Atomic blocking 72569 <i>nbyte</i>	Atomic blocking 72569 <i>nbyte</i>	Atomic immediate 72569 <i>nbyte</i>
72570 <i>nbyte</i> >{PIPE_BUF}	Blocking <i>nbyte</i>	Blocking <i>nbyte</i>	Blocking <i>nbyte</i>

72571 If the O_NONBLOCK flag is clear, a write request shall block if the amount writable
72572 immediately is less than that requested. If the flag is set (by *fcntl()*), a write request shall never
72573 block.

72574 **Write to a Pipe or FIFO with O_NONBLOCK set**

72575 Immediately Writable:	None	Some	<i>nbyte</i>
72576 <i>nbyte</i> ≤{PIPE_BUF}	-1, [EAGAIN]	-1, [EAGAIN]	Atomic <i>nbyte</i>
72577 <i>nbyte</i> >{PIPE_BUF}	-1, [EAGAIN]	< <i>nbyte</i> or -1, 72578 [EAGAIN]	≤ <i>nbyte</i> or -1, [EAGAIN]

72579 There is no exception regarding partial writes when O_NONBLOCK is set. With the exception
72580 of writing to an empty pipe, this volume of POSIX.1-2008 does not specify exactly when a partial
72581 write is performed since that would require specifying internal details of the implementation.
72582 Every application should be prepared to handle partial writes when O_NONBLOCK is set and
72583 the requested amount is greater than {PIPE_BUF}, just as every application should be prepared
72584 to handle partial writes on other kinds of file descriptors.

72585 The intent of forcing writing at least one byte if any can be written is to assure that each write
72586 makes progress if there is any room in the pipe. If the pipe is empty, {PIPE_BUF} bytes must be
72587 written; if not, at least some progress must have been made.

72588 Where this volume of POSIX.1-2008 requires -1 to be returned and *errno* set to [EAGAIN], most

historical implementations return zero (with the `O_NDELAY` flag set, which is the historical predecessor of `O_NONBLOCK`, but is not itself in this volume of POSIX.1-2008). The error indications in this volume of POSIX.1-2008 were chosen so that an application can distinguish these cases from end-of-file. While `write()` cannot receive an indication of end-of-file, `read()` can, and the two functions have similar return values. Also, some existing systems (for example, Eighth Edition) permit a write of zero bytes to mean that the reader should get an end-of-file indication; for those systems, a return value of zero from `write()` indicates a successful write of an end-of-file indication.

Implementations are allowed, but not required, to perform error checking for `write()` requests of zero bytes.

The concept of a `{PIPE_MAX}` limit (indicating the maximum number of bytes that can be written to a pipe in a single operation) was considered, but rejected, because this concept would unnecessarily limit application writing.

See also the discussion of `O_NONBLOCK` in `read()`.

Writes can be serialized with respect to other reads and writes. If a `read()` of file data can be proven (by any means) to occur after a `write()` of the data, it must reflect that `write()`, even if the calls are made by different processes. A similar requirement applies to multiple write operations to the same file position. This is needed to guarantee the propagation of data from `write()` calls to subsequent `read()` calls. This requirement is particularly significant for networked file systems, where some caching schemes violate these semantics.

Note that this is specified in terms of `read()` and `write()`. The XSI extensions `readv()` and `writew()` also obey these semantics. A new “high-performance” write analog that did not follow these serialization requirements would also be permitted by this wording. This volume of POSIX.1-2008 is also silent about any effects of application-level caching (such as that done by `stdio`).

This volume of POSIX.1-2008 does not specify the value of the file offset after an error is returned; there are too many cases. For programming errors, such as `[EBADF]`, the concept is meaningless since no file is involved. For errors that are detected immediately, such as `[EAGAIN]`, clearly the pointer should not change. After an interrupt or hardware error, however, an updated value would be very useful and is the behavior of many implementations.

This volume of POSIX.1-2008 does not specify behavior of concurrent writes to a file from multiple processes. Applications should use some form of concurrency control.

This volume of POSIX.1-2008 intentionally does not specify any `pwrite()` errors related to pipes, + FIFOs, and sockets other than `[ESPIPE]`.

FUTURE DIRECTIONS

None.

SEE ALSO

`chmod()`, `creat()`, `dup()`, `fcntl()`, `getrlimit()`, `lseek()`, `open()`, `pipe()`, `read()`, `ulimit()`, `writew()`

XBD `<limits.h>`, `<stropts.h>`, `<sys/uio.h>`, `<unistd.h>`

CHANGE HISTORY

First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 5

The DESCRIPTION is updated for alignment with the POSIX Realtime Extension and the POSIX Threads Extension.

Large File Summit extensions are added.

72634 The *pwrite()* function is added.

72635 Issue 6

72636 The DESCRIPTION states that the *write()* function does not block the thread. Previously this
72637 said “process” rather than “thread”.

72638 The DESCRIPTION and ERRORS sections are updated so that references to STREAMS are
72639 marked as part of the XSI STREAMS Option Group.

72640 The following new requirements on POSIX implementations derive from alignment with the
72641 Single UNIX Specification:

- 72642 • The DESCRIPTION now states that if *write()* is interrupted by a signal after it has
- 72643 successfully written some data, it returns the number of bytes written. In the POSIX.1-1988
- 72644 standard, it was optional whether *write()* returned the number of bytes written, or whether
- 72645 it returned `-1` with *errno* set to `[EINTR]`. This is a FIPS requirement.
- 72646 • The following changes are made to support large files:
 - 72647 — For regular files, no data transfer occurs past the offset maximum established in the
 - 72648 open file description associated with the *files*.
 - 72649 — A second `[EFBIG]` error condition is added.
- 72650 • The `[EIO]` error condition is added.
- 72651 • The `[EPIPE]` error condition is added for when a pipe has only one end open.
- 72652 • The `[ENXIO]` optional error condition is added.

72653 Text referring to sockets is added to the DESCRIPTION.

72654 The following changes were made to align with the IEEE P1003.1a draft standard:

- 72655 • The effect of reading zero bytes is clarified.

72656 The DESCRIPTION is updated for alignment with IEEE Std 1003.1j-2000 by specifying that
72657 *write()* results are unspecified for typed memory objects.

72658 The following error conditions are added for operations on sockets: `[EAGAIN]`,
72659 `[EWOULDBLOCK]`, `[ECONNRESET]`, `[ENOTCONN]`, and `[EPIPE]`.

72660 The `[EIO]` error is made optional.

72661 The `[ENOBUFFS]` error is added for sockets.

72662 The following error conditions are added for operations on sockets: `[EACCES]`, `[ENETDOWN]`,
72663 and `[ENETUNREACH]`.

72664 The *writenv()* function is split out into a separate reference page.

72665 IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/146 is applied, updating text in the
72666 ERRORS section from “a SIGPIPE signal is generated to the calling process” to “a SIGPIPE
72667 signal shall also be sent to the thread”.

72668 IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/147 is applied, making a correction to the
72669 RATIONALE.

72670 Issue 7

72671 The *pwrite()* function is moved from the XSI option to the Base.

72672 Functionality relating to the XSI STREAMS option is marked obsolescent.

72673 SD5-XSH-ERN-160 is applied, updating the DESCRIPTION to clarify the requirements for the

72674 *pwrite()* function, and to change the use of the phrase “file pointer” to “file offset”.

72675 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0742 [219], XSH/TC1-2008/0743 [215], +

72676 XSH/TC1-2008/0744 [79], and XSH/TC1-2008/0745 [215] are applied.

NAME

writev — write a vector

SYNOPSISXSI `#include <sys/uio.h>``ssize_t writev(int filides, const struct iovec *iov, int iovcnt);`**DESCRIPTION**

The *writev()* function shall be equivalent to *write()*, except as described below. The *writev()* function shall gather output data from the *iovcnt* buffers specified by the members of the *iov* array: *iov*[0], *iov*[1], ..., *iov*[*iovcnt*−1]. The *iovcnt* argument is valid if greater than 0 and less than or equal to {IOV_MAX}, as defined in **<limits.h>**.

Each *iovec* entry specifies the base address and length of an area in memory from which data should be written. The *writev()* function shall always write a complete area before proceeding to the next.

If *filides* refers to a regular file and all of the *iov_len* members in the array pointed to by *iov* are 0, *writev()* shall return 0 and have no other effect. For other file types, the behavior is unspecified.

If the sum of the *iov_len* values is greater than {SSIZE_MAX}, the operation shall fail and no data shall be transferred.

RETURN VALUE

Upon successful completion, *writev()* shall return the number of bytes actually written. Otherwise, it shall return a value of −1, the file-pointer shall remain unchanged, and *errno* shall be set to indicate an error.

ERRORS

Refer to *write()*.

In addition, the *writev()* function shall fail if:

[EINVAL] The sum of the *iov_len* values in the *iov* array would overflow an **ssize_t**.

The *writev()* function may fail and set *errno* to:

[EINVAL] The *iovcnt* argument was less than or equal to 0, or greater than {IOV_MAX}.

EXAMPLES**Writing Data from an Array**

The following example writes data from the buffers specified by members of the *iov* array to the file associated with the file descriptor *fd*.

```
#include <sys/types.h>
#include <sys/uio.h>
#include <unistd.h>
...
ssize_t bytes_written;
int fd;
char *buf0 = "short string\n";
char *buf1 = "This is a longer string\n";
char *buf2 = "This is the longest string in this example\n";
int iocnt;
struct iovec iov[3];
```

```
72719         iov[0].iov_base = buf0;
72720         iov[0].iov_len = strlen(buf0);
72721         iov[1].iov_base = buf1;
72722         iov[1].iov_len = strlen(buf1);
72723         iov[2].iov_base = buf2;
72724         iov[2].iov_len = strlen(buf2);
72725         ...
72726         iovcnt = sizeof(iov) / sizeof(struct iovec);
72727         bytes_written = writev(fd, iov, iovcnt);
72728         ...
```

72729 APPLICATION USAGE

72730 None.

72731 RATIONALE

72732 Refer to *write()*.

72733 FUTURE DIRECTIONS

72734 None.

72735 SEE ALSO

72736 *readv()*, *write()*

72737 XBD [<limits.h>](#), [<sys/uio.h>](#)

72738 CHANGE HISTORY

72739 First released in Issue 4, Version 2.

72740 Issue 6

72741 Split out from the *write()* reference page.

72742 **NAME**

72743 wscanf — convert formatted wide-character input

72744 **SYNOPSIS**

72745 #include <stdio.h>

72746 #include <wchar.h>

72747 int wscanf(const wchar_t *restrict *format*, ...);72748 **DESCRIPTION**72749 Refer to *fwscanf()*.

72750 **NAME**

72751 y0, y1, yn — Bessel functions of the second kind

72752 **SYNOPSIS**

```
72753 XSI      #include <math.h>
72754          double y0(double x);
72755          double y1(double x);
72756          double yn(int n, double x);
```

72757 **DESCRIPTION**

72758 The *y0()*, *y1()*, and *yn()* functions shall compute Bessel functions of *x* of the second kind of
 72759 orders 0, 1, and *n*, respectively.

72760 An application wishing to check for error situations should set *errno* to zero and call
 72761 *feclearexcept*(FE_ALL_EXCEPT) before calling these functions. On return, if *errno* is non-zero or
 72762 *fetestexcept*(FE_INVALID | FE_DIVBYZERO | FE_OVERFLOW | FE_UNDERFLOW) is non-
 72763 zero, an error has occurred.

72764 **RETURN VALUE**

72765 Upon successful completion, these functions shall return the relevant Bessel value of *x* of the
 72766 second kind.

72767 MXC If *x* is NaN, NaN shall be returned.

72768 If the *x* argument to these functions is negative, *-HUGE_VAL* or NaN shall be returned, and a
 72769 domain error may occur.

72770 If *x* is 0.0, *-HUGE_VAL* shall be returned and a pole error may occur.

72771 If the correct result would cause underflow, 0.0 shall be returned and a range error may occur.

72772 If the correct result would cause overflow, *-HUGE_VAL* or 0.0 shall be returned and a range
 72773 error may occur.

72774 **ERRORS**

72775 These functions may fail if:

72776 Domain Error The value of *x* is negative.

72777 If the integer expression (*math_errhandling* & MATH_ERRNO) is non-zero,
 72778 then *errno* shall be set to [EDOM]. If the integer expression (*math_errhandling*
 72779 & MATH_ERREXCEPT) is non-zero, then the invalid floating-point exception
 72780 shall be raised.

72781 Pole Error The value of *x* is zero.

72782 If the integer expression (*math_errhandling* & MATH_ERRNO) is non-zero,
 72783 then *errno* shall be set to [ERANGE]. If the integer expression
 72784 (*math_errhandling* & MATH_ERREXCEPT) is non-zero, then the divide-by-zero
 72785 floating-point exception shall be raised.

72786 Range Error The correct result would cause overflow.

72787 If the integer expression (*math_errhandling* & MATH_ERRNO) is non-zero,
 72788 then *errno* shall be set to [ERANGE]. If the integer expression
 72789 (*math_errhandling* & MATH_ERREXCEPT) is non-zero, then the overflow
 72790 floating-point exception shall be raised.

72791 Range Error The value of x is too large in magnitude, or the correct result would cause
72792 underflow.

72793 If the integer expression $(\text{math\_errhandling} \ \& \ \text{MATH\_ERRNO})$ is non-zero,
72794 then *errno* shall be set to [ERANGE]. If the integer expression
72795 $(\text{math\_errhandling} \ \& \ \text{MATH\_ERREXCEPT})$ is non-zero, then the underflow
72796 floating-point exception shall be raised.

72797 EXAMPLES

72798 None.

72799 APPLICATION USAGE

72800 On error, the expressions $(\text{math\_errhandling} \ \& \ \text{MATH\_ERRNO})$ and $(\text{math\_errhandling} \ \& \ \text{MATH\_ERREXCEPT})$
72801 are independent of each other, but at least one of them must be non-zero.

72802 RATIONALE

72803 None.

72804 FUTURE DIRECTIONS

72805 None.

72806 SEE ALSO

72807 [*feclearexcept\(\)*](#), [*fetestexcept\(\)*](#), [*isnan\(\)*](#), [*j0\(\)*](#)

72808 XBD [Section 4.19](#) (on page 116), [**<math.h>**](#)

72809 CHANGE HISTORY

72810 First released in Issue 1. Derived from Issue 1 of the SVID.

72811 Issue 5

72812 The DESCRIPTION is updated to indicate how an application should check for an error. This
72813 text was previously published in the APPLICATION USAGE section.

72814 Issue 6

72815 The normative text is updated to avoid use of the term “must” for application requirements.

72816 The RETURN VALUE and ERRORS sections are reworked for alignment of the error handling
72817 with the ISO/IEC 9899:1999 standard.

72818 IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/148 is applied, updating the RETURN
72819 VALUE and ERRORS sections. The changes are made for consistency with the general rules
72820 stated in “Treatment of Error Conditions for Mathematical Functions” in the Base Definitions
72821 volume of POSIX.1-2008.

72822 Issue 7

72823 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0746 [68] is applied.

+

72824

Technical Standard

72825

Vol. 3:

72826

Shell and Utilities, Issue 7

72827

The Open Group

72828

The Institute of Electrical and Electronics Engineers, Inc.

72829

72830

Introduction

72831

72832

The Shell and Utilities volume of POSIX.1-2008 describes the commands and utilities offered to application programs by POSIX-conformant systems.

72833

1.1 Relationship to Other Documents

72834

1.1.1 System Interfaces

72835

72836

72837

72838

72839

72840

This subsection describes some of the features provided by the System Interfaces volume of POSIX.1-2008 that are assumed to be globally available on all systems conforming to this volume of POSIX.1-2008. This subsection does not attempt to detail all of the features defined in the System Interfaces volume of POSIX.1-2008 that are required by all of the utilities defined in this volume of POSIX.1-2008; the utility and function descriptions point out additional functionality required to provide the corresponding specific features needed by each.

72841

72842

72843

The following subsections describe frequently used concepts. Many of these concepts are described in the Base Definitions volume of POSIX.1-2008. Utility and function description statements override these defaults when appropriate.

72844

1.1.1.1 Process Attributes

72845

72846

The following process attributes, as described in the System Interfaces volume of POSIX.1-2008, are assumed to be supported for all processes in this volume of POSIX.1-2008:

72847

72848

72849

72850

72851

72852

72853

72854

Controlling Terminal	Real Group ID
Current Working Directory	Real User ID
Effective Group ID	Root Directory
Effective User ID	Saved Set-Group-ID
File Descriptors	Saved Set-User-ID
File Mode Creation Mask	Session Membership
Process Group ID	Supplementary Group IDs
Process ID	

72855

A conforming implementation may include additional process attributes.

72856

1.1.1.2 Concurrent Execution of Processes

72857

72858

The following functionality of the *fork()* function defined in the System Interfaces volume of POSIX.1-2008 shall be available on all systems conforming to this volume of POSIX.1-2008:

72859

72860

1. Independent processes shall be capable of executing independently without either process terminating.

72861 2. A process shall be able to create a new process with all of the attributes referenced in
 72862 [Section 1.1.1.1](#) (on page 2301), determined according to the semantics of a call to the *fork()*
 72863 function defined in the System Interfaces volume of POSIX.1-2008 followed by a call in
 72864 the child process to one of the *exec* functions defined in the System Interfaces volume of
 72865 POSIX.1-2008.

72866 1.1.1.3 File Access Permissions

72867 The file access control mechanism described by XBD [Section 4.4](#) (on page 108) shall apply to all
 72868 files on an implementation conforming to this volume of POSIX.1-2008.

72869 1.1.1.4 File Read, Write, and Creation

72870 If a file that does not exist is to be written, it shall be created as described below, unless the
 72871 utility description states otherwise.

72872 When a file that does not exist is created, the following features defined in the System Interfaces
 72873 volume of POSIX.1-2008 shall apply unless the utility or function description states otherwise:

- 72874 1. The user ID of the file shall be set to the effective user ID of the calling process.
- 72875 2. The group ID of the file shall be set to the effective group ID of the calling process or the
 72876 group ID of the directory in which the file is being created.
- 72877 3. If the file is a regular file, the permission bits of the file shall be set to:

72878 S_IROTH | S_IWOTH | S_IRGRP | S_IWGRP | S_IRUSR | S_IWUSR

72879 (see the description of *File Modes* in XBD [Chapter 13](#) (on page 219), `<sys/stat.h>`) except
 72880 that the bits specified by the file mode creation mask of the process shall be cleared. If the
 72881 file is a directory, the permission bits shall be set to:

72882 S_IRWXU | S_IRWXG | S_IRWXO

72883 except that the bits specified by the file mode creation mask of the process shall be
 72884 cleared.

- 72885 4. The last data access, last data modification, and last file status change timestamps of the
 72886 file shall be updated as specified in XBD [Section 4.8](#) (on page 109).
- 72887 5. If the file is a directory, it shall be an empty directory; otherwise, the file shall have length
 72888 zero.
- 72889 6. If the file is a symbolic link, the effect shall be undefined unless the {POSIX2_SYMLINKS}
 72890 variable is in effect for the directory in which the symbolic link would be created.
- 72891 7. Unless otherwise specified, the file created shall be a regular file.

72892 When an attempt is made to create a file that already exists, the utility shall take the action
 72893 indicated in [Table 1-1](#) (on page 2303) corresponding to the type of the file the utility is trying to
 72894 create and the type of the existing file, unless the utility description states otherwise.

Table 1-1 Actions when Creating a File that Already Exists

Existing Type	New Type												Function
	B	C	D	F	L	M	P	Q	R	S	T	Creating New	
A <i>fattach()</i> -ed STREAM	F	F	F	F	F	—	—	—	OF	—	U	N/A	
B Block Special	F	F	F	F	F	U	U	U	OF	U	U	<i>mknod()</i> **	
C Character Special	F	F	F	F	F	U	U	U	OF	U	U	<i>mknod()</i> **	
D Directory	F	F	F	F	F	—	—	—	F	—	U	<i>mkdir()</i>	
F FIFO Special File	F	F	F	F	F	—	—	—	O	—	U	<i>mkfifo()</i>	
L Symbolic Link	F	F	F	F	F	—	—	—	FL	—	U	<i>symlink()</i>	
M Shared Memory	F	F	F	F	F	—	—	—	—	—	U	<i>shm_open()</i>	
P Semaphore	F	F	F	F	F	—	—	—	—	—	U	<i>sem_open()</i>	
Q Message Queue	F	F	F	F	F	—	—	—	—	—	U	<i>mq_open()</i>	
R Regular File	F	F	F	F	F	—	—	—	RF	—	U	<i>open()</i>	
S Socket	F	F	F	F	F	—	—	—	—	—	U	<i>bind()</i>	
T Typed Memory	F	F	F	F	F	U	U	U	U	U	U	*	

The following codes are used in [Table 1-1](#):

F Fail. The attempt to create the new file shall fail and the utility shall either continue with its operation or exit immediately with a non-zero exit status, depending on the description of the utility.

FL Follow link. Unless otherwise specified, the symbolic link shall be followed as specified for pathname resolution, and the operation performed shall be as if the target of the symbolic link (after all resolution) had been named. If the target of the symbolic link does not exist, it shall be as if that nonexistent target had been named directly.

O Open FIFO. When attempting to create a regular file, and the existing file is a FIFO special file:

1. If the FIFO is not already open for reading, the attempt shall block until the FIFO is opened for reading.
2. Once the FIFO is open for reading, the utility shall open the FIFO for writing and continue with its operation.

OF The named file shall be opened with the consequences defined for that file type.

RF Regular file. When attempting to create a regular file, and the existing file is a regular file:

1. The user ID, group ID, and permission bits of the file shall not be changed.
2. The file shall be truncated to zero length.
3. The last data modification and last file status change timestamps shall be marked for update.

— The effect is implementation-defined unless specified by the utility description.

U The effect is unspecified unless specified by the utility description.

***** There is no portable way to create a file of this type.

****** Not portable.

When a file is to be appended, the file shall be opened in a manner equivalent to using the `O_APPEND` flag, without the `O_TRUNC` flag, in the *open()* function defined in the System Interfaces volume of POSIX.1-2008.

72937 When a file is to be read or written, the file shall be opened with an access mode corresponding
 72938 to the operation to be performed. If file access permissions deny access, the requested operation
 72939 shall fail.

72940 1.1.1.5 File Removal

72941 When a directory that is the root directory or current working directory of any process is
 72942 removed, the effect is implementation-defined. If file access permissions deny access, the
 72943 requested operation shall fail. Otherwise, when a file is removed:

- 72944 1. Its directory entry shall be removed from the file system.
- 72945 2. The link count of the file shall be decremented.
- 72946 3. If the file is an empty directory (see XBD [Section 3.144](#), on page 56):
 - 72947 a. If no process has the directory open, the space occupied by the directory shall be
 72948 freed and the directory shall no longer be accessible.
 - 72949 b. If one or more processes have the directory open, the directory contents shall be
 72950 preserved until all references to the file have been closed.
- 72951 4. If the file is a directory that is not empty, the last file status change timestamp shall be
 72952 marked for update.
- 72953 5. If the file is not a directory:
 - 72954 a. If the link count becomes zero:
 - 72955 i. If no process has the file open, the space occupied by the file shall be freed
 72956 and the file shall no longer be accessible.
 - 72957 ii. If one or more processes have the file open, the file contents shall be
 72958 preserved until all references to the file have been closed.
 - 72959 b. If the link count is not reduced to zero, the last file status change timestamp shall
 72960 be marked for update.
- 72961 6. The last data modification and last file status change timestamps of the containing
 72962 directory shall be marked for update.

72963 1.1.1.6 File Time Values

72964 All files shall have the three time values described by XBD [Section 4.8](#) (on page 109).

72965 1.1.1.7 File Contents

72966 When a reference is made to the contents of a file, *pathname*, this means the equivalent of all of
 72967 the data placed in the space pointed to by *buf* when performing the *read()* function calls in the
 72968 following operations defined in the System Interfaces volume of POSIX.1-2008:

```
72969 while (read (fildes, buf, nbytes) > 0)
72970     ;
```

72971 If the file is indicated by a *pathname* *pathname*, the file descriptor shall be determined by the
 72972 equivalent of the following operation defined in the System Interfaces volume of POSIX.1-2008:

```
72973 fildes = open (pathname, O_RDONLY);
```

72974 The value of *nbytes* in the above sequence is unspecified; if the file is of a type where the data

72975 returned by *read()* would vary with different values, the value shall be one that results in the
 72976 most data being returned.

72977 If the *read()* function calls would return an error, it is unspecified whether the contents of the file
 72978 are considered to include any data from offsets in the file beyond where the error would be
 72979 returned.

72980 1.1.1.8 Pathname Resolution

72981 The pathname resolution algorithm, described by XBD [Section 4.12](#) (on page 111), shall be used
 72982 by implementations conforming to this volume of POSIX.1-2008; see also XBD [Section 4.5](#) (on
 72983 page 108).

72984 1.1.1.9 Changing the Current Working Directory

72985 When the current working directory (see XBD [Section 3.122](#), on page 53) is to be changed, unless
 72986 the utility or function description states otherwise, the operation shall succeed unless a call to
 72987 the *chdir()* function defined in the System Interfaces volume of POSIX.1-2008 would fail when
 72988 invoked with the new working directory pathname as its argument.

72989 1.1.1.10 Establish the Locale

72990 The functionality of the *setlocale()* function defined in the System Interfaces volume of
 72991 POSIX.1-2008 shall be available on all systems conforming to this volume of POSIX.1-2008; that
 72992 is, utilities that require the capability of establishing an international operating environment
 72993 shall be permitted to set the specified category of the international environment.

72994 1.1.1.11 Actions Equivalent to Functions

72995 Some utility descriptions specify that a utility performs actions equivalent to a function defined
 72996 in the System Interfaces volume of POSIX.1-2008. Such specifications require only that the
 72997 external effects be equivalent, not that any effect within the utility and visible only to the utility
 72998 be equivalent.

72999 1.1.2 Concepts Derived from the ISO C Standard

73000 Some of the standard utilities perform complex data manipulation using their own procedure
 73001 and arithmetic languages, as defined in their EXTENDED DESCRIPTION or OPERANDS
 73002 sections. Unless otherwise noted, the arithmetic and semantic concepts (precision, type
 73003 conversion, control flow, and so on) shall be equivalent to those defined in the ISO C standard,
 73004 as described in the following sections. Note that there is no requirement that the standard
 73005 utilities be implemented in any particular programming language.

73006 1.1.2.1 Arithmetic Precision and Operations

73007 Integer variables and constants, including the values of operands and option-arguments, used
 73008 by the standard utilities listed in this volume of POSIX.1-2008 shall be implemented as
 73009 equivalent to the ISO C standard **signed long** data type; floating point shall be implemented as
 73010 equivalent to the ISO C standard **double** type. Conversions between types shall be as described
 73011 in the ISO C standard. All variables shall be initialized to zero if they are not otherwise assigned
 73012 by the input to the application.

Arithmetic operators and control flow keywords shall be implemented as equivalent to those in the cited ISO C standard section, as listed in [Table 1-2](#).

Table 1-2 Selected ISO C Standard Operators and Control Flow Keywords

Operation	ISO C Standard Equivalent Reference
()	Section 6.5.1, Primary Expressions
postfix ++ postfix --	Section 6.5.2, Postfix Operators
unary + unary - prefix ++ prefix -- ~ ! sizeof()	Section 6.5.3, Unary Operators
* / %	Section 6.5.5, Multiplicative Operators
+ -	Section 6.5.6, Additive Operators
<< >>	Section 6.5.7, Bitwise Shift Operators
<, <= >, >=	Section 6.5.8, Relational Operators
= = !=	Section 6.5.9, Equality Operators
&	Section 6.5.10, Bitwise AND Operator
^	Section 6.5.11, Bitwise Exclusive OR Operator
	Section 6.5.12, Bitwise Inclusive OR Operator
&&	Section 6.5.13, Logical AND Operator
	Section 6.5.14, Logical OR Operator
expr?expr:expr	Section 6.5.15, Conditional Operator
=, *=, /=, % =, +=, -= <<=, >>=, &=, ^=, =	Section 6.5.16, Assignment Operators
if () if () ... else switch ()	Section 6.8.4, Selection Statements
while () do ... while () for ()	Section 6.8.5, Iteration Statements
goto continue break return	Section 6.8.6, Jump Statements

73056 The evaluation of arithmetic expressions shall be equivalent to that described in Section 6.5,
73057 Expressions, of the ISO C standard.

73058 1.1.2.2 Mathematical Functions

73059 Any mathematical functions with the same names as those in the following sections of the ISO C
73060 standard:

- 73061 • Section 7.12, Mathematics, `<math.h>`
- 73062 • Section 7.20.2, Pseudo-Random Sequence Generation Functions

73063 shall be implemented to return the results equivalent to those returned from a call to the
73064 corresponding function described in the ISO C standard.

73065 1.2 Utility Limits

73066 This section lists magnitude limitations imposed by a specific implementation. The braces
73067 notation, {LIMIT}, is used in this volume of POSIX.1-2008 to indicate these values, but the braces
73068 are not part of the name.

73069 **Table 1-3** Utility Limit Minimum Values

Name	Description	Value
{POSIX2_BC_BASE_MAX}	The maximum <i>obase</i> value allowed by the <i>bc</i> utility.	99
{POSIX2_BC_DIM_MAX}	The maximum number of elements permitted in an array by the <i>bc</i> utility.	2 048
{POSIX2_BC_SCALE_MAX}	The maximum <i>scale</i> value allowed by the <i>bc</i> utility.	99
{POSIX2_BC_STRING_MAX}	The maximum length of a string constant accepted by the <i>bc</i> utility.	1 000
{POSIX2_COLL_WEIGHTS_MAX}	The maximum number of weights that can be assigned to an entry of the <i>LC_COLLATE</i> order keyword in the locale definition file; see the border_start keyword in XBD Section 7.3.2 (on page 147).	2
{POSIX2_EXPR_NEST_MAX}	The maximum number of expressions that can be nested within parentheses by the <i>expr</i> utility.	32
{POSIX2_LINE_MAX}	Unless otherwise noted, the maximum length, in bytes, of the input line of a utility (either standard input or another file), when the utility is described as processing text files. The length includes room for the trailing <code><newline></code> .	2 048
{POSIX2_RE_DUP_MAX}	The maximum number of repeated occurrences of a BRE permitted when using the interval notation <code>\{m,n\}</code> ; see XBD Section 9.3.6 (on page 186).	255

73095 The values specified in [Table 1-3](#) represent the lowest values conforming implementations shall
73096 provide and, consequently, the largest values on which an application can rely without further

enquiries, as described below. These values shall be accessible to applications via the *getconf* utility (see *getconf*, on page 2796).

Implementations may provide more liberal, or less restrictive, values than shown in [Table 1-3](#) (on page 2307). These possibly more liberal values are accessible using the symbols in [Table 1-4](#).

The *sysconf()* function defined in the System Interfaces volume of POSIX.1-2008 or the *getconf* utility return the value of each symbol on each specific implementation. The value so retrieved is the largest, or most liberal, value that is available throughout the session lifetime, as determined at session creation. The literal names shown in the table apply only to the *getconf* utility; the high-level language binding describes the exact form of each name to be used by the interfaces in that binding.

All numeric limits defined by the System Interfaces volume of POSIX.1-2008, such as {PATH_MAX}, shall also apply to this volume of POSIX.1-2008. All the utilities defined by this volume of POSIX.1-2008 are implicitly limited by these values, unless otherwise noted in the utility descriptions.

It is not guaranteed that the application can actually reach the specified limit of an implementation in any given case, or at all, as a lack of virtual memory or other resources may prevent this. The limit value indicates only that the implementation does not specifically impose any arbitrary, more restrictive limit.

Table 1-4 Symbolic Utility Limits

Name	Description	Minimum Value
{BC_BASE_MAX}	The maximum <i>obase</i> value allowed by the <i>bc</i> utility.	{POSIX2_BC_BASE_MAX}
{BC_DIM_MAX}	The maximum number of elements permitted in an array by the <i>bc</i> utility.	{POSIX2_BC_DIM_MAX}
{BC_SCALE_MAX}	The maximum <i>scale</i> value allowed by the <i>bc</i> utility.	{POSIX2_BC_SCALE_MAX}
{BC_STRING_MAX}	The maximum length of a string constant accepted by the <i>bc</i> utility.	{POSIX2_BC_STRING_MAX}
{COLL_WEIGHTS_MAX}	The maximum number of weights that can be assigned to an entry of the <i>LC_COLLATE</i> order keyword in the locale definition file; see the order_start keyword in XBD Section 7.3.2 (on page 147).	{POSIX2_COLL_WEIGHTS_MAX}
{EXPR_NEST_MAX}	The maximum number of expressions that can be nested within parentheses by the <i>expr</i> utility.	{POSIX2_EXPR_NEST_MAX}

Name	Description	Minimum Value
{LINE_MAX}	Unless otherwise noted, the maximum length, in bytes, of the input line of a utility (either standard input or another file), when the utility is described as processing text files. The length includes room for the trailing <newline>.	{POSIX2_LINE_MAX}
{RE_DUP_MAX}	The maximum number of repeated occurrences of a BRE permitted when using the interval notation $\{m,n\}$; see XBD Section 9.3.6 (on page 186).	{POSIX2_RE_DUP_MAX}

The following value may be a constant within an implementation or may vary from one pathname to another.

{POSIX2_SYMLINKS}

When referring to a directory, the system supports the creation of symbolic links within that directory; for non-directory files, the meaning of {POSIX2_SYMLINKS} is undefined.

1.3 Grammar Conventions

Portions of this volume of POSIX.1-2008 are expressed in terms of a special grammar notation. It is used to portray the complex syntax of certain program input. The grammar is based on the syntax used by the *yacc* utility. However, it does not represent fully functional *yacc* input, suitable for program use; the lexical processing and all semantic requirements are described only in textual form. The grammar is not based on source used in any traditional implementation and has not been tested with the semantic code that would normally be required to accompany it. Furthermore, there is no implication that the partial *yacc* code presented represents the most efficient, or only, means of supporting the complex syntax within the utility. Implementations may use other programming languages or algorithms, as long as the syntax supported is the same as that represented by the grammar.

The following typographical conventions are used in the grammar; they have no significance except to aid in reading.

- The identifiers for the reserved words of the language are shown with a leading capital letter. (These are terminals in the grammar; for example, **While**, **Case**.)
- The identifiers for terminals in the grammar are all named with uppercase letters and underscores; for example, **NEWLINE**, **ASSIGN_OP**, **NAME**.
- The identifiers for non-terminals are all lowercase.

1.4 Utility Description Defaults

This section describes all of the subsections used within the utility descriptions, including:

- Intended usage of the section
- Global defaults that affect all the standard utilities
- The meanings of notations used in this volume of POSIX.1-2008 that are specific to individual utility sections

NAME

This section gives the name or names of the utility and briefly states its purpose.

SYNOPSIS

The SYNOPSIS section summarizes the syntax of the calling sequence for the utility, including options, option-arguments, and operands. Standards for utility naming are described in XBD [Section 12.2](#) (on page 215); for describing the utility's arguments in XBD [Section 12.1](#) (on page 213).

DESCRIPTION

The DESCRIPTION section describes the actions of the utility. If the utility has a very complex set of subcommands or its own procedural language, an EXTENDED DESCRIPTION section is also provided. Most explanations of optional functionality are omitted here, as they are usually explained in the OPTIONS section.

As stated in [Section 1.1.1.11](#) (on page 2305), some functions are described in terms of equivalent functionality. When specific functions are cited, the implementation shall provide equivalent functionality including side-effects associated with successful execution of the function. The treatment of errors and intermediate results from the individual functions cited is generally not specified by this volume of POSIX.1-2008. See the utility's EXIT STATUS and CONSEQUENCES OF ERRORS sections for all actions associated with errors encountered by the utility.

OPTIONS

The OPTIONS section describes the utility options and option-arguments, and how they modify the actions of the utility. Standard utilities that have options either fully comply with XBD [Section 12.2](#) (on page 215) or describe all deviations. Apparent disagreements between functionality descriptions in the OPTIONS and DESCRIPTION (or EXTENDED DESCRIPTION) sections are always resolved in favor of the OPTIONS section.

Each OPTIONS section that uses the phrase “The ... utility shall conform to the Utility Syntax Guidelines ...” refers only to the use of the utility as specified by this volume of POSIX.1-2008; implementation extensions should also conform to the guidelines, but may allow exceptions for historical practice.

Unless otherwise stated in the utility description, when given an option unrecognized by the implementation, or when a required option-argument is not provided, standard utilities shall issue a diagnostic message to standard error and exit with a non-zero exit status.

All utilities in this volume of POSIX.1-2008 shall be capable of processing arguments using eight-bit transparency.

Default Behavior: When this section is listed as “None.”, it means that the implementation need not support any options. Standard utilities that do not accept options, but that do accept operands, shall recognize “--” as a first argument to be discarded.

The requirement for recognizing "--" is because conforming applications need a way to shield their operands from any arbitrary options that the implementation may provide as an extension. For example, if the standard utility *foo* is listed as taking no options, and the application needed to give it a pathname with a leading <hyphen>, it could safely do it as:

```
foo -- -myfile
```

and avoid any problems with **-m** used as an extension.

OPERANDS

The OPERANDS section describes the utility operands, and how they affect the actions of the utility. Apparent disagreements between functionality descriptions in the OPERANDS and DESCRIPTION (or EXTENDED DESCRIPTION) sections shall be resolved in favor of the OPERANDS section.

If an operand naming a file can be specified as '-', which means to use the standard input instead of a named file, this is explicitly stated in this section. Unless otherwise stated, the use of multiple instances of '-' to mean standard input in a single command produces unspecified results.

Unless otherwise stated, the standard utilities that accept operands shall process those operands in the order specified in the command line.

Default Behavior: When this section is listed as "None.", it means that the implementation need not support any operands.

STDIN

The STDIN section describes the standard input of the utility. This section is frequently merely a reference to the following section, as many utilities treat standard input and input files in the same manner. Unless otherwise stated, all restrictions described in the INPUT FILES section shall apply to this section as well.

Use of a terminal for standard input can cause any of the standard utilities that read standard input to stop when used in the background. For this reason, applications should not use interactive features in scripts to be placed in the background.

The specified standard input format of the standard utilities shall not depend on the existence or value of the environment variables defined in this volume of POSIX.1-2008, except as provided by this volume of POSIX.1-2008.

Default Behavior: When this section is listed as "Not used.", it means that the standard input shall not be read when the utility is used as described by this volume of POSIX.1-2008.

INPUT FILES

The INPUT FILES section describes the files, other than the standard input, used as input by the utility. It includes files named as operands and option-arguments as well as other files that are referred to, such as start-up and initialization files, databases, and so on. Commonly-used files are generally described in one place and cross-referenced by other utilities.

All utilities in this volume of POSIX.1-2008 shall be capable of processing input files using eight-bit transparency.

When a standard utility reads a seekable input file and terminates without an error before it reaches end-of-file, the utility shall ensure that the file offset in the open file description is properly positioned just past the last byte processed by the utility. For files that are not seekable, the state of the file offset in the open file description for that

file is unspecified. A conforming application shall not assume that the following three commands are equivalent:

```
tail -n +2 file
(sed -n 1q; cat) < file
cat file | (sed -n 1q; cat)
```

The second command is equivalent to the first only when the file is seekable. The third command leaves the file offset in the open file description in an unspecified state. Other utilities, such as *head*, *read*, and *sh*, have similar properties.

Some of the standard utilities, such as filters, process input files a line or a block at a time and have no restrictions on the maximum input file size. Some utilities may have size limitations that are not as obvious as file space or memory limitations. Such limitations should reflect resource limitations of some sort, not arbitrary limits set by implementors. Implementations shall document those utilities that are limited by constraints other than file system space, available memory, and other limits specifically cited by this volume of POSIX.1-2008, and identify what the constraint is and indicate a way of estimating when the constraint would be reached. Similarly, some utilities descend the directory tree (recursively). Implementations shall also document any limits that they may have in descending the directory tree that are beyond limits cited by this volume of POSIX.1-2008.

When an input file is described as a “text file”, the utility produces undefined results if given input that is not from a text file, unless otherwise stated. Some utilities (for example, *make*, *read*, *sh*) allow for continued input lines using an escaped <newline> convention; unless otherwise stated, the utility need not be able to accumulate more than {LINE_MAX} bytes from a set of multiple, continued input lines. Thus, for a conforming application the total of all the continued lines in a set cannot exceed {LINE_MAX}. If a utility using the escaped <newline> convention detects an end-of-file condition immediately after an escaped <newline>, the results are unspecified.

Record formats are described in a notation similar to that used by the C-language function, *printf()*. See XBD Chapter 5 (on page 121) for a description of this notation. The format description is intended to be sufficiently rigorous to allow other applications to generate these input files. However, since <blank>s can legitimately be included in some of the fields described by the standard utilities, particularly in locales other than the POSIX locale, this intent is not always realized.

Default Behavior: When this section is listed as “None.”, it means that no input files are required to be supplied when the utility is used as described by this volume of POSIX.1-2008.

ENVIRONMENT VARIABLES

The ENVIRONMENT VARIABLES section lists what variables affect the utility’s execution.

The entire manner in which environment variables described in this volume of POSIX.1-2008 affect the behavior of each utility is described in the ENVIRONMENT VARIABLES section for that utility, in conjunction with the global effects of the *LANG*, *LC_ALL*, and *NLSPATH* environment variables described in XBD Chapter 8 (on page 173). The existence or value of environment variables described in this volume of POSIX.1-2008 shall not otherwise affect the specified behavior of the standard utilities. Any effects of the existence or value of environment variables not described by this volume of POSIX.1-2008 upon the standard utilities are unspecified.

For those standard utilities that use environment variables as a means for selecting a

utility to execute (such as CC in *make*), the string provided to the utility is subjected to the path search described for *PATH* in XBD [Chapter 8](#) (on page 173).

All utilities in this volume of POSIX.1-2008 shall be capable of processing environment variable names and values using eight-bit transparency.

Default Behavior: When this section is listed as “None.”, it means that the behavior of the utility is not directly affected by environment variables described by this volume of POSIX.1-2008 when the utility is used as described by this volume of POSIX.1-2008.

ASYNCHRONOUS EVENTS

The ASYNCHRONOUS EVENTS section lists how the utility reacts to such events as signals and what signals are caught.

Default Behavior: When this section is listed as “Default.”, or it refers to “the standard action for all other signals; see [Section 1.4](#) (on page 2310)” it means that the action taken as a result of the signal shall be one of the following:

1. The action shall be that inherited from the parent according to the rules of inheritance of signal actions defined in the System Interfaces volume of POSIX.1-2008.
2. When no action has been taken to change the default, the default action shall be that specified by the System Interfaces volume of POSIX.1-2008.
3. The result of the utility’s execution is as if default actions had been taken.

A utility is permitted to catch a signal, perform some additional processing (such as deleting temporary files), restore the default signal action (or action inherited from the parent process), and resignal itself.

STDOUT

The STDOUT section completely describes the standard output of the utility. This section is frequently merely a reference to the following section, OUTPUT FILES, because many utilities treat standard output and output files in the same manner.

Use of a terminal for standard output may cause any of the standard utilities that write standard output to stop when used in the background. For this reason, applications should not use interactive features in scripts to be placed in the background.

Record formats are described in a notation similar to that used by the C-language function, *printf()*. See XBD [Chapter 5](#) (on page 121) for a description of this notation.

The specified standard output of the standard utilities shall not depend on the existence or value of the environment variables defined in this volume of POSIX.1-2008, except as provided by this volume of POSIX.1-2008.

Some of the standard utilities describe their output using the verb *display*, defined in XBD [Section 3.133](#) (on page 54). Output described in the STDOUT sections of such utilities may be produced using means other than standard output. When standard output is directed to a terminal, the output described shall be written directly to the terminal. Otherwise, the results are undefined.

Default Behavior: When this section is listed as “Not used.”, it means that the standard output shall not be written when the utility is used as described by this volume of POSIX.1-2008.

STDERR

The STDERR section describes the standard error output of the utility. Only those messages that are purposely sent by the utility are described.

73363 Use of a terminal for standard error may cause any of the standard utilities that write
 73364 standard error output to stop when used in the background. For this reason,
 73365 applications should not use interactive features in scripts to be placed in the
 73366 background.

73367 The format of diagnostic messages for most utilities is unspecified, but the language
 73368 and cultural conventions of diagnostic and informative messages whose format is
 73369 unspecified by this volume of POSIX.1-2008 should be affected by the setting of
 73370 XSI `LC_MESSAGES` and `NLSPATH`.

73371 The specified standard error output of standard utilities shall not depend on the
 73372 existence or value of the environment variables defined in this volume of POSIX.1-2008,
 73373 except as provided by this volume of POSIX.1-2008.

73374 **Default Behavior:** When this section is listed as “The standard error shall be used only
 73375 for diagnostic messages.”, it means that, unless otherwise stated, the diagnostic
 73376 messages shall be sent to the standard error only when the exit status indicates that an
 73377 error occurred and the utility is used as described by this volume of POSIX.1-2008.

73378 When this section is listed as “Not used.”, it means that the standard error shall not be
 73379 used when the utility is used as described in this volume of POSIX.1-2008.

73380 OUTPUT FILES

73381 The OUTPUT FILES section completely describes the files created or modified by the
 73382 utility. Temporary or system files that are created for internal usage by this utility or
 73383 other parts of the implementation (for example, spool, log, and audit files) are not
 73384 described in this, or any, section. The utilities creating such files and the names of such
 73385 files are unspecified. If applications are written to use temporary or intermediate files,
 73386 they should use the `TMPDIR` environment variable, if it is set and represents an
 73387 accessible directory, to select the location of temporary files.

73388 Implementations shall ensure that temporary files, when used by the standard utilities,
 73389 are named so that different utilities or multiple instances of the same utility can operate
 73390 simultaneously without regard to their working directories, or any other process
 73391 characteristic other than process ID. There are two exceptions to this rule:

- 73392 1. Resources for temporary files other than the name space (for example, disk
 73393 space, available directory entries, or number of processes allowed) are not
 73394 guaranteed.
- 73395 2. Certain standard utilities generate output files that are intended as input for
 73396 other utilities (for example, *lex* generates `lex.yy.c`), and these cannot have unique
 73397 names. These cases are explicitly identified in the descriptions of the respective
 73398 utilities.

73399 Any temporary file created by the implementation shall be removed by the
 73400 implementation upon a utility’s successful exit, exit because of errors, or before
 73401 termination by any of the `SIGHUP`, `SIGINT`, or `SIGTERM` signals, unless specified
 73402 otherwise by the utility description.

73403 Receipt of the `SIGQUIT` signal should generally cause termination (unless in some
 73404 debugging mode) that would bypass any attempted recovery actions.

73405 Record formats are described in a notation similar to that used by the C-language
 73406 function, `printf()`; see XBD [Chapter 5](#) (on page 121) for a description of this notation.

73407 **Default Behavior:** When this section is listed as “None.”, it means that no files are
 73408 created or modified as a consequence of direct action on the part of the utility when the
 73409 utility is used as described by this volume of POSIX.1-2008. However, the utility may

create or modify system files, such as log files, that are outside the utility's normal execution environment.

EXTENDED DESCRIPTION

The EXTENDED DESCRIPTION section provides a place for describing the actions of very complicated utilities, such as text editors or language processors, which typically have elaborate command languages.

Default Behavior: When this section is listed as "None.", no further description is necessary.

EXIT STATUS

The EXIT STATUS section describes the values the utility shall return to the calling program, or shell, and the conditions that cause these values to be returned. Usually, utilities return zero for successful completion and values greater than zero for various error conditions. If specific numeric values are listed in this section, the system shall use those values for the errors described. In some cases, status values are listed more loosely, such as >0. A strictly conforming application shall not rely on any specific value in the range shown and shall be prepared to receive any value in the range.

For example, a utility may list zero as a successful return, 1 as a failure for a specific reason, and >1 as "an error occurred". In this case, unspecified conditions may cause a 2 or 3, or other value, to be returned. A conforming application should be written so that it tests for successful exit status values (zero in this case), rather than relying upon the single specific error value listed in this volume of POSIX.1-2008. In that way, it has maximum portability, even on implementations with extensions.

Unspecified error conditions may be represented by specific values not listed in this volume of POSIX.1-2008.

CONSEQUENCES OF ERRORS

The CONSEQUENCES OF ERRORS section describes the effects on the environment, file systems, process state, and so on, when error conditions occur. It does not describe error messages produced or exit status values used.

The many reasons for failure of a utility are generally not specified by the utility descriptions. Utilities may terminate prematurely if they encounter: invalid usage of options, arguments, or environment variables; invalid usage of the complex syntaxes expressed in EXTENDED DESCRIPTION sections; difficulties accessing, creating, reading, or writing files; or difficulties associated with the privileges of the process.

The following shall apply to each utility, unless otherwise stated:

- If the requested action cannot be performed on an operand representing a file, directory, user, process, and so on, the utility shall issue a diagnostic message to standard error and continue processing the next operand in sequence, but the final exit status shall be returned as non-zero.

For a utility that recursively traverses a file hierarchy (such as *find* or *chown -R*), if the requested action cannot be performed on a file or directory encountered in the hierarchy, the utility shall issue a diagnostic message to standard error and continue processing the remaining files in the hierarchy, but the final exit status shall be returned as non-zero.

- If the requested action characterized by an option or option-argument cannot be performed, the utility shall issue a diagnostic message to standard error and the exit status returned shall be non-zero.

- When an unrecoverable error condition is encountered, the utility shall exit with a non-zero exit status.
- A diagnostic message shall be written to standard error whenever an error condition occurs.

When a utility encounters an error condition several actions are possible, depending on the severity of the error and the state of the utility. Included in the possible actions of various utilities are: deletion of temporary or intermediate work files; deletion of incomplete files; validity checking of the file system or directory.

Default Behavior: When this section is listed as “Default.”, it means that any changes to the environment, file systems, process state, and so on are unspecified.

APPLICATION USAGE

This section is informative.

The APPLICATION USAGE section gives advice to the application programmer or user about the way the utility should be used.

EXAMPLES

This section is informative.

The EXAMPLES section gives one or more examples of usage, where appropriate. In the event of conflict between an example and a normative part of the specification, the normative material is to be taken as correct.

In all examples, quoting has been used, showing how sample commands (utility names combined with arguments) could be passed correctly to a shell (see *sh*) or as a string to the *system()* function defined in the System Interfaces volume of POSIX.1-2008. Such quoting would not be used if the utility is invoked using one of the *exec* functions defined in the System Interfaces volume of POSIX.1-2008.

RATIONALE

This section is informative.

This section contains historical information concerning the contents of this volume of POSIX.1-2008 and why features were included or discarded by the standard developers.

FUTURE DIRECTIONS

This section is informative.

The FUTURE DIRECTIONS section should be used as a guide to current thinking; there is not necessarily a commitment to implement all of these future directions in their entirety.

SEE ALSO

This section is informative.

The SEE ALSO section lists related entries.

CHANGE HISTORY

This section is informative.

This section shows the derivation of the entry and any significant changes that have been made to it.

Certain of the standard utilities describe how they can invoke other utilities or applications, such as by passing a command string to the command interpreter. The external influences (STDIN, ENVIRONMENT VARIABLES, and so on) and external effects (STDOUT, CONSEQUENCES OF

73500 ERRORS, and so on) of such invoked utilities are not described in the section concerning the
 73501 standard utility that invokes them.

73502 1.5 Considerations for Utilities in Support of Files of Arbitrary Size

73503 The following utilities support files of any size up to the maximum that can be created by the
 73504 implementation. This support includes correct writing of file size-related values (such as file
 73505 sizes and offsets, line numbers, and block counts) and correct interpretation of command line
 73506 arguments that contain such values.

73507	<i>basename</i>	Return non-directory portion of pathname.
73508	<i>cat</i>	Concatenate and print files.
73509	<i>cd</i>	Change working directory.
73510	<i>chgrp</i>	Change file group ownership.
73511	<i>chmod</i>	Change file modes.
73512	<i>chown</i>	Change file ownership.
73513	<i>cksum</i>	Write file checksums and sizes.
73514	<i>cmp</i>	Compare two files.
73515	<i>cp</i>	Copy files.
73516	<i>dd</i>	Convert and copy a file.
73517	<i>df</i>	Report free disk space.
73518	<i>dirname</i>	Return directory portion of pathname.
73519	<i>du</i>	Estimate file space usage.
73520	<i>find</i>	Find files.
73521	<i>ln</i>	Link files.
73522	<i>ls</i>	List directory contents.
73523	<i>mkdir</i>	Make directories.
73524	<i>mv</i>	Move files.
73525	<i>pathchk</i>	Check pathnames.
73526	<i>pwd</i>	Return working directory name.
73527	<i>rm</i>	Remove directory entries.
73528	<i>rmdir</i>	Remove directories.
73529	<i>sh</i>	Shell, the standard command language interpreter.
73530	<i>sum</i>	Print checksum and block or byte count of a file.
73531	<i>test</i>	Evaluate expression.
73532	<i>touch</i>	Change file access and modification times.
73533	<i>ulimit</i>	Set or report file size limit.

- Exceptions to the requirement that utilities support files of any size up to the maximum are as follows:
1. Uses of files as command scripts, or for configuration or control, are exempt. For example, it is not required that *sh* be able to read an arbitrarily large **.profile**.
 2. Shell input and output redirection are exempt. For example, it is not required that the redirections *sum < file* or *echo foo > file* succeed for an arbitrarily large existing file.

1.6 Built-In Utilities

Any of the standard utilities may be implemented as regular built-in utilities within the command language interpreter. This is usually done to increase the performance of frequently used utilities or to achieve functionality that would be more difficult in a separate environment. The utilities named in [Table 1-5](#) are frequently provided in built-in form. All of the utilities named in the table have special properties in terms of command search order within the shell, as described in [Section 2.9.1.1](#) (on page 2339).

Table 1-5 Regular Built-In Utilities

<i>alias</i>	<i>false</i>	<i>jobs</i>	<i>read</i>	<i>wait</i>
<i>bg</i>	<i>fc</i>	<i>kill</i>	<i>true</i>	
<i>cd</i>	<i>fg</i>	<i>newgrp</i>	<i>umask</i>	
<i>command</i>	<i>getopts</i>	<i>pwd</i>	<i>unalias</i>	

However, all of the standard utilities, including the regular built-ins in the table, but not the special built-ins described in [Section 2.14](#) (on page 2356), shall be implemented in a manner so that they can be accessed via the *exec* family of functions as defined in the System Interfaces volume of POSIX.1-2008 and can be invoked directly by those standard utilities that require it (*env*, *find*, *nice*, *nohup*, *time*, *xargs*).

Shell Command Language

This chapter contains the definition of the Shell Command Language.

2.1 Shell Introduction

The shell is a command language interpreter. This chapter describes the syntax of that command language as it is used by the *sh* utility and the *system()* and *popen()* functions defined in the System Interfaces volume of POSIX.1-2008.

The shell operates according to the following general overview of operations. The specific details are included in the cited sections of this chapter.

1. The shell reads its input from a file (see *sh*), from the `-c` option or from the *system()* and *popen()* functions defined in the System Interfaces volume of POSIX.1-2008. If the first line of a file of shell commands starts with the characters "#!", the results are unspecified.

2. The shell breaks the input into tokens: words and operators; see [Section 2.3](#).

3. The shell parses the input into simple commands (see [Section 2.9.1](#)) and compound commands (see [Section 2.9.4](#)).

4. The shell performs various expansions (separately) on different parts of each command, resulting in a list of pathnames and fields to be treated as a command and arguments; see [Section 2.6](#).

5. The shell performs redirection (see [Section 2.7](#)) and removes redirection operators and their operands from the parameter list.

6. The shell executes a function (see [Section 2.9.5](#)), built-in (see [Section 2.14](#)), executable file, or script, giving the names of the arguments as positional parameters numbered 1 to *n*, and the name of the command (or in the case of a function within a script, the name of the script) as the positional parameter numbered 0 (see [Section 2.9.1.1](#)).

7. The shell optionally waits for the command to complete and collects the exit status (see [Section 2.8.2](#)).

2.2 Quoting

Quoting is used to remove the special meaning of certain characters or words to the shell. Quoting can be used to preserve the literal meaning of the special characters in the next paragraph, prevent reserved words from being recognized as such, and prevent parameter expansion and command substitution within here-document processing (see [Section 2.7.4](#)).

The application shall quote the following characters if they are to represent themselves:

| & ; < > () \$ ` \ " ' <space> <tab> <newline>

and the following may need to be quoted under certain circumstances. That is, these characters may be special depending on conditions described elsewhere in this volume of POSIX.1-2008:

* ? [# ~ = %

The various quoting mechanisms are the escape character, single-quotes, and double-quotes. The here-document represents another form of quoting; see [Section 2.7.4](#).

2.2.1 Escape Character (Backslash)

A <backslash> that is not quoted shall preserve the literal value of the following character, with the exception of a <newline>. If a <newline> follows the <backslash>, the shell shall interpret this as line continuation. The <backslash> and <newline> shall be removed before splitting the input into tokens. Since the escaped <newline> is removed entirely from the input and is not replaced by any white space, it cannot serve as a token separator.

2.2.2 Single-Quotes

Enclosing characters in single-quotes (' ') shall preserve the literal value of each character within the single-quotes. A single-quote cannot occur within single-quotes.

2.2.3 Double-Quotes

Enclosing characters in double-quotes (" ") shall preserve the literal value of all characters within the double-quotes, with the exception of the characters backquote, <dollar-sign>, and <backslash>, as follows:

\$ The <dollar-sign> shall retain its special meaning introducing parameter expansion (see [Section 2.6.2](#)), a form of command substitution (see [Section 2.6.3](#)), and arithmetic expansion (see [Section 2.6.4](#)).

The input characters within the quoted string that are also enclosed between "\$ (" and the matching ') ' shall not be affected by the double-quotes, but rather shall define that command whose output replaces the "\$ (. . .) " when the word is expanded. The tokenizing rules in [Section 2.3](#), not including the alias substitutions in [Section 2.3.1](#), shall be applied recursively to find the matching ') '.

Within the string of characters from an enclosed "\$ { " to the matching ' } ' , an even number of unescaped double-quotes or single-quotes, if any, shall occur. A preceding <backslash> character shall be used to escape a literal ' { ' or ' } ' . The rule in [Section 2.6.2](#) shall be used to determine the matching ' } ' .

73621 ``` The backquote shall retain its special meaning introducing the other form of command
 73622 substitution (see [Section 2.6.3](#)). The portion of the quoted string from the initial backquote
 73623 and the characters up to the next backquote that is not preceded by a `<backslash>`, having
 73624 escape characters removed, defines that command whose output replaces `"`...`"` when
 73625 the word is expanded. Either of the following cases produces undefined results:

- 73626 • A single-quoted or double-quoted string that begins, but does not end, within the
 73627 `"`...`"` sequence
- 73628 • A `"`...`"` sequence that begins, but does not end, within the same double-quoted
 73629 string

73630 `\` The `<backslash>` shall retain its special meaning as an escape character (see [Section 2.2.1](#))
 73631 only when followed by one of the following characters when considered special:

73632 `$    ` " \ <newline>`

73633 The application shall ensure that a double-quote is preceded by a `<backslash>` to be included
 73634 within double-quotes. The parameter `'@'` has special meaning inside double-quotes and is
 73635 described in [Section 2.5.2](#).

73636 2.3 Token Recognition

73637 The shell shall read its input in terms of lines from a file, from a terminal in the case of an
 73638 interactive shell, or from a string in the case of `sh -c` or `system()`. The input lines can be of
 73639 unlimited length. These lines shall be parsed using two major modes: ordinary token recognition
 73640 and processing of here-documents.

73641 When an `io_here` token has been recognized by the grammar (see [Section 2.10](#)), one or more of
 73642 the subsequent lines immediately following the next `NEWLINE` token form the body of one or
 73643 more here-documents and shall be parsed according to the rules of [Section 2.7.4](#).

73644 When it is not processing an `io_here`, the shell shall break its input into tokens by applying the
 73645 first applicable rule below to the next character in its input. The token shall be from the current
 73646 position in the input until a token is delimited according to one of the rules below; the characters
 73647 forming the token are exactly those in the input, including any quoting characters. If it is
 73648 indicated that a token is delimited, and no characters have been included in a token, processing
 73649 shall continue until an actual token is delimited.

- 73650 1. If the end of input is recognized, the current token shall be delimited. If there is no
 73651 current token, the end-of-input indicator shall be returned as the token.
- 73652 2. If the previous character was used as part of an operator and the current character is not
 73653 quoted and can be used with the current characters to form an operator, it shall be used as
 73654 part of that (operator) token.
- 73655 3. If the previous character was used as part of an operator and the current character cannot
 73656 be used with the current characters to form an operator, the operator containing the
 73657 previous character shall be delimited.
- 73658 4. If the current character is `<backslash>`, single-quote, or double-quote and it is not quoted,
 73659 it shall affect quoting for subsequent characters up to the end of the quoted text. The rules
 73660 for quoting are as described in [Section 2.2](#). During token recognition no substitutions
 73661 shall be actually performed, and the result token shall contain exactly the characters that
 73662 appear in the input (except for `<newline>` joining), unmodified, including any embedded
 73663 or enclosing quotes or substitution operators, between the `<quotation-mark>` and the end
 73664 of the quoted text. The token shall not be delimited by the end of the quoted field.

5. If the current character is an unquoted ' \$ ' or ' ` ', the shell shall identify the start of any candidates for parameter expansion ([Section 2.6.2](#)), command substitution ([Section 2.6.3](#)), or arithmetic expansion ([Section 2.6.4](#)) from their introductory unquoted character sequences: ' \$ ' or "\$ {", "\$ (" or ' ` ', and "\$ ((", respectively. The shell shall read sufficient input to determine the end of the unit to be expanded (as explained in the cited sections). While processing the characters, if instances of expansions or quoting are found nested within the substitution, the shell shall recursively process them in the manner specified for the construct that is found. The characters found from the beginning of the substitution to its end, allowing for any recursion necessary to recognize embedded constructs, shall be included unmodified in the result token, including any embedded or enclosing substitution operators or quotes. The token shall not be delimited by the end of the substitution.
 6. If the current character is not quoted and can be used as the first character of a new operator, the current token (if any) shall be delimited. The current character shall be used as the beginning of the next (operator) token.
 7. If the current character is an unquoted <newline>, the current token shall be delimited.
 8. If the current character is an unquoted <blank>, any token containing the previous character is delimited and the current character shall be discarded.
 9. If the previous character was part of a word, the current character shall be appended to that word.
 10. If the current character is a ' # ', it and all subsequent characters up to, but excluding, the next <newline> shall be discarded as a comment. The <newline> that ends the line is not considered part of the comment.
 11. The current character is used as the start of a new word.
- Once a token is delimited, it is categorized as required by the grammar in [Section 2.10](#).

2.3.1 Alias Substitution

After a token has been delimited, but before applying the grammatical rules in [Section 2.10](#), a resulting word that is identified to be the command name word of a simple command shall be examined to determine whether it is an unquoted, valid alias name. However, reserved words in correct grammatical context shall not be candidates for alias substitution. A valid alias name (see [XBD Section 3.10](#)) shall be one that has been defined by the *alias* utility and not subsequently undefined using *unalias*. Implementations also may provide predefined valid aliases that are in effect when the shell is invoked. To prevent infinite loops in recursive aliasing, if the shell is not currently processing an alias of the same name, the word shall be replaced by the value of the alias; otherwise, it shall not be replaced.

If the value of the alias replacing the word ends in a <blank>, the shell shall check the next command word for alias substitution; this process shall continue until a word is found that is not a valid alias or an alias value does not end in a <blank>.

When used as specified by this volume of POSIX.1-2008, alias definitions shall not be inherited by separate invocations of the shell or by the utility execution environments invoked by the shell; see [Section 2.12](#).

2.4 Reserved Words

Reserved words are words that have special meaning to the shell; see [Section 2.9](#). The following words shall be recognized as reserved words:

!	do	esac	in
{	done	fi	then
}	elif	for	until
case	else	if	while

This recognition shall only occur when none of the characters is quoted and when the word is used as:

- The first word of a command
- The first word following one of the reserved words other than **case**, **for**, or **in**
- The third word in a **case** command (only **in** is valid in this case)
- The third word in a **for** command (only **in** and **do** are valid in this case)

See the grammar in [Section 2.10](#).

The following words may be recognized as reserved words on some implementations (when none of the characters are quoted), causing unspecified results:

[[	]]	function	select

Words that are the concatenation of a name and a <colon> (' : ') are reserved; their use produces unspecified results.

2.5 Parameters and Variables

A parameter can be denoted by a name, a number, or one of the special characters listed in [Section 2.5.2](#). A variable is a parameter denoted by a name.

A parameter is set if it has an assigned value (null is a valid value). Once a variable is set, it can only be unset by using the *unset* special built-in command.

2.5.1 Positional Parameters

A positional parameter is a parameter denoted by the decimal value represented by one or more digits, other than the single digit 0. The digits denoting the positional parameters shall always be interpreted as a decimal value, even if there is a leading zero. When a positional parameter with more than one digit is specified, the application shall enclose the digits in braces (see [Section 2.6.2](#)). Positional parameters are initially assigned when the shell is invoked (see *sh*), temporarily replaced when a shell function is invoked (see [Section 2.9.5](#)), and can be reassigned with the *set* special built-in command.

2.5.2 Special Parameters

Listed below are the special parameters and the values to which they shall expand. Only the values of the special parameters are listed; see [Section 2.6](#) for a detailed summary of all the stages involved in expanding words.

- @ Expands to the positional parameters, starting from one. When the expansion occurs within double-quotes, and where field splitting (see [Section 2.6.5](#)) is performed, each positional parameter shall expand as a separate field, with the provision that the expansion of the first parameter shall still be joined with the beginning part of the original word (assuming that the expanded parameter was embedded within a word), and the expansion of the last parameter shall still be joined with the last part of the original word. If there are no positional parameters, the expansion of '@' shall generate zero fields, even when '@' is double-quoted.
- * Expands to the positional parameters, starting from one. When the expansion occurs within a double-quoted string (see [Section 2.2.3](#)), it shall expand to a single field with the value of each parameter separated by the first character of the *IFS* variable, or by a <space> if *IFS* is unset. If *IFS* is set to a null string, this is not equivalent to unsetting it; its first character does not exist, so the parameter values are concatenated.
- # Expands to the decimal number of positional parameters. The command name (parameter 0) shall not be counted in the number given by '#' because it is a special parameter, not a positional parameter.
- ? Expands to the decimal exit status of the most recent pipeline (see [Section 2.9.2](#)).
- (Hyphen.) Expands to the current option flags (the single-letter option names concatenated into a string) as specified on invocation, by the *set* special built-in command, or implicitly by the shell.
- \$ Expands to the decimal process ID of the invoked shell. In a subshell (see [Section 2.12](#)), '\$' shall expand to the same value as that of the current shell.
- ! Expands to the decimal process ID of the most recent background command (see [Section 2.9.3](#)) executed from the current shell. (For example, background commands executed from subshells do not affect the value of "\$!" in the current shell environment.) For a pipeline, the process ID is that of the last command in the pipeline.
- 0 (Zero.) Expands to the name of the shell or shell script. See *sh* for a detailed description of how this name is derived.

See the description of the *IFS* variable in [Section 2.5.3](#).

2.5.3 Shell Variables

Variables shall be initialized from the environment (as defined by XBD [Chapter 8](#) and the *exec* function in the System Interfaces volume of POSIX.1-2008) and can be given new values with variable assignment commands. If a variable is initialized from the environment, it shall be marked for export immediately; see the *export* special built-in. New variables can be defined and initialized with variable assignments, with the *read* or *getopts* utilities, with the *name* parameter in a *for* loop, with the `${name=word}` expansion, or with other mechanisms provided as implementation extensions.

73779		The following variables shall affect the execution of the shell:
73780	UP	ENV The processing of the <i>ENV</i> shell variable shall be supported if the system supports the User Portability Utilities option.
73781		
73782		This variable, when and only when an interactive shell is invoked, shall be subjected to parameter expansion (see Section 2.6.2) by the shell and the resulting value shall be used as a pathname of a file containing shell commands to execute in the current environment. The file need not be executable. If the expanded value of <i>ENV</i> is not an absolute pathname, the results are unspecified. <i>ENV</i> shall be ignored if the user's real and effective user IDs or real and effective group IDs are different.
73783		
73784		
73785		
73786		
73787		
73788		
73789		HOME The pathname of the user's home directory. The contents of <i>HOME</i> are used in tilde expansion (see Section 2.6.1).
73790		
73791		IFS A string treated as a list of characters that is used for field splitting and to split lines into fields with the <i>read</i> command.
73792		
73793		If <i>IFS</i> is not set, it shall behave as normal for an unset variable, except that field splitting by the shell and line splitting by the <i>read</i> command shall be performed as if the value of <i>IFS</i> is <space><tab><newline>; see Section 2.6.5 .
73794		
73795		
73796		Implementations may ignore the value of <i>IFS</i> in the environment, or the absence of <i>IFS</i> from the environment, at the time the shell is invoked, in which case the shell shall set <i>IFS</i> to <space><tab><newline> when it is invoked.
73797		
73798		
73799		LANG Provide a default value for the internationalization variables that are unset or null. (See XBD Section 8.2 for the precedence of internationalization variables used to determine the values of locale categories.)
73800		
73801		
73802		LC_ALL The value of this variable overrides the <i>LC_*</i> variables and <i>LANG</i> , as described in XBD Chapter 8 .
73803		
73804		LC_COLLATE Determine the behavior of range expressions, equivalence classes, and multi-character collating elements within pattern matching.
73805		
73806		LC_CTYPE Determine the interpretation of sequences of bytes of text data as characters (for example, single-byte as opposed to multi-byte characters), which characters are defined as letters (character class alpha) and <blank> characters (character class blank), and the behavior of character classes within pattern matching. Changing the value of <i>LC_CTYPE</i> after the shell has started shall not affect the lexical processing of shell commands in the current shell execution environment or its subshells. Invoking a shell script or performing <i>exec sh</i> subjects the new shell to the changes in <i>LC_CTYPE</i> .
73807		
73808		
73809		
73810		
73811		
73812		
73813		
73814		LC_MESSAGES Determine the language in which messages should be written.
73815		
73816		LINENO Set by the shell to a decimal number representing the current sequential line number (numbered starting with 1) within a script or function before it executes each command. If the user unsets or resets <i>LINENO</i> , the variable may lose its special meaning for the life of the shell. If the shell is not currently executing a script or function, the value of <i>LINENO</i> is unspecified. This volume of POSIX.1-2008 specifies the effects of the variable only for systems supporting the User Portability Utilities option.
73817		
73818		
73819		
73820		
73821		
73822	XSI	NLSPATH Determine the location of message catalogs for the processing of <i>LC_MESSAGES</i> .
73823		

73824	<i>PATH</i>	A string formatted as described in XBD Chapter 8 , used to effect command interpretation; see Section 2.9.1.1 .
73825		
73826	<i>PPID</i>	Set by the shell to the decimal value of its parent process ID during initialization of the shell. In a subshell (see Section 2.12), <i>PPID</i> shall be set to the same value as that of the parent of the current shell. For example, <i>echo \$PPID</i> and (<i>echo \$PPID</i>) would produce the same value. This volume of POSIX.1-2008 specifies the effects of the variable only for systems supporting the User Portability Utilities option.
73827		
73828		
73829		
73830		
73831		
73832	<i>PS1</i>	Each time an interactive shell is ready to read a command, the value of this variable shall be subjected to parameter expansion and written to standard error. The default value shall be "\$ ". For users who have specific additional implementation-defined privileges, the default may be another, implementation-defined value. The shell shall replace each instance of the character '!' in <i>PS1</i> with the history file number of the next command to be typed. Escaping the '!' with another '!' (that is, "!!") shall place the literal character '!' in the prompt. This volume of POSIX.1-2008 specifies the effects of the variable only for systems supporting the User Portability Utilities option.
73833		
73834		
73835		
73836		
73837		
73838		
73839		
73840		
73841		
73842	<i>PS2</i>	Each time the user enters a <newline> prior to completing a command line in an interactive shell, the value of this variable shall be subjected to parameter expansion and written to standard error. The default value is "> ". This volume of POSIX.1-2008 specifies the effects of the variable only for systems supporting the User Portability Utilities option.
73843		
73844		
73845		
73846		
73847	<i>PS4</i>	When an execution trace (<i>set -x</i>) is being performed in an interactive shell, before each line in the execution trace, the value of this variable shall be subjected to parameter expansion and written to standard error. The default value is "+ ". This volume of POSIX.1-2008 specifies the effects of the variable only for systems supporting the User Portability Utilities option.
73848		
73849		
73850		
73851		
73852	<i>PWD</i>	Set by the shell and by the <i>cd</i> utility. In the shell the value shall be initialized from the environment as follows. If a value for <i>PWD</i> is passed to the shell in the environment when it is executed, the value is an absolute pathname of the current working directory that is no longer than {PATH_MAX} bytes including the terminating null byte, and the value does not contain any components that are dot or dot-dot, then the shell shall set <i>PWD</i> to the value from the environment. Otherwise, if a value for <i>PWD</i> is passed to the shell in the environment when it is executed, the value is an absolute pathname of the current working directory, and the value does not contain any components that are dot or dot-dot, then it is unspecified whether the shell sets <i>PWD</i> to the value from the environment or sets <i>PWD</i> to the pathname that would be output by <i>pwd -P</i> . Otherwise, the <i>sh</i> utility sets <i>PWD</i> to the pathname that would be output by <i>pwd -P</i> . In cases where <i>PWD</i> is set to the value from the environment, the value can contain components that refer to files of type symbolic link. In cases where <i>PWD</i> is set to the pathname that would be output by <i>pwd -P</i> , if there is insufficient permission on the current working directory, or on any parent of that directory, to determine what that pathname would be, the value of <i>PWD</i> is unspecified. Assignments to this variable may be ignored. If an application sets or unsets the value of <i>PWD</i> , the behaviors of the <i>cd</i> and <i>pwd</i> utilities are unspecified.
73853		
73854		
73855		
73856		
73857		
73858		
73859		
73860		
73861		
73862		
73863		
73864		
73865		
73866		
73867		
73868		
73869		
73870		
73871		

2.6 Word Expansions

This section describes the various expansions that are performed on words. Not all expansions are performed on every word, as explained in the following sections.

Tilde expansions, parameter expansions, command substitutions, arithmetic expansions, and quote removals that occur within a single word expand to a single field. It is only field splitting or pathname expansion that can create multiple fields from a single word. The single exception to this rule is the expansion of the special parameter '@' within double-quotes, as described in [Section 2.5.2](#).

The order of word expansion shall be as follows:

1. Tilde expansion (see [Section 2.6.1](#)), parameter expansion (see [Section 2.6.2](#)), command substitution (see [Section 2.6.3](#)), and arithmetic expansion (see [Section 2.6.4](#)) shall be performed, beginning to end. See item 5 in [Section 2.3](#).
2. Field splitting (see [Section 2.6.5](#)) shall be performed on the portions of the fields generated by step 1, unless *IFS* is null.
3. Pathname expansion (see [Section 2.6.6](#)) shall be performed, unless *set -f* is in effect.
4. Quote removal (see [Section 2.6.7](#)) shall always be performed last.

The expansions described in this section shall occur in the same shell environment as that in which the command is executed.

If the complete expansion appropriate for a word results in an empty field, that empty field shall be deleted from the list of fields that form the completely expanded command, unless the original word contained single-quote or double-quote characters.

The '\$' character is used to introduce parameter expansion, command substitution, or arithmetic evaluation. If an unquoted '\$' is followed by a character that is not one of the following:

- A numeric character
- The name of one of the special parameters (see [Section 2.5.2](#))
- A valid first character of a variable name
- A <left-curly-bracket> ('{')
- A <left-parenthesis> '(')

the result is unspecified.

2.6.1 Tilde Expansion

A “tilde-prefix” consists of an unquoted <tilde> character at the beginning of a word, followed by all of the characters preceding the first unquoted <slash> in the word, or all the characters in the word if there is no <slash>. In an assignment (see [XBD Section 4.22](#)), multiple tilde-prefixes can be used: at the beginning of the word (that is, following the <equals-sign> of the assignment), following any unquoted <colon>, or both. A tilde-prefix in an assignment is terminated by the first unquoted <colon> or <slash>. If none of the characters in the tilde-prefix are quoted, the characters in the tilde-prefix following the <tilde> are treated as a possible login name from the user database. A portable login name cannot contain characters outside the set given in the description of the *LOGNAME* environment variable in [XBD Section 8.3](#). If the login name is null (that is, the tilde-prefix contains only the tilde), the tilde-prefix is replaced by the value of the variable *HOME*. If *HOME* is unset, the results are unspecified. Otherwise, the tilde-

prefix shall be replaced by a pathname of the initial working directory associated with the login name obtained using the `getpwnam()` function as defined in the System Interfaces volume of POSIX.1-2008. If the system does not recognize the login name, the results are undefined.

The pathname resulting from tilde expansion shall be treated as if quoted to prevent it being altered by field splitting and pathname expansion.

2.6.2 Parameter Expansion

The format for parameter expansion is as follows:

`${expression}`

where *expression* consists of all characters until the matching `'}'`. Any `'}'` escaped by a `<backslash>` or within a quoted string, and characters in embedded arithmetic expansions, command substitutions, and variable expansions, shall not be examined in determining the matching `'}'`.

The simplest form for parameter expansion is:

`${parameter}`

The value, if any, of *parameter* shall be substituted.

The parameter name or symbol can be enclosed in braces, which are optional except for positional parameters with more than one digit or when *parameter* is a name and is followed by a character that could be interpreted as part of the name. The matching closing brace shall be determined by counting brace levels, skipping over enclosed quoted strings, and command substitutions.

If the parameter is not enclosed in braces, and is a name, the expansion shall use the longest valid name (see XBD Section 3.231), whether or not the variable represented by that name exists. Otherwise, the parameter is a single-character symbol, and behavior is unspecified if that character is neither a digit nor one of the special parameters (see Section 2.5.2).

If a parameter expansion occurs inside double-quotes:

- Pathname expansion shall not be performed on the results of the expansion.
- Field splitting shall not be performed on the results of the expansion, with the exception of `'@'`; see Section 2.5.2.

In addition, a parameter expansion can be modified by using one of the following formats. In each case that a value of *word* is needed (based on the state of *parameter*, as described below), *word* shall be subjected to tilde expansion, parameter expansion, command substitution, and arithmetic expansion. If *word* is not needed, it shall not be expanded. The `'}'` character that delimits the following parameter expansion modifications shall be determined as described previously in this section and in Section 2.2.3. (For example, `${foo-bar}xyz` would result in the expansion of **foo** followed by the string **xyz** if **foo** is set, else the string `"barxyz"`).

`${parameter}:[word]` **Use Default Values.** If *parameter* is unset or null, the expansion of *word* (or an empty string if *word* is omitted) shall be substituted; otherwise, the value of *parameter* shall be substituted.

`${parameter}:= [word]` **Assign Default Values.** If *parameter* is unset or null, the expansion of *word* (or an empty string if *word* is omitted) shall be assigned to *parameter*. In all cases, the final value of *parameter* shall be substituted. Only variables, not positional parameters or special parameters, can be assigned in this way.

`${parameter}:[word]` **Indicate Error if Null or Unset.** If *parameter* is unset or null, the expansion of *word* (or a message indicating it is unset if *word* is omitted) shall be written to standard error and the shell exits with a non-zero exit status. Otherwise, the value of *parameter* shall be substituted. An interactive shell need not exit.

`${parameter}:[word]` **Use Alternative Value.** If *parameter* is unset or null, null shall be substituted; otherwise, the expansion of *word* (or an empty string if *word* is omitted) shall be substituted.

In the parameter expansions shown previously, use of the <colon> in the format shall result in a test for a parameter that is unset or null; omission of the <colon> shall result in a test for a parameter that is only unset. If *parameter* is '# ' and the colon is omitted, the application shall ensure that *word* is specified (this is necessary to avoid ambiguity with the string length expansion). The following table summarizes the effect of the <colon>:

	<i>parameter</i> Set and Not Null	<i>parameter</i> Set But Null	<i>parameter</i> Unset
<code>\${parameter:-word}</code>	substitute <i>parameter</i>	substitute <i>word</i>	substitute <i>word</i>
<code>\${parameter-word}</code>	substitute <i>parameter</i>	substitute null	substitute <i>word</i>
<code>\${parameter:=word}</code>	substitute <i>parameter</i>	assign <i>word</i>	assign <i>word</i>
<code>\${parameter=word}</code>	substitute <i>parameter</i>	substitute null	assign <i>word</i>
<code>\${parameter}?word}</code>	substitute <i>parameter</i>	error, exit	error, exit
<code>\${parameter?word}</code>	substitute <i>parameter</i>	substitute null	error, exit
<code>\${parameter:+word}</code>	substitute <i>word</i>	substitute null	substitute null
<code>\${parameter+word}</code>	substitute <i>word</i>	substitute <i>word</i>	substitute null

In all cases shown with “substitute”, the expression is replaced with the value shown. In all cases shown with “assign”, *parameter* is assigned that value, which also replaces the expression.

`${#parameter}` **String Length.** The length in characters of the value of *parameter* shall be substituted. If *parameter* is ' * ' or ' @ ' , the result of the expansion is unspecified. If *parameter* is unset and *set -u* is in effect, the expansion shall fail.

The following four varieties of parameter expansion provide for substring processing. In each case, pattern matching notation (see [Section 2.13](#)), rather than regular expression notation, shall be used to evaluate the patterns. If *parameter* is ' # ' , ' * ' , or ' @ ' , the result of the expansion is unspecified. If *parameter* is unset and *set -u* is in effect, the expansion shall fail. Enclosing the full parameter expansion string in double-quotes shall not cause the following four varieties of pattern characters to be quoted, whereas quoting characters within the braces shall have this effect. In each variety, if *word* is omitted, the empty pattern shall be used.

`${parameter}%[word]` **Remove Smallest Suffix Pattern.** The *word* shall be expanded to produce a pattern. The parameter expansion shall then result in *parameter*, with the smallest portion of the suffix matched by the *pattern* deleted. If present, *word* shall not begin with an unquoted ' % ' .

`${parameter}%%[word]` **Remove Largest Suffix Pattern.** The *word* shall be expanded to produce a pattern. The parameter expansion shall then result in *parameter*, with the largest portion of the suffix matched by the *pattern* deleted.

`${parameter}#[word]` **Remove Smallest Prefix Pattern.** The *word* shall be expanded to produce a pattern. The parameter expansion shall then result in *parameter*, with the smallest portion of the prefix matched by the *pattern* deleted. If present, *word* shall not begin with an unquoted ' # ' .

74004 `${parameter##[word]}` **Remove Largest Prefix Pattern.** The *word* shall be expanded to produce a
 74005 pattern. The parameter expansion shall then result in *parameter*, with the
 74006 largest portion of the prefix matched by the *pattern* deleted.

74007 Examples

74008 `${parameter}`

74009 In this example, the effects of omitting braces are demonstrated.

74010 `a=1`

74011 `set 2`

74012 `echo ${a}b-$ab-${1}0-${10}-${10}`

74013 **1b--20--20**

74014 `${parameter:-word}`

74015 In this example, *ls* is executed only if *x* is null or unset. (The `$(ls)` command substitution
 74016 notation is explained in [Section 2.6.3](#).)

74017 `${x:-$(ls)}`

74018 `${parameter:=word}`

74019 `unset X`

74020 `echo ${X:=abc}`

74021 **abc**

74022 `${parameter:?word}`

74023 `unset posix`

74024 `echo ${posix:?}`

74025 **sh: posix: parameter null or not set**

74026 `${parameter:+word}`

74027 `set a b c`

74028 `echo ${3:+posix}`

74029 **posix**

74030 `${#parameter}`

74031 `HOME=/usr/posix`

74032 `echo ${#HOME}`

74033 **10**

74034 `${parameter%word}`

74035 `x=file.c`

74036 `echo ${x%.c}.o`

74037 **file.o**

74038 `${parameter%%word}`

74039 `x=posix/src/std`

74040 `echo ${x%/*}`

74041 **posix**

74042 `${parameter#word}`

74043 `x=$HOME/src/cmd`

74044 `echo ${x#$HOME}`

74045 **/src/cmd**

74046 `${parameter###word}`

74047 `x=/one/two/three`

74048 `echo ${x###*/}`

three

The double-quoting of patterns is different depending on where the double-quotes are placed:

"\${x#*}" The <asterisk> is a pattern character.

\${x#"*"} The literal <asterisk> is quoted and not special.

2.6.3 Command Substitution

Command substitution allows the output of a command to be substituted in place of the command name itself. Command substitution shall occur when the command is enclosed as follows:

\$(*command*)

or (backquoted version):

`*command*`

The shell shall expand the command substitution by executing *command* in a subshell environment (see [Section 2.12](#)) and replacing the command substitution (the text of *command* plus the enclosing "\${()}" or backquotes) with the standard output of the command, removing sequences of one or more <newline> characters at the end of the substitution. Embedded <newline> characters before the end of the output shall not be removed; however, they may be treated as field delimiters and eliminated during field splitting, depending on the value of *IFS* and quoting that is in effect. If the output contains any null bytes, the behavior is unspecified.

Within the backquoted style of command substitution, <backslash> shall retain its literal meaning, except when followed by: '\$', '`', or <backslash>. The search for the matching backquote shall be satisfied by the first unquoted non-escaped backquote; during this search, if a non-escaped backquote is encountered within a shell comment, a here-document, an embedded command substitution of the \$(*command*) form, or a quoted string, undefined results occur. A single-quoted or double-quoted string that begins, but does not end, within the "`...`" sequence produces undefined results.

With the \$(*command*) form, all characters following the open parenthesis to the matching closing parenthesis constitute the *command*. Any valid shell script can be used for *command*, except a script consisting solely of redirections which produces unspecified results.

The results of command substitution shall not be processed for further tilde expansion, parameter expansion, command substitution, or arithmetic expansion. If a command substitution occurs inside double-quotes, field splitting and pathname expansion shall not be performed on the results of the substitution.

Command substitution can be nested. To specify nesting within the backquoted version, the application shall precede the inner backquotes with <backslash> characters; for example:

\`*command*\`

The syntax of the shell command language has an ambiguity for expansions beginning with "\${(", which can introduce an arithmetic expansion or a command substitution that starts with a subshell. Arithmetic expansion has precedence; that is, the shell shall first determine whether it can parse the expansion as an arithmetic expansion and shall only parse the expansion as a command substitution if it determines that it cannot parse the expansion as an arithmetic expansion. The shell need not evaluate nested expansions when performing this determination. If it encounters the end of input without already having determined that it cannot parse the expansion as an arithmetic expansion, the shell shall treat the expansion as an incomplete

arithmetic expansion and report a syntax error. A conforming application shall ensure that it separates the "\$ (" and ' (' into two tokens (that is, separate them with white space) in a command substitution that starts with a subshell. For example, a command substitution containing a single subshell could be written as:

```
$ ( (command) )
```

2.6.4 Arithmetic Expansion

Arithmetic expansion provides a mechanism for evaluating an arithmetic expression and substituting its value. The format for arithmetic expansion shall be as follows:

```
$ ((expression))
```

The expression shall be treated as if it were in double-quotes, except that a double-quote inside the expression is not treated specially. The shell shall expand all tokens in the expression for parameter expansion, command substitution, and quote removal.

Next, the shell shall treat this as an arithmetic expression and substitute the value of the expression. The arithmetic expression shall be processed according to the rules given in [Section 1.1.2.1](#), with the following exceptions:

- Only signed long integer arithmetic is required.
- Only the decimal-constant, octal-constant, and hexadecimal-constant constants specified in the ISO C standard, Section 6.4.4.1 are required to be recognized as constants.
- The `sizeof()` operator and the prefix and postfix "++" and "--" operators are not required.
- Selection, iteration, and jump statements are not supported.

All changes to variables in an arithmetic expression shall be in effect after the arithmetic expansion, as in the parameter expansion "\${x=value}".

If the shell variable *x* contains a value that forms a valid integer constant, optionally including a leading plus or minus sign, then the arithmetic expansions "\$ ((x))" and "\$ ((\$x))" shall return the same value.

As an extension, the shell may recognize arithmetic expressions beyond those listed. The shell may use a signed integer type with a rank larger than the rank of **signed long**. The shell may use a real-floating type instead of **signed long** as long as it does not affect the results in cases where there is no overflow. If the expression is invalid, or the contents of a shell variable used in the expression are not recognized by the shell, the expansion fails and the shell shall write a diagnostic message to standard error indicating the failure.

Examples

A simple example using arithmetic expansion:

```
# repeat a command 100 times
x=100
while [ $x -gt 0 ]
do
    command
    x=$((x-1))
done
```

2.6.5 Field Splitting

After parameter expansion (Section 2.6.2), command substitution (Section 2.6.3), and arithmetic expansion (Section 2.6.4), the shell shall scan the results of expansions and substitutions that did not occur in double-quotes for field splitting and multiple fields can result.

The shell shall treat each character of the *IFS* as a delimiter and use the delimiters as field terminators to split the results of parameter expansion and command substitution into fields.

1. If the value of *IFS* is a <space>, <tab>, and <newline>, or if it is unset, any sequence of <space>, <tab>, or <newline> characters at the beginning or end of the input shall be ignored and any sequence of those characters within the input shall delimit a field. For example, the input:

```
<newline><space><tab>foo<tab><tab>bar<space>
```

yields two fields, **foo** and **bar**.

2. If the value of *IFS* is null, no field splitting shall be performed.
3. Otherwise, the following rules shall be applied in sequence. The term “*IFS* white space” is used to mean any sequence (zero or more instances) of white-space characters that are in the *IFS* value (for example, if *IFS* contains <space>/<comma>/<tab>, any sequence of <space> and <tab> characters is considered *IFS* white space).
 - a. *IFS* white space shall be ignored at the beginning and end of the input.
 - b. Each occurrence in the input of an *IFS* character that is not *IFS* white space, along with any adjacent *IFS* white space, shall delimit a field, as described previously.
 - c. Non-zero-length *IFS* white space shall delimit a field.

2.6.6 Pathname Expansion

After field splitting, if *set -f* is not in effect, each field in the resulting command line shall be expanded using the algorithm described in Section 2.13, qualified by the rules in Section 2.13.3.

2.6.7 Quote Removal

The quote characters (<backslash>, single-quote, and double-quote) that were present in the original word shall be removed unless they have themselves been quoted.

2.7 Redirection

Redirection is used to open and close files for the current shell execution environment (see [Section 2.12](#)) or for any command. Redirection operators can be used with numbers representing file descriptors (see [XBD Section 3.166](#)) as described below.

The overall format used for redirection is:

```
[n]redir-op word
```

The number *n* is an optional decimal number designating the file descriptor number; the application shall ensure it is delimited from any preceding text and immediately precede the redirection operator *redir-op*. If *n* is quoted, the number shall not be recognized as part of the redirection expression. For example:

```
echo \2>a
```

writes the character 2 into file **a**. If any part of *redir-op* is quoted, no redirection expression is recognized. For example:

```
echo 2\>a
```

writes the characters 2>a to standard output. The optional number, redirection operator, and *word* shall not appear in the arguments provided to the command to be executed (if any).

Open files are represented by decimal numbers starting with zero. The largest possible value is implementation-defined; however, all implementations shall support at least 0 to 9, inclusive, for use by the application. These numbers are called “file descriptors”. The values 0, 1, and 2 have special meaning and conventional uses and are implied by certain redirection operations; they are referred to as *standard input*, *standard output*, and *standard error*, respectively. Programs usually take their input from standard input, and write output on standard output. Error messages are usually written on standard error. The redirection operators can be preceded by one or more digits (with no intervening <blank> characters allowed) to designate the file descriptor number.

If the redirection operator is "<<" or "<<=", the word that follows the redirection operator shall be subjected to quote removal; it is unspecified whether any of the other expansions occur. For the other redirection operators, the word that follows the redirection operator shall be subjected to tilde expansion, parameter expansion, command substitution, arithmetic expansion, and quote removal. Pathname expansion shall not be performed on the word by a non-interactive shell; an interactive shell may perform it, but shall do so only when the expansion would result in one word.

If more than one redirection operator is specified with a command, the order of evaluation is from beginning to end.

A failure to open or create a file shall cause a redirection to fail.

2.7.1 Redirecting Input

Input redirection shall cause the file whose name results from the expansion of *word* to be opened for reading on the designated file descriptor, or standard input if the file descriptor is not specified.

The general format for redirecting input is:

```
[n]<word
```

where the optional *n* represents the file descriptor number. If the number is omitted, the

74201 redirection shall refer to standard input (file descriptor 0).

74202 2.7.2 Redirecting Output

74203 The two general formats for redirecting output are:

74204 `[n]>word`

74205 `[n]>|word`

74206 where the optional *n* represents the file descriptor number. If the number is omitted, the
74207 redirection shall refer to standard output (file descriptor 1).

74208 Output redirection using the '`>`' format shall fail if the *noclobber* option is set (see the
74209 description of *set -C*) and the file named by the expansion of *word* exists and is a regular file.
74210 Otherwise, redirection using the '`>`' or '`>|`' formats shall cause the file whose name results
74211 from the expansion of *word* to be created and opened for output on the designated file
74212 descriptor, or standard output if none is specified. If the file does not exist, it shall be created;
74213 otherwise, it shall be truncated to be an empty file after being opened.

74214 2.7.3 Appending Redirected Output

74215 Appended output redirection shall cause the file whose name results from the expansion of
74216 *word* to be opened for output on the designated file descriptor. The file is opened as if the *open()*
74217 function as defined in the System Interfaces volume of POSIX.1-2008 was called with the
74218 *O_APPEND* flag. If the file does not exist, it shall be created.

74219 The general format for appending redirected output is as follows:

74220 `[n]>>word`

74221 where the optional *n* represents the file descriptor number. If the number is omitted, the
74222 redirection refers to standard output (file descriptor 1).

74223 2.7.4 Here-Document

74224 The redirection operators "`<<`" and "`<<-`" both allow redirection of lines contained in a shell
74225 input file, known as a "here-document", to the input of a command.

74226 The here-document shall be treated as a single word that begins after the next `<newline>` and
74227 continues until there is a line containing only the delimiter and a `<newline>`, with no `<blank>`
74228 characters in between. Then the next here-document starts, if there is one. The format is as
74229 follows:

74230 `[n]<<word`

74231 `here-document`

74232 `delimiter`

74233 where the optional *n* represents the file descriptor number. If the number is omitted, the here-
74234 document refers to standard input (file descriptor 0).

74235 If any character in *word* is quoted, the delimiter shall be formed by performing quote removal on
74236 *word*, and the here-document lines shall not be expanded. Otherwise, the delimiter shall be the
74237 *word* itself.

74238 If no characters in *word* are quoted, all lines of the here-document shall be expanded for
74239 parameter expansion, command substitution, and arithmetic expansion. In this case, the

<backslash> in the input behaves as the <backslash> inside double-quotes (see [Section 2.2.3](#)). However, the double-quote character ('"') shall not be treated specially within a here-document, except when the double-quote appears within "\$()", "\$`", or "\${ }".

If the redirection symbol is "<<-", all leading <tab> characters shall be stripped from input lines and the line containing the trailing delimiter. If more than one "<<" or "<<-" operator is specified on a line, the here-document associated with the first operator shall be supplied first by the application and shall be read first by the shell.

When a here-document is read from a terminal device and the shell is interactive, it shall write the contents of the variable *PS2*, processed as described in [Section 2.5.3](#), to standard error before reading each line of input until the delimiter has been recognized.

Examples

An example of a here-document follows:

```
cat <<eof1; cat <<eof2
Hi,
eof1
Helene.
eof2
```

2.7.5 Duplicating an Input File Descriptor

The redirection operator:

```
[n]<&word
```

shall duplicate one input file descriptor from another, or shall close one. If *word* evaluates to one or more digits, the file descriptor denoted by *n*, or standard input if *n* is not specified, shall be made to be a copy of the file descriptor denoted by *word*; if the digits in *word* do not represent a file descriptor already open for input, a redirection error shall result; see [Section 2.8.1](#). If *word* evaluates to '-', file descriptor *n*, or standard input if *n* is not specified, shall be closed. Attempts to close a file descriptor that is not open shall not constitute an error. If *word* evaluates to something else, the behavior is unspecified.

2.7.6 Duplicating an Output File Descriptor

The redirection operator:

```
[n]>&word
```

shall duplicate one output file descriptor from another, or shall close one. If *word* evaluates to one or more digits, the file descriptor denoted by *n*, or standard output if *n* is not specified, shall be made to be a copy of the file descriptor denoted by *word*; if the digits in *word* do not represent a file descriptor already open for output, a redirection error shall result; see [Section 2.8.1](#). If *word* evaluates to '-', file descriptor *n*, or standard output if *n* is not specified, is closed. Attempts to close a file descriptor that is not open shall not constitute an error. If *word* evaluates to something else, the behavior is unspecified.

2.7.7 Open File Descriptors for Reading and Writing

The redirection operator:

`[n] <>word`

shall cause the file whose name is the expansion of *word* to be opened for both reading and writing on the file descriptor denoted by *n*, or standard input if *n* is not specified. If the file does not exist, it shall be created.

2.8 Exit Status and Errors

2.8.1 Consequences of Shell Errors

For a non-interactive shell, an error condition encountered by a special built-in (see [Section 2.14](#)) or other type of utility shall cause the shell to write a diagnostic message to standard error and exit as shown in the following table:

Error	Special Built-In	Other Utilities
Shell language syntax error	Shall exit	Shall exit
Utility syntax error (option or operand error)	Shall exit	Shall not exit
Redirection error	Shall exit	Shall not exit
Variable assignment error	Shall exit	Shall not exit
Expansion error	Shall exit	Shall exit
Command not found	N/A	May exit
Dot script not found	Shall exit	N/A

An expansion error is one that occurs when the shell expansions defined in [Section 2.6](#) are carried out (for example, "`${x!y}`", because '`!`' is not a valid operator); an implementation may treat these as syntax errors if it is able to detect them during tokenization, rather than during expansion.

If any of the errors shown as “shall exit” or “(may) exit” occur in a subshell, the subshell shall (respectively may) exit with a non-zero status, but the script containing the subshell shall not exit because of the error.

In all of the cases shown in the table, an interactive shell shall write a diagnostic message to standard error without exiting.

2.8.2 Exit Status for Commands

Each command has an exit status that can influence the behavior of other shell commands. The exit status of commands that are not utilities is documented in this section. The exit status of the standard utilities is documented in their respective sections.

If a command is not found, the exit status shall be 127. If the command name is found, but it is not an executable utility, the exit status shall be 126. Applications that invoke utilities without using the shell should use these exit status values to report similar errors.

If a command fails during word expansion or redirection, its exit status shall be greater than zero.

Internally, for purposes of deciding whether a command exits with a non-zero exit status, the shell shall recognize the entire status value retrieved for the command by the equivalent of the `wait()` function WEXITSTATUS macro (as defined in the System Interfaces volume of POSIX.1-2008). When reporting the exit status with the special parameter `'?'`, the shell shall report the full eight bits of exit status available. The exit status of a command that terminated because it received a signal shall be reported as greater than 128.

2.9 Shell Commands

This section describes the basic structure of shell commands. The following command descriptions each describe a format of the command that is only used to aid the reader in recognizing the command type, and does not formally represent the syntax. Each description discusses the semantics of the command; for a formal definition of the command language, consult [Section 2.10](#).

A *command* is one of the following:

- Simple command (see [Section 2.9.1](#))
- Pipeline (see [Section 2.9.2](#))
- List compound-list (see [Section 2.9.3](#))
- Compound command (see [Section 2.9.4](#))
- Function definition (see [Section 2.9.5](#))

Unless otherwise stated, the exit status of a command shall be that of the last simple command executed by the command. There shall be no limit on the size of any shell command other than that imposed by the underlying system (memory constraints, {ARG_MAX}, and so on).

2.9.1 Simple Commands

A “simple command” is a sequence of optional variable assignments and redirections, in any sequence, optionally followed by words and redirections, terminated by a control operator.

When a given simple command is required to be executed (that is, when any conditional construct such as an AND-OR list or a **case** statement has not bypassed the simple command), the following expansions, assignments, and redirections shall all be performed from the beginning of the command text to the end:

1. The words that are recognized as variable assignments or redirections according to [Section 2.10.2](#) are saved for processing in steps 3 and 4.
2. The words that are not variable assignments or redirections shall be expanded. If any fields remain following their expansion, the first field shall be considered the command name and remaining fields are the arguments for the command.
3. Redirections shall be performed as described in [Section 2.7](#).
4. Each variable assignment shall be expanded for tilde expansion, parameter expansion, command substitution, arithmetic expansion, and quote removal prior to assigning the value.

In the preceding list, the order of steps 3 and 4 may be reversed if no command name results from step 2 or if the command name matches the name of a special built-in utility; see [Section 2.14](#).

If no command name results, or if the command name is a special built-in or function, variable assignments shall affect the current execution environment. Otherwise, the variable assignments shall be exported for the execution environment of the command and shall not affect the current execution environment except as a side-effect of the expansions performed in step 4. In this case it is unspecified:

- Whether or not the assignments are visible for subsequent expansions in step 4
- Whether variable assignments made as side-effects of these expansions are visible for subsequent expansions in step 4, or in the current shell execution environment, or both

If any of the variable assignments attempt to assign a value to a variable for which the *readonly* attribute is set in the current shell environment (regardless of whether the assignment is made in that environment), a variable assignment error shall occur. See [Section 2.8.1](#) for the consequences of these errors.

If there is no command name, any redirections shall be performed in a subshell environment; it is unspecified whether this subshell environment is the same one as that used for a command substitution within the command. (To affect the current execution environment, see the *exec* special built-in.) If any of the redirections performed in the current shell execution environment fail, the command shall immediately fail with an exit status greater than zero, and the shell shall write an error message indicating the failure. See [Section 2.8.1](#) for the consequences of these failures on interactive and non-interactive shells.

If there is a command name, execution shall continue as described in [Section 2.9.1.1](#). If there is no command name, but the command contained a command substitution, the command shall complete with the exit status of the last command substitution performed. Otherwise, the command shall complete with a zero exit status.

2.9.1.1 Command Search and Execution

If a simple command results in a command name and an optional list of arguments, the following actions shall be performed:

1. If the command name does not contain any <slash> characters, the first successful step in the following sequence shall occur:
 - a. If the command name matches the name of a special built-in utility, that special built-in utility shall be invoked.
 - b. If the command name matches the name of a function known to this shell, the function shall be invoked as described in [Section 2.9.5](#). If the implementation has provided a standard utility in the form of a function, it shall not be recognized at this point. It shall be invoked in conjunction with the path search in step 1d.
 - c. If the command name matches the name of a utility listed in the following table, that utility shall be invoked.

<i>alias</i>	<i>false</i>	<i>jobs</i>	<i>read</i>	<i>wait</i>
<i>bg</i>	<i>fc</i>	<i>kill</i>	<i>true</i>	
<i>cd</i>	<i>fg</i>	<i>newgrp</i>	<i>umask</i>	
<i>command</i>	<i>getopts</i>	<i>pwd</i>	<i>unalias</i>	

- d. Otherwise, the command shall be searched for using the *PATH* environment variable as described in XBD [Chapter 8](#):

- 74396 i. If the search is successful:
- 74397 a. If the system has implemented the utility as a regular built-in or as a
- 74398 shell function, it shall be invoked at this point in the path search.
- 74399 b. Otherwise, the shell executes the utility in a separate utility
- 74400 environment (see [Section 2.12](#)) with actions equivalent to calling the
- 74401 *execl()* function as defined in the System Interfaces volume of
- 74402 POSIX.1-2008 with the *path* argument set to the pathname resulting
- 74403 from the search, *arg0* set to the command name, and the remaining
- 74404 *execl()* arguments set to the command arguments (if any) and the
- 74405 null terminator.
- 74406 If the *execl()* function fails due to an error equivalent to the
- 74407 [ENOEXEC] error defined in the System Interfaces volume of
- 74408 POSIX.1-2008, the shell shall execute a command equivalent to
- 74409 having a shell invoked with the pathname resulting from the search
- 74410 as its first operand, with any remaining arguments passed to the new
- 74411 shell, except that the value of "\$0" in the new shell may be set to the
- 74412 command name. If the executable file is not a text file, the shell may
- 74413 bypass this command execution. In this case, it shall write an error
- 74414 message, and shall return an exit status of 126.
- 74415 It is unspecified whether environment variables that were passed to
- 74416 the shell when it was invoked, but were not used to initialize shell
- 74417 variables (see [Section 2.5.3](#)) because they had invalid names, are
- 74418 included in the environment passed to *execl()* and (if *execl()* fails as
- 74419 described above) to the new shell.
- 74420 Once a utility has been searched for and found (either as a result of this
- 74421 specific search or as part of an unspecified shell start-up activity), an
- 74422 implementation may remember its location and need not search for the
- 74423 utility again unless the *PATH* variable has been the subject of an assignment.
- 74424 If the remembered location fails for a subsequent invocation, the shell shall
- 74425 repeat the search to find the new location for the utility, if any.
- 74426 ii. If the search is unsuccessful, the command shall fail with an exit status of
- 74427 127 and the shell shall write an error message.
- 74428 2. If the command name contains at least one <slash>, the shell shall execute the utility in a
- 74429 separate utility environment with actions equivalent to calling the *execl()* function
- 74430 defined in the System Interfaces volume of POSIX.1-2008 with the *path* and *arg0*
- 74431 arguments set to the command name, and the remaining *execl()* arguments set to the
- 74432 command arguments (if any) and the null terminator.
- 74433 If the *execl()* function fails due to an error equivalent to the [ENOEXEC] error, the shell
- 74434 shall execute a command equivalent to having a shell invoked with the command name
- 74435 as its first operand, with any remaining arguments passed to the new shell. If the
- 74436 executable file is not a text file, the shell may bypass this command execution. In this case,
- 74437 it shall write an error message and shall return an exit status of 126.
- 74438 It is unspecified whether environment variables that were passed to the shell when it was
- 74439 invoked, but were not used to initialize shell variables (see [Section 2.5.3](#)) because they had
- 74440 invalid names, are included in the environment passed to *execl()* and (if *execl()* fails as
- 74441 described above) to the new shell.
- 74442 If the utility would be executed with file descriptor 0, 1, or 2 closed, implementations may
- 74443 execute the utility with the file descriptor open to an unspecified file. If a standard utility or a

conforming application is executed with file descriptor 0 not open for reading or with file descriptor 1 or 2 not open for writing, the environment in which the utility or application is executed shall be deemed non-conforming, and consequently the utility or application might not behave as described in this standard.

2.9.2 Pipelines

A *pipeline* is a sequence of one or more commands separated by the control operator '|'. For each command but the last, the shell shall connect the standard output of the command to the standard input of the next command as if by creating a pipe and passing the write end of the pipe as the standard output of the command and the read end of the pipe as the standard input of the next command.

The format for a pipeline is:

```
[!] command1 [ | command2 ...]
```

The standard output of *command1* shall be connected to the standard input of *command2*. The standard input, standard output, or both of a command shall be considered to be assigned by the pipeline before any redirection specified by redirection operators that are part of the command (see [Section 2.7](#)).

If the pipeline is not in the background (see [Section 2.9.3.1](#)), the shell shall wait for the last command specified in the pipeline to complete, and may also wait for all commands to complete.

Exit Status

If the pipeline does not begin with the ! reserved word, the exit status shall be the exit status of the last command specified in the pipeline. Otherwise, the exit status shall be the logical NOT of the exit status of the last command. That is, if the last command returns zero, the exit status shall be 1; if the last command returns greater than zero, the exit status shall be zero.

2.9.3 Lists

An *AND-OR list* is a sequence of one or more pipelines separated by the operators "&&" and "||".

A *list* is a sequence of one or more AND-OR lists separated by the operators ';' and '& '.

The operators "&&" and "||" shall have equal precedence and shall be evaluated with left associativity. For example, both of the following commands write solely **bar** to standard output:

```
false && echo foo || echo bar
true || echo foo && echo bar
```

A ';' separator or a '& ' or <newline> terminator shall cause the preceding AND-OR list to be executed sequentially; an '&' separator or terminator shall cause asynchronous execution of the preceding AND-OR list.

The term "compound-list" is derived from the grammar in [Section 2.10](#); it is equivalent to a sequence of *lists*, separated by <newline> characters, that can be preceded or followed by an arbitrary number of <newline> characters.

Examples

The following is an example that illustrates <newline> characters in compound-lists:

```
while
    # a couple of <newline>s
    # a list
    date && who || ls; cat file
    # a couple of <newline>s
    # another list
    wc file > output & true
do
    # 2 lists
    ls
    cat file
done
```

2.9.3.1 Asynchronous Lists

If a command is terminated by the control operator <ampersand> ('&'), the shell shall execute the command asynchronously in a subshell. This means that the shell shall not wait for the command to finish before executing the next command.

The format for running a command in the background is:

```
command1 & [command2 & ... ]
```

The standard input for an asynchronous list, before any explicit redirections are performed, shall be considered to be assigned to a file that has the same properties as **/dev/null**. If it is an interactive shell, this need not happen. In all cases, explicit redirection of standard input shall override this activity.

When an element of an asynchronous list (the portion of the list ended by an <ampersand>, such as *command1*, above) is started by the shell, the process ID of the last command in the asynchronous list element shall become known in the current shell execution environment; see [Section 2.12](#). This process ID shall remain known until:

1. The command terminates and the application waits for the process ID.
2. Another asynchronous list is invoked before "\$!" (corresponding to the previous asynchronous list) is expanded in the current execution environment.

The implementation need not retain more than the {CHILD_MAX} most recent entries in its list of known process IDs in the current shell execution environment.

Exit Status

The exit status of an asynchronous list shall be zero.

2.9.3.2 Sequential Lists

Commands that are separated by a <semicolon> (' ; ') shall be executed sequentially.

The format for executing commands sequentially shall be:

command1 [; *command2*] . . .

Each command shall be expanded and executed in the order specified.

Exit Status

The exit status of a sequential list shall be the exit status of the last command in the list.

2.9.3.3 AND Lists

The control operator "&&" denotes an AND list. The format shall be:

command1 [&& *command2*] . . .

First *command1* shall be executed. If its exit status is zero, *command2* shall be executed, and so on, until a command has a non-zero exit status or there are no more commands left to execute. The commands are expanded only if they are executed.

Exit Status

The exit status of an AND list shall be the exit status of the last command that is executed in the list.

2.9.3.4 OR Lists

The control operator "||" denotes an OR List. The format shall be:

command1 [|| *command2*] . . .

First, *command1* shall be executed. If its exit status is non-zero, *command2* shall be executed, and so on, until a command has a zero exit status or there are no more commands left to execute.

Exit Status

The exit status of an OR list shall be the exit status of the last command that is executed in the list.

2.9.4 Compound Commands

The shell has several programming constructs that are “compound commands”, which provide control flow for commands. Each of these compound commands has a reserved word or control operator at the beginning, and a corresponding terminator reserved word or operator at the end. In addition, each can be followed by redirections on the same line as the terminator. Each redirection shall apply to all the commands within the compound command that do not explicitly override that redirection.

74548 2.9.4.1 *Grouping Commands*

74549 The format for grouping commands is as follows:

74550 (*compound-list*) Execute *compound-list* in a subshell environment; see [Section 2.12](#).
 74551 Variable assignments and built-in commands that affect the environment
 74552 shall not remain in effect after the list finishes.

74553 If a character sequence beginning with "`(`" would be parsed by the shell +
 74554 as an arithmetic expansion if preceded by a '`$`', shells which implement +
 74555 an extension whereby "`(expression)`" is evaluated as an arithmetic +
 74556 expression may treat the "`(`" as introducing as an arithmetic evaluation +
 74557 instead of a grouping command. A conforming application shall ensure +
 74558 that it separates the two leading '`(`' characters with white space to +
 74559 prevent the shell from performing an arithmetic evaluation.

74560 { *compound-list*; } Execute *compound-list* in the current process environment. The semicolon
 74561 shown here is an example of a control operator delimiting the `}` reserved
 74562 word. Other delimiters are possible, as shown in [Section 2.10](#); a
 74563 `<newline>` is frequently used.

74564 **Exit Status**74565 The exit status of a grouping command shall be the exit status of *compound-list*.74566 2.9.4.2 *The for Loop*

74567 The **for** loop shall execute a sequence of commands for each member in a list of *items*. The **for**
 74568 loop requires that the reserved words **do** and **done** be used to delimit the sequence of
 74569 commands.

74570 The format for the **for** loop is as follows:

```
74571 for name [ in [word ... ] ]
74572 do
74573     compound-list
74574 done
```

74575 First, the list of words following **in** shall be expanded to generate a list of items. Then, the
 74576 variable *name* shall be set to each item, in turn, and the *compound-list* executed each time. If no
 74577 items result from the expansion, the *compound-list* shall not be executed. Omitting:

74578 `in word...`

74579 shall be equivalent to:

74580 `in "$@"`74581 **Exit Status**

74582 The exit status of a **for** command shall be the exit status of the last command that executes. If
 74583 there are no items, the exit status shall be zero.

74584 2.9.4.3 Case Conditional Construct

74585 The conditional construct **case** shall execute the *compound-list* corresponding to the first one of
 74586 several *patterns* (see [Section 2.13](#)) that is matched by the string resulting from the tilde
 74587 expansion, parameter expansion, command substitution, arithmetic expansion, and quote
 74588 removal of the given word. The reserved word **in** shall denote the beginning of the patterns to
 74589 be matched. Multiple patterns with the same *compound-list* shall be delimited by the '|'
 74590 symbol. The control operator ')' terminates a list of patterns corresponding to a given action.
 74591 The *compound-list* for each list of patterns, with the possible exception of the last, shall be
 74592 terminated with ';;'. The **case** construct terminates with the reserved word **esac** (**case**
 74593 reversed).

74594 The format for the **case** construct is as follows:

```
74595 case word in
74596     [(pattern1) compound-list;;
74597     [(pattern[ | pattern] ... ) compound-list;;] ...
74598     [(pattern[ | pattern] ... ) compound-list]
74599 esac
```

74600 The ";;" is optional for the last *compound-list*.

74601 In order from the beginning to the end of the **case** statement, each *pattern* that labels a *compound-*
 74602 *list* shall be subjected to tilde expansion, parameter expansion, command substitution, and
 74603 arithmetic expansion, and the result of these expansions shall be compared against the
 74604 expansion of *word*, according to the rules described in [Section 2.13](#) (which also describes the
 74605 effect of quoting parts of the pattern). After the first match, no more patterns shall be expanded,
 74606 and the *compound-list* shall be executed. The order of expansion and comparison of multiple
 74607 *patterns* that label a *compound-list* statement is unspecified.

74608 **Exit Status**

74609 The exit status of **case** shall be zero if no patterns are matched. Otherwise, the exit status shall be
 74610 the exit status of the last command executed in the *compound-list*.

74611 2.9.4.4 The if Conditional Construct

74612 The **if** command shall execute a *compound-list* and use its exit status to determine whether to
 74613 execute another *compound-list*.

74614 The format for the **if** construct is as follows:

```
74615 if compound-list
74616 then
74617     compound-list
74618 [elif compound-list
74619 then
74620     compound-list] ...
74621 [else
74622     compound-list]
74623 fi
```

74624 The **if** *compound-list* shall be executed; if its exit status is zero, the **then** *compound-list* shall be
 74625 executed and the command shall complete. Otherwise, each **elif** *compound-list* shall be executed,
 74626 in turn, and if its exit status is zero, the **then** *compound-list* shall be executed and the command
 74627 shall complete. Otherwise, the **else** *compound-list* shall be executed.

Exit Status

The exit status of the **if** command shall be the exit status of the **then** or **else** *compound-list* that was executed, or zero, if none was executed.

2.9.4.5 The while Loop

The **while** loop shall continuously execute one *compound-list* as long as another *compound-list* has a zero exit status.

The format of the **while** loop is as follows:

```
while compound-list-1
do
    compound-list-2
done
```

The *compound-list-1* shall be executed, and if it has a non-zero exit status, the **while** command shall complete. Otherwise, the *compound-list-2* shall be executed, and the process shall repeat.

Exit Status

The exit status of the **while** loop shall be the exit status of the last *compound-list-2* executed, or zero if none was executed.

2.9.4.6 The until Loop

The **until** loop shall continuously execute one *compound-list* as long as another *compound-list* has a non-zero exit status.

The format of the **until** loop is as follows:

```
until compound-list-1
do
    compound-list-2
done
```

The *compound-list-1* shall be executed, and if it has a zero exit status, the **until** command completes. Otherwise, the *compound-list-2* shall be executed, and the process repeats.

Exit Status

The exit status of the **until** loop shall be the exit status of the last *compound-list-2* executed, or zero if none was executed.

2.9.5 Function Definition Command

A function is a user-defined name that is used as a simple command to call a compound command with new positional parameters. A function is defined with a “function definition command”.

The format of a function definition command is as follows:

```
fname() compound-command[io-redirect ...]
```

The function is named *fname*; the application shall ensure that it is a name (see XBD [Section 3.231](#)) and that it is not the name of a special built-in utility. An implementation may allow other characters in a function name as an extension. The implementation shall maintain separate name

spaces for functions and variables.

The argument *compound-command* represents a compound command, as described in [Section 2.9.4](#).

When the function is declared, none of the expansions in [Section 2.6](#) shall be performed on the text in *compound-command* or *io-redirect*; all expansions shall be performed as normal each time the function is called. Similarly, the optional *io-redirect* redirections and any variable assignments within *compound-command* shall be performed during the execution of the function itself, not the function definition. See [Section 2.8.1](#) for the consequences of failures of these operations on interactive and non-interactive shells.

When a function is executed, it shall have the syntax-error and variable-assignment properties described for special built-in utilities in the enumerated list at the beginning of [Section 2.14](#).

The *compound-command* shall be executed whenever the function name is specified as the name of a simple command (see [Section 2.9.1.1](#)). The operands to the command temporarily shall become the positional parameters during the execution of the *compound-command*; the special parameter ' #' also shall be changed to reflect the number of operands. The special parameter 0 shall be unchanged. When the function completes, the values of the positional parameters and the special parameter ' #' shall be restored to the values they had before the function was executed. If the special built-in *return* (see [return](#)) is executed in the *compound-command*, the function completes and execution shall resume with the next command after the function call. +

Exit Status

The exit status of a function definition shall be zero if the function was declared successfully; otherwise, it shall be greater than zero. The exit status of a function invocation shall be the exit status of the last command executed by the function.

2.10 Shell Grammar

The following grammar defines the Shell Command Language. This formal syntax shall take precedence over the preceding text syntax description.

2.10.1 Shell Grammar Lexical Conventions

The input language to the shell must be first recognized at the character level. The resulting tokens shall be classified by their immediate context according to the following rules (applied in order). These rules shall be used to determine what a “token” is that is subject to parsing at the token level. The rules for token recognition in [Section 2.3](#) shall apply.

1. A <newline> shall be returned as the token identifier **NEWLINE**.
2. If the token is an operator, the token identifier for that operator shall result.
3. If the string consists solely of digits and the delimiter character is one of ' < ' or ' > ', the token identifier **IO_NUMBER** shall be returned.
4. Otherwise, the token identifier **TOKEN** results.

Further distinction on **TOKEN** is context-dependent. It may be that the same **TOKEN** yields **WORD**, a **NAME**, an **ASSIGNMENT**, or one of the reserved words below, dependent upon the context. Some of the productions in the grammar below are annotated with a rule number from the following list. When a **TOKEN** is seen where one of those annotated productions could be

used to reduce the symbol, the applicable rule shall be applied to convert the token identifier type of the **TOKEN** to a token identifier acceptable at that point in the grammar. The reduction shall then proceed based upon the token identifier type yielded by the rule applied. When more than one rule applies, the highest numbered rule shall apply (which in turn may refer to another rule). (Note that except in rule 7, the presence of an '=' in the token has no effect.)

The **WORD** tokens shall have the word expansion rules applied to them immediately before the associated command is executed, not at the time the command is parsed.

2.10.2 Shell Grammar Rules

1. [Command Name]

When the **TOKEN** is exactly a reserved word, the token identifier for that reserved word shall result. Otherwise, the token **WORD** shall be returned. Also, if the parser is in any state where only a reserved word could be the next correct token, proceed as above.

Note: Because at this point <quotation-mark> characters are retained in the token, quoted strings cannot be recognized as reserved words. This rule also implies that reserved words are not recognized except in certain positions in the input, such as after a <newline> or <semicolon>; the grammar presumes that if the reserved word is intended, it is properly delimited by the user, and does not attempt to reflect that requirement directly. Also note that line joining is done before tokenization, as described in [Section 2.2.1](#), so escaped <newline> characters are already removed at this point.

Rule 1 is not directly referenced in the grammar, but is referred to by other rules, or applies globally.

2. [Redirection to or from filename]

The expansions specified in [Section 2.7](#) shall occur. As specified there, exactly one field can result (or the result is unspecified), and there are additional requirements on pathname expansion.

3. [Redirection from here-document]

Quote removal shall be applied to the word to determine the delimiter that is used to find the end of the here-document that begins after the next <newline>.

4. [Case statement termination]

When the **TOKEN** is exactly the reserved word **esac**, the token identifier for **esac** shall result. Otherwise, the token **WORD** shall be returned.

5. [NAME in for]

When the **TOKEN** meets the requirements for a name (see XBD [Section 3.231](#)), the token identifier **NAME** shall result. Otherwise, the token **WORD** shall be returned.

6. [Third word of for and case]

a. [case only]

When the **TOKEN** is exactly the reserved word **in**, the token identifier for **in** shall result. Otherwise, the token **WORD** shall be returned.

b. [for only]

When the **TOKEN** is exactly the reserved word **in** or **do**, the token identifier for **in** or **do** shall result, respectively. Otherwise, the token **WORD** shall be returned.

(For a. and b.: As indicated in the grammar, a *linebreak* precedes the tokens **in** and **do**. If

74748 <newline> characters are present at the indicated location, it is the token after them that is
74749 treated in this fashion.)

74750 7. [Assignment preceding command name]

74751 a. [When the first word]

74752 If the **TOKEN** does not contain the character '=' , rule 1 is applied. Otherwise, 7b
74753 shall be applied.

74754 b. [Not the first word]

74755 If the **TOKEN** contains the <equals-sign> character:

74756 — If it begins with '=' , the token **WORD** shall be returned.

74757 — If all the characters preceding '=' form a valid name (see XBD [Section](#)
74758 [3.231](#)), the token **ASSIGNMENT_WORD** shall be returned. (Quoted
74759 characters cannot participate in forming a valid name.)

74760 — Otherwise, it is unspecified whether it is **ASSIGNMENT_WORD** or **WORD**
74761 that is returned.

74762 Assignment to the **NAME** shall occur as specified in [Section 2.9.1](#).

74763 8. [**NAME** in function]

74764 When the **TOKEN** is exactly a reserved word, the token identifier for that reserved word
74765 shall result. Otherwise, when the **TOKEN** meets the requirements for a name, the token
74766 identifier **NAME** shall result. Otherwise, rule 7 applies.

74767 9. [Body of function]

74768 Word expansion and assignment shall never occur, even when required by the rules
74769 above, when this rule is being parsed. Each **TOKEN** that might either be expanded or
74770 have assignment applied to it shall instead be returned as a single **WORD** consisting only
74771 of characters that are exactly the token described in [Section 2.3](#).

74772 /* -----
74773 The grammar symbols
74774 ----- */

74775 %token WORD
74776 %token ASSIGNMENT_WORD
74777 %token NAME
74778 %token NEWLINE
74779 %token IO_NUMBER

74780 /* The following are the operators mentioned above. */

74781 %token AND_IF OR_IF DSEMI
74782 /* '&&' '||' ';;' */

74783 %token DLESS DGREAT LESSAND GREATAND LESSGREAT DLESSDASH
74784 /* '<<' '>>' '<&' '>&' '<>' '<<-' */

74785 %token CLOBBER
74786 /* '>|' */

74787 /* The following are the reserved words. */

74788 %token If Then Else Elif Fi Do Done
74789 /* 'if' 'then' 'else' 'elif' 'fi' 'do' 'done' */

```

74790 %token Case Esac While Until For
74791 /* 'case' 'esac' 'while' 'until' 'for' */

74792 /* These are reserved words, not operator tokens, and are
74793    recognized when reserved words are recognized. */

74794 %token Lbrace Rbrace Bang
74795 /* '{' '}' '!' */

74796 %token In
74797 /* 'in' */

74798 /* -----
74799    The Grammar
74800    ----- */

74801 %start complete_command
74802 %%
74803 complete_command : list separator
74804                  | list
74805                  ;
74806 list              : list separator_op and_or
74807                  | and_or
74808                  ;
74809 and_or            : pipeline
74810                  | and_or AND_IF linebreak pipeline
74811                  | and_or OR_IF linebreak pipeline
74812                  ;
74813 pipeline          : pipe_sequence
74814                  | Bang pipe_sequence
74815                  ;
74816 pipe_sequence     : command
74817                  | pipe_sequence '|' linebreak command
74818                  ;
74819 command           : simple_command
74820                  | compound_command
74821                  | compound_command redirect_list
74822                  | function_definition
74823                  ;
74824 compound_command  : brace_group
74825                  | subshell
74826                  | for_clause
74827                  | case_clause
74828                  | if_clause
74829                  | while_clause
74830                  | until_clause
74831                  ;
74832 subshell          : '(' compound_list ')'
74833                  ;
74834 compound_list     : term
74835                  | newline_list term
74836                  | term separator
74837                  | newline_list term separator
74838                  ;
74839 term              : term separator and_or
74840                  | and_or

```

```

74841      ;
74842      for_clause      : For name linebreak                               do_group
74843                      | For name linebreak in                          sequential_sep do_group
74844                      | For name linebreak in wordlist sequential_sep do_group
74845      ;
74846      name            : NAME                                           /* Apply rule 5 */
74847      ;
74848      in              : In                                             /* Apply rule 6 */
74849      ;
74850      wordlist        : wordlist WORD
74851                      | WORD
74852      ;
74853      case_clause     : Case WORD linebreak in linebreak case_list      Esac
74854                      | Case WORD linebreak in linebreak case_list_ns Esac
74855                      | Case WORD linebreak in linebreak                  Esac
74856      ;
74857      case_list_ns    : case_list case_item_ns
74858                      | case_item_ns
74859      ;
74860      case_list       : case_list case_item
74861                      | case_item
74862      ;
74863      case_item_ns    : pattern ')' linebreak
74864                      | pattern ')' compound_list linebreak
74865                      | '(' pattern ')' linebreak
74866                      | '(' pattern ')' compound_list linebreak
74867      ;
74868      case_item       : pattern ')' linebreak DSEMI linebreak
74869                      | pattern ')' compound_list DSEMI linebreak
74870                      | '(' pattern ')' linebreak DSEMI linebreak
74871                      | '(' pattern ')' compound_list DSEMI linebreak
74872      ;
74873      pattern         : WORD                                           /* Apply rule 4 */
74874                      | pattern '|' WORD                               /* Do not apply rule 4 */
74875      ;
74876      if_clause       : If compound_list Then compound_list else_part Fi
74877                      | If compound_list Then compound_list           Fi
74878      ;
74879      else_part       : Elif compound_list Then compound_list          |
74880                      | Elif compound_list Then compound_list else_part |
74881                      | Else compound_list
74882      ;
74883      while_clause    : While compound_list do_group
74884      ;
74885      until_clause    : Until compound_list do_group
74886      ;
74887      function_definition : fname '(' ')' linebreak function_body
74888      ;
74889      function_body   : compound_command                               /* Apply rule 9 */
74890                      | compound_command redirect_list               /* Apply rule 9 */
74891      ;
74892      fname           : NAME                                           /* Apply rule 8 */
74893      ;

```

```

74894 brace_group      : Lbrace compound_list Rbrace
74895                    ;
74896 do_group           : Do compound_list Done           /* Apply rule 6 */
74897                    ;
74898 simple_command     : cmd_prefix cmd_word cmd_suffix
74899                    | cmd_prefix cmd_word
74900                    | cmd_prefix
74901                    | cmd_name cmd_suffix
74902                    | cmd_name
74903                    ;
74904 cmd_name            : WORD                               /* Apply rule 7a */
74905                    ;
74906 cmd_word            : WORD                               /* Apply rule 7b */
74907                    ;
74908 cmd_prefix          : io_redirect
74909                    | cmd_prefix io_redirect
74910                    | ASSIGNMENT_WORD
74911                    | cmd_prefix ASSIGNMENT_WORD
74912                    ;
74913 cmd_suffix          : io_redirect
74914                    | cmd_suffix io_redirect
74915                    | WORD
74916                    | cmd_suffix WORD
74917                    ;
74918 redirect_list      : io_redirect
74919                    | redirect_list io_redirect
74920                    ;
74921 io_redirect         : io_file
74922                    | IO_NUMBER io_file
74923                    | io_here
74924                    | IO_NUMBER io_here
74925                    ;
74926 io_file            : '<' filename
74927                    | LESSAND filename
74928                    | '>' filename
74929                    | GREATAND filename
74930                    | DGREAT filename
74931                    | LESSGREAT filename
74932                    | CLOBBER filename
74933                    ;
74934 filename            : WORD                               /* Apply rule 2 */
74935                    ;
74936 io_here             : DLESS here_end
74937                    | DLESSDASH here_end
74938                    ;
74939 here_end            : WORD                               /* Apply rule 3 */
74940                    ;
74941 newline_list        : NEWLINE
74942                    | newline_list NEWLINE
74943                    ;
74944 linebreak           : newline_list
74945                    | /* empty */
74946                    ;

```

```

74947 separator_op      : '&'
74948                   | ';'
74949                   ;
74950 separator           : separator_op linebreak
74951                   | newline_list
74952                   ;
74953 sequential_sep      : ';' linebreak
74954                   | newline_list
74955                   ;

```

74956 2.11 Signals and Error Handling

74957 When a command is in an asynchronous list, it shall inherit from the shell a signal action of
 74958 ignored (SIG_IGN) for the SIGQUIT and SIGINT signals, and may inherit a signal mask in
 74959 which SIGQUIT and SIGINT are blocked. Otherwise, the signal actions and signal mask
 74960 inherited by the command shall be the same as those inherited by the shell from its parent
 74961 unless a signal action is modified by the *trap* special built-in (see *trap*)

74962 When a signal for which a trap has been set is received while the shell is waiting for the
 74963 completion of a utility executing a foreground command, the trap associated with that signal
 74964 shall not be executed until after the foreground command has completed. When the shell is
 74965 waiting, by means of the *wait* utility, for asynchronous commands to complete, the reception of a
 74966 signal for which a trap has been set shall cause the *wait* utility to return immediately with an exit
 74967 status >128, immediately after which the trap associated with that signal shall be taken.

74968 If multiple signals are pending for the shell for which there are associated trap actions, the order
 74969 of execution of trap actions is unspecified.

74970 2.12 Shell Execution Environment

74971 A shell execution environment consists of the following:

- 74972 • Open files inherited upon invocation of the shell, plus open files controlled by *exec*
- 74973 • Working directory as set by *cd*
- 74974 • File creation mask set by *umask*
- 74975 • Current traps set by *trap*
- 74976 • Shell parameters that are set by variable assignment (see the *set* special built-in) or from
 74977 the System Interfaces volume of POSIX.1-2008 environment inherited by the shell when it
 74978 begins (see the *export* special built-in)
- 74979 • Shell functions; see [Section 2.9.5](#)
- 74980 • Options turned on at invocation or by *set*
- 74981 • Process IDs of the last commands in asynchronous lists known to this shell environment;
 74982 see [Section 2.9.3.1](#)
- 74983 • Shell aliases; see [Section 2.3.1](#)

74984 Utilities other than the special built-ins (see [Section 2.14](#)) shall be invoked in a separate
 74985 environment that consists of the following. The initial value of these objects shall be the same as

that for the parent shell, except as noted below.

- Open files inherited on invocation of the shell, open files controlled by the *exec* special built-in plus any modifications, and additions specified by any redirections to the utility
- Current working directory
- File creation mask
- If the utility is a shell script, traps caught by the shell shall be set to the default values and traps ignored by the shell shall be set to be ignored by the utility; if the utility is not a shell script, the trap actions (default or ignore) shall be mapped into the appropriate signal handling actions for the utility
- Variables with the *export* attribute, along with those explicitly exported for the duration of the command, shall be passed to the utility environment variables

The environment of the shell process shall not be changed by the utility unless explicitly specified by the utility description (for example, *cd* and *umask*).

A subshell environment shall be created as a duplicate of the shell environment, except that signal traps that are not being ignored shall be set to the default action. Changes made to the subshell environment shall not affect the shell environment. Command substitution, commands that are grouped with parentheses, and asynchronous lists shall be executed in a subshell environment. Additionally, each command of a multi-command pipeline is in a subshell environment; as an extension, however, any or all commands in a pipeline may be executed in the current environment. All other commands shall be executed in the current shell environment.

2.13 Pattern Matching Notation

The pattern matching notation described in this section is used to specify patterns for matching strings in the shell. Historically, pattern matching notation is related to, but slightly different from, the regular expression notation described in XBD [Chapter 9](#). For this reason, the description of the rules for this pattern matching notation are based on the description of regular expression notation, modified to account for the differences.

2.13.1 Patterns Matching a Single Character

The following patterns matching a single character shall match a single character: ordinary characters, special pattern characters, and pattern bracket expressions. The pattern bracket expression also shall match a single collating element. A `<backslash>` character shall escape the following character. The escaping `<backslash>` shall be discarded.

An ordinary character is a pattern that shall match itself. It can be any character in the supported character set except for NUL, those special shell characters in [Section 2.2](#) that require quoting, and the following three special pattern characters. Matching shall be based on the bit pattern used for encoding the character, not on the graphic representation of the character. If any character (ordinary, shell special, or pattern special) is quoted, that pattern shall match the character itself. The shell special characters always require quoting.

When unquoted and outside a bracket expression, the following three characters shall have special meaning in the specification of patterns:

- ? A <question-mark> is a pattern that shall match any character.
- * An <asterisk> is a pattern that shall match multiple characters, as described in [Section 2.13.2](#).
- [If an open bracket introduces a bracket expression as in XBD [Section 9.3.5](#), except that the <exclamation-mark> character ('!') shall replace the <circumflex> character ('^') in its role in a non-matching list in the regular expression notation, it shall introduce a pattern bracket expression. A bracket expression starting with an unquoted <circumflex> character produces unspecified results. Otherwise, '[' shall match the character itself.

When pattern matching is used where shell quote removal is not performed (such as in the argument to the *find -name* primary when *find* is being called using one of the *exec* functions as defined in the System Interfaces volume of POSIX.1-2008, or in the *pattern* argument to the *fnmatch()* function), special characters can be escaped to remove their special meaning by preceding them with a <backslash> character. This escaping <backslash> is discarded. The sequence "\\\" represents one literal <backslash>. All of the requirements and effects of quoting on ordinary, shell special, and special pattern characters shall apply to escaping in this context.

2.13.2 Patterns Matching Multiple Characters

The following rules are used to construct patterns matching multiple characters from patterns matching a single character:

1. The <asterisk> ('*') is a pattern that shall match any string, including the null string.
2. The concatenation of patterns matching a single character is a valid pattern that shall match the concatenation of the single characters or collating elements matched by each of the concatenated patterns.
3. The concatenation of one or more patterns matching a single character with one or more <asterisk> characters is a valid pattern. In such patterns, each <asterisk> shall match a string of zero or more characters, matching the greatest possible number of characters that still allows the remainder of the pattern to match the string.

2.13.3 Patterns Used for Filename Expansion

The rules described so far in [Section 2.13.1](#) and [Section 2.13.2](#) are qualified by the following rules that apply when pattern matching notation is used for filename expansion:

1. The <slash> character in a pathname shall be explicitly matched by using one or more <slash> characters in the pattern; it shall neither be matched by the <asterisk> or <question-mark> special characters nor by a bracket expression. <slash> characters in the pattern shall be identified before bracket expressions; thus, a <slash> cannot be included in a pattern bracket expression used for filename expansion. If a <slash> character is found following an unescaped <left-square-bracket> character before a corresponding <right-square-bracket> is found, the open bracket shall be treated as an ordinary character. For example, the pattern "a[b/c]d" does not match such pathnames as **abd** or **a/d**. It only matches a pathname of literally **a[b/c]d**.
2. If a filename begins with a <period> ('.'), the <period> shall be explicitly matched by using a <period> as the first character of the pattern or immediately following a <slash> character. The leading <period> shall not be matched by:

- The <asterisk> or <question-mark> special characters
- A bracket expression containing a non-matching list, such as "[!a]", a range expression, such as "[%–0]", or a character class expression, such as "[[:punct:]]"

It is unspecified whether an explicit <period> in a bracket expression matching list, such as "[.abc]", can match a leading <period> in a filename.

3. Specified patterns shall be matched against existing filenames and pathnames, as appropriate. Each component that contains a pattern character shall require read permission in the directory containing that component. Any component, except the last, that does not contain a pattern character shall require search permission. For example, given the pattern:

```
/foo/bar/x*/bam
```

search permission is needed for directories **/** and **foo**, search and read permissions are needed for directory **bar**, and search permission is needed for each **x*** directory. If the pattern matches any existing filenames or pathnames, the pattern shall be replaced with those filenames and pathnames, sorted according to the collating sequence in effect in the current locale.

If the pattern contains an open bracket ('[') that does not introduce a bracket expression as in XBD [Section 9.3.5](#), it is unspecified whether other unquoted pattern matching characters within the same slash-delimited component of the pattern retain their special meanings or are treated as ordinary characters. For example, the pattern "a*[/b*" may match all filenames beginning with 'b' in the directory "a*[or it may match all filenames beginning with 'b' in all directories with names beginning with 'a' and ending with '['.

If the pattern does not match any existing filenames or pathnames, the pattern string shall be left unchanged.

2.14 Special Built-In Utilities

The following “special built-in” utilities shall be supported in the shell command language. The output of each command, if any, shall be written to standard output, subject to the normal redirection and piping possible with all commands.

The term “built-in” implies that the shell can execute the utility directly and does not need to search for it. An implementation may choose to make any utility a built-in; however, the special built-in utilities described here differ from regular built-in utilities in two respects:

1. A syntax error in a special built-in utility may cause a shell executing that utility to abort, while a syntax error in a regular built-in utility shall not cause a shell executing that utility to abort. (See [Section 2.8.1](#) for the consequences of errors on interactive and non-interactive shells.) If a special built-in utility encountering a syntax error does not abort the shell, its exit value shall be non-zero.
2. Variable assignments specified with special built-in utilities remain in effect after the built-in completes; this shall not be the case with a regular built-in or other utility.

The special built-in utilities in this section need not be provided in a manner accessible via the *exec* family of functions defined in the System Interfaces volume of POSIX.1-2008.

Some of the special built-ins are described as conforming to XBD [Section 12.2](#). For those that are

75110 not, the requirement in [Section 1.4](#) that "--" be recognized as a first argument to be discarded
75111 does not apply and a conforming application shall not use that argument.

75112 **NAME**

75113 break — exit from for, while, or until loop

75114 **SYNOPSIS**75115 break [*n*]75116 **DESCRIPTION**

75117 The *break* utility shall exit from the smallest enclosing **for**, **while**, or **until** loop, if any; or from
75118 the *n*th enclosing loop if *n* is specified. The value of *n* is an unsigned decimal integer greater
75119 than or equal to 1. The default shall be equivalent to *n*=1. If *n* is greater than the number of
75120 enclosing loops, the outermost enclosing loop shall be exited. Execution shall continue with the
75121 command immediately following the loop.

75122 **OPTIONS**

75123 None.

75124 **OPERANDS**

75125 See the DESCRIPTION.

75126 **STDIN**

75127 Not used.

75128 **INPUT FILES**

75129 None.

75130 **ENVIRONMENT VARIABLES**

75131 None.

75132 **ASYNCHRONOUS EVENTS**

75133 Default.

75134 **STDOUT**

75135 Not used.

75136 **STDERR**

75137 The standard error shall be used only for diagnostic messages.

75138 **OUTPUT FILES**

75139 None.

75140 **EXTENDED DESCRIPTION**

75141 None.

75142 **EXIT STATUS**

75143 0 Successful completion.

75144 >0 The *n* value was not an unsigned decimal integer greater than or equal to 1.75145 **CONSEQUENCES OF ERRORS**

75146 Default.

APPLICATION USAGE

None.

EXAMPLES

```
for i in *
do
    if test -d "$i"
    then break
    fi
done
```

RATIONALE

In early proposals, consideration was given to expanding the syntax of *break* and *continue* to refer to a label associated with the appropriate loop as a preferable alternative to the *n* method. However, this volume of POSIX.1-2008 does reserve the name space of command names ending with a <colon>. It is anticipated that a future implementation could take advantage of this and provide something like:

```
outofloop: for i in a b c d e
do
    for j in 0 1 2 3 4 5 6 7 8 9
    do
        if test -r "${i}${j}"
        then break outofloop
        fi
    done
done
```

and that this might be standardized after implementation experience is achieved.

FUTURE DIRECTIONS

None.

SEE ALSO

[Section 2.14](#)

CHANGE HISTORY**Issue 6**

IEEE Std 1003.1-2001/Cor 1-2002, item XCU/TC1/D6/5 is applied so that the reference page sections use terms as described in the Utility Description Defaults ([Section 1.4](#)). No change in behavior is intended.

75181 **NAME**

75182 colon — null utility

75183 **SYNOPSIS**75184 : [*argument...*]75185 **DESCRIPTION**

75186 This utility shall only expand command *arguments*. It is used when a command is needed, as in
75187 the **then** condition of an **if** command, but nothing is to be done by the command.

75188 **OPTIONS**

75189 None.

75190 **OPERANDS**

75191 See the DESCRIPTION.

75192 **STDIN**

75193 Not used.

75194 **INPUT FILES**

75195 None.

75196 **ENVIRONMENT VARIABLES**

75197 None.

75198 **ASYNCHRONOUS EVENTS**

75199 Default.

75200 **STDOUT**

75201 Not used.

75202 **STDERR**

75203 The standard error shall be used only for diagnostic messages.

75204 **OUTPUT FILES**

75205 None.

75206 **EXTENDED DESCRIPTION**

75207 None.

75208 **EXIT STATUS**

75209 Zero.

75210 **CONSEQUENCES OF ERRORS**

75211 Default.

75212 **APPLICATION USAGE**

75213 None.

75214 **EXAMPLES**

```
75215       : ${X=abc}  
75216       if       false  
75217       then     :  
75218       else     echo $X  
75219       fi  
75220       abc
```

75221 As with any of the special built-ins, the null utility can also have variable assignments and
75222 redirections associated with it, such as:

75223 x=y : > z

75224 which sets variable *x* to the value *y* (so that it persists after the null utility completes) and creates
75225 or truncates file *z*.

75226 **RATIONALE**

75227 None.

75228 **FUTURE DIRECTIONS**

75229 None.

75230 **SEE ALSO**

75231 [Section 2.14](#)

75232 **CHANGE HISTORY**

75233 **Issue 6**

75234 IEEE Std 1003.1-2001/Cor 1-2002, item XCU/TC1/D6/5 is applied so that the reference page
75235 sections use terms as described in the Utility Description Defaults ([Section 1.4](#)). No change in
75236 behavior is intended.

75237 **Issue 7**

75238 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

75239 **NAME**

75240 continue — continue for, while, or until loop

75241 **SYNOPSIS**75242 continue [*n*]75243 **DESCRIPTION**

75244 The *continue* utility shall return to the top of the smallest enclosing **for**, **while**, or **until** loop, or to
75245 the top of the *n*th enclosing loop, if *n* is specified. This involves repeating the condition list of a
75246 **while** or **until** loop or performing the next assignment of a **for** loop, and re-executing the loop if
75247 appropriate.

75248 The value of *n* is a decimal integer greater than or equal to 1. The default shall be equivalent to
75249 *n*=1. If *n* is greater than the number of enclosing loops, the outermost enclosing loop shall be
75250 used.

75251 **OPTIONS**

75252 None.

75253 **OPERANDS**

75254 See the DESCRIPTION.

75255 **STDIN**

75256 Not used.

75257 **INPUT FILES**

75258 None.

75259 **ENVIRONMENT VARIABLES**

75260 None.

75261 **ASYNCHRONOUS EVENTS**

75262 Default.

75263 **STDOUT**

75264 Not used.

75265 **STDERR**

75266 The standard error shall be used only for diagnostic messages.

75267 **OUTPUT FILES**

75268 None.

75269 **EXTENDED DESCRIPTION**

75270 None.

75271 **EXIT STATUS**

75272 0 Successful completion.

75273 >0 The *n* value was not an unsigned decimal integer greater than or equal to 1.75274 **CONSEQUENCES OF ERRORS**

75275 Default.

75276 **APPLICATION USAGE**

75277 None.

75278 **EXAMPLES**

```
75279     for i in *
75280     do
75281         if test -d "$i"
75282         then continue
75283         fi
75284         printf '"%s" is not a directory.\n' "$i"
75285     done
```

75286 **RATIONALE**

75287 None.

75288 **FUTURE DIRECTIONS**

75289 None.

75290 **SEE ALSO**75291 [Section 2.14](#)75292 **CHANGE HISTORY**75293 **Issue 6**

75294 IEEE Std 1003.1-2001/Cor 1-2002, item XCU/TC1/D6/5 is applied so that the reference page
75295 sections use terms as described in the Utility Description Defaults ([Section 1.4](#)). No change in
75296 behavior is intended.

75297 **Issue 7**75298 The example is changed to use the *printf* utility rather than *echo*.

75299 **NAME**

75300 dot — execute commands in the current environment

75301 **SYNOPSIS**75302 . *file*75303 **DESCRIPTION**75304 The shell shall execute commands from the *file* in the current environment.

75305 If *file* does not contain a <slash>, the shell shall use the search path specified by *PATH* to find the
75306 directory containing *file*. Unlike normal command search, however, the file searched for by the
75307 *dot* utility need not be executable. If no readable file is found, a non-interactive shell shall abort;
75308 an interactive shell shall write a diagnostic message to standard error, but this condition shall
75309 not be considered a syntax error.

75310 **OPTIONS**

75311 None.

75312 **OPERANDS**

75313 See the DESCRIPTION.

75314 **STDIN**

75315 Not used.

75316 **INPUT FILES**

75317 See the DESCRIPTION.

75318 **ENVIRONMENT VARIABLES**

75319 See the DESCRIPTION.

75320 **ASYNCHRONOUS EVENTS**

75321 Default.

75322 **STDOUT**

75323 Not used.

75324 **STDERR**

75325 The standard error shall be used only for diagnostic messages.

75326 **OUTPUT FILES**

75327 None.

75328 **EXTENDED DESCRIPTION**

75329 None.

75330 **EXIT STATUS**

75331 If no readable file was found or if the commands in the file could not be parsed, and the shell is |
75332 interactive (and therefore does not abort; see [Section 2.8.1](#)), the exit status shall be non-zero. |
75333 Otherwise, return the value of the last command executed, or a zero exit status if no command is |
75334 executed.

75335 **CONSEQUENCES OF ERRORS**

75336 Default.

75337 **APPLICATION USAGE**

75338 None.

75339 **EXAMPLES**

75340 cat foobar

75341 **foo=hello bar=world**

75342 . ./foobar

75343 echo \$foo \$bar

75344 **hello world**75345 **RATIONALE**

75346 Some older implementations searched the current directory for the *file*, even if the value of *PATH*
75347 disallowed it. This behavior was omitted from this volume of POSIX.1-2008 due to concerns
75348 about introducing the susceptibility to trojan horses that the user might be trying to avoid by
75349 leaving **dot** out of *PATH*.

75350 The KornShell version of *dot* takes optional arguments that are set to the positional parameters.
75351 This is a valid extension that allows a *dot* script to behave identically to a function.

75352 **FUTURE DIRECTIONS**

75353 None.

75354 **SEE ALSO**75355 [Section 2.14, *return*](#)75356 **CHANGE HISTORY**75357 **Issue 6**

75358 IEEE Std 1003.1-2001/Cor 1-2002, item XCU/TC1/D6/5 is applied so that the reference page
75359 sections use terms as described in the Utility Description Defaults ([Section 1.4](#)). No change in
75360 behavior is intended.

75361 **Issue 7**

75362 SD5-XCU-ERN-164 is applied.

75363 POSIX.1-2008, Technical Corrigendum 1, XCU/TC1-2008/0038 [114] and XCU/TC1-2008/0039 +
75364 [214] are applied.

75365 **NAME**

75366 eval — construct command by concatenating arguments

75367 **SYNOPSIS**75368 eval [*argument...*]75369 **DESCRIPTION**75370 The *eval* utility shall construct a command by concatenating *arguments* together, separating each
75371 with a <space> character. The constructed command shall be read and executed by the shell.75372 **OPTIONS**

75373 None.

75374 **OPERANDS**

75375 See the DESCRIPTION.

75376 **STDIN**

75377 Not used.

75378 **INPUT FILES**

75379 None.

75380 **ENVIRONMENT VARIABLES**

75381 None.

75382 **ASYNCHRONOUS EVENTS**

75383 Default.

75384 **STDOUT**

75385 Not used.

75386 **STDERR**

75387 The standard error shall be used only for diagnostic messages.

75388 **OUTPUT FILES**

75389 None.

75390 **EXTENDED DESCRIPTION**

75391 None.

75392 **EXIT STATUS**75393 If there are no *arguments*, or only null *arguments*, *eval* shall return a zero exit status; otherwise, it
75394 shall return the exit status of the command defined by the string of concatenated *arguments*
75395 separated by <space> characters, or a non-zero exit status if the concatenation could not be
75396 parsed as a command and the shell is interactive (and therefore did not abort).75397 **CONSEQUENCES OF ERRORS**

75398 Default.

75399 **APPLICATION USAGE**75400 Since *eval* is not required to recognize the "--" end of options delimiter, in cases where the
75401 argument(s) to *eval* might begin with '-' it is recommended that the first argument is prefixed
75402 by a string that will not alter the commands to be executed, such as a <space> character:

75403 eval " \$commands" |

75404 or: |

75405 eval " \$(some_command)" |

EXAMPLES

```
foo=10 x=foo
y=' $' $x
echo $y
$foo
eval y=' $' $x
echo $y
10
```

RATIONALE

This standard allows, but does not require, *eval* to recognize "--". Although this means applications cannot use "--" to protect against options supported as an extension (or errors reported for unsupported options), the nature of the *eval* utility is such that other means can be used to provide this protection (see APPLICATION USAGE above).

FUTURE DIRECTIONS

None.

SEE ALSO

[Section 2.14](#)

CHANGE HISTORY**Issue 6**

IEEE Std 1003.1-2001/Cor 1-2002, item XCU/TC1/D6/5 is applied so that the reference page sections use terms as described in the Utility Description Defaults ([Section 1.4](#)). No change in behavior is intended.

Issue 7

SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

POSIX.1-2008, Technical Corrigendum 1, XCU/TC1-2008/0040 [114], XCU/TC1-2008/0041 [163], + and XCU/TC1-2008/0042 [163] are applied.

75432 **NAME**

75433 exec — execute commands and open, close, or copy file descriptors

75434 **SYNOPSIS**75435 exec [*command* [*argument...*]]75436 **DESCRIPTION**75437 The *exec* utility shall open, close, and/or copy file descriptors as specified by any redirections as
75438 part of the command.75439 If *exec* is specified without *command* or *arguments*, and any file descriptors with numbers greater
75440 than 2 are opened with associated redirection statements, it is unspecified whether those file
75441 descriptors remain open when the shell invokes another utility. Scripts concerned that child
75442 shells could misuse open file descriptors can always close them explicitly, as shown in one of the
75443 following examples.75444 If *exec* is specified with *command*, it shall replace the shell with *command* without creating a new
75445 process. If *arguments* are specified, they shall be arguments to *command*. Redirection affects the
75446 current shell execution environment.75447 **OPTIONS**

75448 None.

75449 **OPERANDS**

75450 See the DESCRIPTION.

75451 **STDIN**

75452 Not used.

75453 **INPUT FILES**

75454 None.

75455 **ENVIRONMENT VARIABLES**

75456 None.

75457 **ASYNCHRONOUS EVENTS**

75458 Default.

75459 **STDOUT**

75460 Not used.

75461 **STDERR**

75462 The standard error shall be used only for diagnostic messages.

75463 **OUTPUT FILES**

75464 None.

75465 **EXTENDED DESCRIPTION**

75466 None.

75467 **EXIT STATUS**75468 If *command* is specified, *exec* shall not return to the shell; rather, the exit status of the process shall
75469 be the exit status of the program implementing *command*, which overlaid the shell. If *command* is
75470 not found, the exit status shall be 127. If *command* is found, but it is not an executable utility, the
75471 exit status shall be 126. If a redirection error occurs (see [Section 2.8.1](#)), the shell shall exit with a
75472 value in the range 1–125. Otherwise, *exec* shall return a zero exit status.

CONSEQUENCES OF ERRORS

Default.

APPLICATION USAGE

None.

EXAMPLES

Open *readfile* as file descriptor 3 for reading:

```
exec 3< readfile
```

Open *writefile* as file descriptor 4 for writing:

```
exec 4> writefile
```

Make file descriptor 5 a copy of file descriptor 0:

```
exec 5<&0
```

Close file descriptor 3:

```
exec 3<&-
```

Cat the file **maggie** by replacing the current shell with the *cat* utility:

```
exec cat maggie
```

RATIONALE

Most historical implementations were not conformant in that:

```
foo=bar exec cmd
```

did not pass **foo** to **cmd**.

FUTURE DIRECTIONS

None.

SEE ALSO

[Section 2.14](#)

CHANGE HISTORY**Issue 6**

IEEE Std 1003.1-2001/Cor 1-2002, item XCU/TC1/D6/5 is applied so that the reference page sections use terms as described in the Utility Description Defaults ([Section 1.4](#)). No change in behavior is intended.

Issue 7

SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

75503 NAME

75504 exit — cause the shell to exit

75505 SYNOPSIS

75506 exit [*n*]

75507 DESCRIPTION

75508 The *exit* utility shall cause the shell to exit with the exit status specified by the unsigned decimal
75509 integer *n*. If *n* is specified, but its value is not between 0 and 255 inclusively, the exit status is
75510 undefined.

75511 A *trap* on **EXIT** shall be executed before the shell terminates, except when the *exit* utility is
75512 invoked in that *trap* itself, in which case the shell shall exit immediately.

75513 OPTIONS

75514 None.

75515 OPERANDS

75516 See the DESCRIPTION.

75517 STDIN

75518 Not used.

75519 INPUT FILES

75520 None.

75521 ENVIRONMENT VARIABLES

75522 None.

75523 ASYNCHRONOUS EVENTS

75524 Default.

75525 STDOUT

75526 Not used.

75527 STDERR

75528 The standard error shall be used only for diagnostic messages.

75529 OUTPUT FILES

75530 None.

75531 EXTENDED DESCRIPTION

75532 None.

75533 EXIT STATUS

75534 The exit status shall be *n*, if specified. Otherwise, the value shall be the exit value of the last
75535 command executed, or zero if no command was executed. When *exit* is executed in a *trap* action,
75536 the last command is considered to be the command that executed immediately preceding the
75537 *trap* action.

75538 CONSEQUENCES OF ERRORS

75539 Default.

75540 APPLICATION USAGE

75541 None.

75542 EXAMPLES

75543 Exit with a *true* value:

75544 `exit 0`

75545 Exit with a *false* value:

75546 `exit 1`

75547 RATIONALE

75548 As explained in other sections, certain exit status values have been reserved for special uses and
75549 should be used by applications only for those purposes:

75550 126 A file to be executed was found, but it was not an executable utility.

75551 127 A utility to be executed was not found.

75552 >128 A command was interrupted by a signal.

75553 FUTURE DIRECTIONS

75554 None.

75555 SEE ALSO

75556 [Section 2.14](#)

75557 CHANGE HISTORY**75558 Issue 6**

75559 IEEE Std 1003.1-2001/Cor 1-2002, item XCU/TC1/D6/5 is applied so that the reference page
75560 sections use terms as described in the Utility Description Defaults ([Section 1.4](#)). No change in
75561 behavior is intended.

75562 **NAME**

75563 export — set the export attribute for variables

75564 **SYNOPSIS**75565 export name[=*word*] . . .

75566 export -p

75567 **DESCRIPTION**

75568 The shell shall give the *export* attribute to the variables corresponding to the specified *names*,
 75569 which shall cause them to be in the environment of subsequently executed commands. If the
 75570 name of a variable is followed by =*word*, then the value of that variable shall be set to *word*.

75571 The *export* special built-in shall support XBD [Section 12.2](#).

75572 When -p is specified, *export* shall write to the standard output the names and values of all
 75573 exported variables, in the following format:

75574 "export %s=%s\n", <name>, <value>

75575 if *name* is set, and:

75576 "export %s\n", <name>

75577 if *name* is unset.

75578 The shell shall format the output, including the proper use of quoting, so that it is suitable for
 75579 reinput to the shell as commands that achieve the same exporting results, except:

- 75580 1. Read-only variables with values cannot be reset.
- 75581 2. Variables that were unset at the time they were output need not be reset to the unset state
 75582 if a value is assigned to the variable between the time the state was saved and the time at
 75583 which the saved output is reinput to the shell.

75584 When no arguments are given, the results are unspecified. If a variable assignment precedes the
 75585 command name of *export* but that variable is not also listed as an operand of *export*, then that
 75586 variable shall be set in the current shell execution environment after the completion of the *export*
 75587 command, but it is unspecified whether that variable is marked for export.

75588 **OPTIONS**

75589 See the DESCRIPTION.

75590 **OPERANDS**

75591 See the DESCRIPTION.

75592 **STDIN**

75593 Not used.

75594 **INPUT FILES**

75595 None.

75596 **ENVIRONMENT VARIABLES**

75597 None.

75598 **ASYNCHRONOUS EVENTS**

75599 Default.

75600 **STDOUT**

75601 See the DESCRIPTION.

75602 STDERR

75603 The standard error shall be used only for diagnostic messages.

75604 OUTPUT FILES

75605 None.

75606 EXTENDED DESCRIPTION

75607 None.

75608 EXIT STATUS

75609 Zero.

75610 CONSEQUENCES OF ERRORS

75611 Default.

75612 APPLICATION USAGE

75613 None.

75614 EXAMPLES

75615 Export *PWD* and *HOME* variables:

75616 `export PWD HOME`

75617 Set and export the *PATH* variable:

75618 `export PATH=/local/bin:$PATH`

75619 Save and restore all exported variables:

75620 `export -p > temp-file`

75621 `unset a lot of variables`

75622 `... processing`

75623 `. temp-file`

75624 RATIONALE

75625 Some historical shells use the no-argument case as the functional equivalent of what is required
75626 here with **-p**. This feature was left unspecified because it is not historical practice in all shells,
75627 and some scripts may rely on the now-unspecified results on their implementations. Attempts to
75628 specify the **-p** output as the default case were unsuccessful in achieving consensus. The **-p**
75629 option was added to allow portable access to the values that can be saved and then later restored
75630 using; for example, a *dot* script.

75631 FUTURE DIRECTIONS

75632 None.

75633 SEE ALSO

75634 [Section 2.14](#)

75635 [XBD Section 12.2](#)

75636 CHANGE HISTORY**75637 Issue 6**

75638 IEEE PASC Interpretation 1003.2 #203 is applied, clarifying the format when a variable is unset.

75639 IEEE Std 1003.1-2001/Cor 1-2002, item XCU/TC1/D6/5 is applied so that the reference page
75640 sections use terms as described in the Utility Description Defaults ([Section 1.4](#)). No change in
75641 behavior is intended.

75642 IEEE Std 1003.1-2001/Cor 1-2002, item XCU/TC1/D6/6 is applied, adding the following text to
75643 the end of the first paragraph of the DESCRIPTION: "If the name of a variable is followed by

75644 =*word*, then the value of that variable shall be set to *word*.". The reason for this change is that the
75645 SYNOPSIS for *export* includes:

75646 `export name[=word] . . .`

75647 but the meaning of the optional "*=word*" is never explained in the text.

75648 **Issue 7**

75649 POSIX.1-2008, Technical Corrigendum 1, XCU/TC1-2008/0043 [352] is applied.

+

75650 **NAME**

75651 readonly — set the readonly attribute for variables

75652 **SYNOPSIS**75653 readonly name[=*word*] . . .

75654 readonly -p

75655 **DESCRIPTION**

75656 The variables whose *names* are specified shall be given the *readonly* attribute. The values of
 75657 variables with the *readonly* attribute cannot be changed by subsequent assignment, nor can those
 75658 variables be unset by the *unset* utility. If the name of a variable is followed by =*word*, then the
 75659 value of that variable shall be set to *word*.

75660 The *readonly* special built-in shall support XBD [Section 12.2](#).

75661 When -p is specified, *readonly* writes to the standard output the names and values of all read-
 75662 only variables, in the following format:

75663 "readonly %s=%s\n", <*name*>, <*value*>75664 if *name* is set, and75665 "readonly %s\n", <*name*>75666 if *name* is unset.

75667 The shell shall format the output, including the proper use of quoting, so that it is suitable for
 75668 reinput to the shell as commands that achieve the same value and *readonly* attribute-setting
 75669 results in a shell execution environment in which:

- 75670 1. Variables with values at the time they were output do not have the *readonly* attribute set.
- 75671 2. Variables that were unset at the time they were output do not have a value at the time at
 75672 which the saved output is reinput to the shell.

75673 When no arguments are given, the results are unspecified.

75674 **OPTIONS**

75675 See the DESCRIPTION.

75676 **OPERANDS**

75677 See the DESCRIPTION.

75678 **STDIN**

75679 Not used.

75680 **INPUT FILES**

75681 None.

75682 **ENVIRONMENT VARIABLES**

75683 None.

75684 **ASYNCHRONOUS EVENTS**

75685 Default.

75686 **STDOUT**

75687 See the DESCRIPTION.

75688 **STDERR**

75689 The standard error shall be used only for diagnostic messages.

75690 **OUTPUT FILES**

75691 None.

75692 **EXTENDED DESCRIPTION**

75693 None.

75694 **EXIT STATUS**

75695 Zero.

75696 **CONSEQUENCES OF ERRORS**

75697 Default.

75698 **APPLICATION USAGE**

75699 None.

75700 **EXAMPLES**

75701 readonly HOME PWD

75702 **RATIONALE**

75703 Some historical shells preserve the *readonly* attribute across separate invocations. This volume of
75704 POSIX.1-2008 allows this behavior, but does not require it.

75705 The **-p** option allows portable access to the values that can be saved and then later restored
75706 using, for example, a *dot* script. Also see the RATIONALE for *export* for a description of the no-
75707 argument and **-p** output cases and a related example.

75708 Read-only functions were considered, but they were omitted as not being historical practice or
75709 particularly useful. Furthermore, functions must not be read-only across invocations to preclude
75710 “spoofing” (spoofing is the term for the practice of creating a program that acts like a well-
75711 known utility with the intent of subverting the real intent of the user) of administrative or
75712 security-relevant (or security-conscious) shell scripts.

75713 **FUTURE DIRECTIONS**

75714 None.

75715 **SEE ALSO**

75716 [Section 2.14](#)

75717 XBD [Section 12.2](#)

75718 **CHANGE HISTORY**75719 **Issue 6**

75720 IEEE PASC Interpretation 1003.2 #203 is applied, clarifying the format when a variable is unset.

75721 IEEE Std 1003.1-2001/Cor 1-2002, item XCU/TC1/D6/5 is applied so that the reference page
75722 sections use terms as described in the Utility Description Defaults ([Section 1.4](#)). No change in
75723 behavior is intended.

75724 IEEE Std 1003.1-2001/Cor 1-2002, item XCU/TC1/D6/7 is applied, adding the following text to
75725 the end of the first paragraph of the DESCRIPTION: “If the name of a variable is followed by
75726 *=word*, then the value of that variable shall be set to *word*.”. The reason for this change is that the
75727 SYNOPSIS for *readonly* includes:

75728 `readonly name[=word] . . .`

75729 but the meaning of the optional “=*word*” is never explained in the text.

75730 **NAME**

75731 return — return from a function or dot script |

75732 **SYNOPSIS**75733 return [*n*]75734 **DESCRIPTION**

75735 The *return* utility shall cause the shell to stop executing the current function or *dot* script. If the
75736 shell is not currently executing a function or *dot* script, the results are unspecified.

75737 **OPTIONS**

75738 None.

75739 **OPERANDS**

75740 See the DESCRIPTION.

75741 **STDIN**

75742 Not used.

75743 **INPUT FILES**

75744 None.

75745 **ENVIRONMENT VARIABLES**

75746 None.

75747 **ASYNCHRONOUS EVENTS**

75748 Default.

75749 **STDOUT**

75750 Not used.

75751 **STDERR**

75752 The standard error shall be used only for diagnostic messages.

75753 **OUTPUT FILES**

75754 None.

75755 **EXTENDED DESCRIPTION**

75756 None.

75757 **EXIT STATUS**

75758 The value of the special parameter '?' shall be set to *n*, an unsigned decimal integer, or to the
75759 exit status of the last command executed if *n* is not specified. If the value of *n* is greater than 255,
75760 the results are undefined. When *return* is executed in a *trap* action, the last command is
75761 considered to be the command that executed immediately preceding the *trap* action.

75762 **CONSEQUENCES OF ERRORS**

75763 Default.

75764 **APPLICATION USAGE**

75765 None.

75766 **EXAMPLES**

75767 None.

75768 **RATIONALE**

75769 The behavior of *return* when not in a function or *dot* script differs between the System V shell
75770 and the KornShell. In the System V shell this is an error, whereas in the KornShell, the effect is
75771 the same as *exit*.

75772 The results of returning a number greater than 255 are undefined because of differing practices

75773 in the various historical implementations. Some shells AND out all but the low-order 8 bits;
75774 others allow larger values, but not of unlimited size.

75775 See the discussion of appropriate exit status values under *exit*.

75776 **FUTURE DIRECTIONS**

75777 None.

75778 **SEE ALSO**

75779 [Section 2.9.5](#), [Section 2.14](#), *dot*

75780 **CHANGE HISTORY**

75781 **Issue 6**

75782 IEEE Std 1003.1-2001/Cor 1-2002, item XCU/TC1/D6/5 is applied so that the reference page
75783 sections use terms as described in the Utility Description Defaults ([Section 1.4](#)). No change in
75784 behavior is intended.

75785 **Issue 7**

75786 POSIX.1-2008, Technical Corrigendum 1, XCU/TC1-2008/0044 [214] and XCU/TC1-2008/0045 +
75787 [214] are applied.

NAME

set — set or unset options and positional parameters

SYNOPSIS

set [-abCefhmnvux] [-o *option*] [*argument...*]

set [+abCefhmnvux] [+o *option*] [*argument...*]

set -- [*argument...*]

set -o

set +o

DESCRIPTION

If no *options* or *arguments* are specified, *set* shall write the names and values of all shell variables in the collation sequence of the current locale. Each *name* shall start on a separate line, using the format:

```
"%s=%s\n", <name>, <value>
```

The *value* string shall be written with appropriate quoting; see the description of shell quoting in [Section 2.2](#). The output shall be suitable for reinput to the shell, setting or resetting, as far as possible, the variables that are currently set; read-only variables cannot be reset.

When options are specified, they shall set or unset attributes of the shell, as described below. When *arguments* are specified, they cause positional parameters to be set or unset, as described below. Setting or unsetting attributes and positional parameters are not necessarily related actions, but they can be combined in a single invocation of *set*.

The *set* special built-in shall support XBD [Section 12.2](#) except that options can be specified with either a leading <hyphen> (meaning enable the option) or <plus-sign> (meaning disable it) unless otherwise specified.

Implementations shall support the options in the following list in both their <hyphen> and <plus-sign> forms. These options can also be specified as options to *sh*.

–a When this option is on, the *export* attribute shall be set for each variable to which an assignment is performed; see XBD [Section 4.22](#). If the assignment precedes a utility name in a command, the *export* attribute shall not persist in the current execution environment after the utility completes, with the exception that preceding one of the special built-in utilities causes the *export* attribute to persist after the built-in has completed. If the assignment does not precede a utility name in the command, or if the assignment is a result of the operation of the *getopts* or *read* utilities, the *export* attribute shall persist until the variable is unset.

–b This option shall be supported if the implementation supports the User Portability Utilities option. It shall cause the shell to notify the user asynchronously of background job completions. The following message is written to standard error:

```
"[%d]%c %s\n", <job-number>, <current>, <status>, <job-name>
```

where the fields shall be as follows:

<current>	The character '+' identifies the job that would be used as a default for the <i>fg</i> or <i>bg</i> utilities; this job can also be specified using the <i>job_id</i> "%+" or "%%". The character '-' identifies the job that would become the default if the current default job were to exit; this job can also be specified using the <i>job_id</i> "%-". For other jobs, this field is a <space>. At most one job can be identified with '+' and at most one job can be identified with '-'. If there is any suspended job, then the current job shall be a suspended

75832 job. If there are at least two suspended jobs, then the previous job also
75833 shall be a suspended job.

75834 *<job-number>* A number that can be used to identify the process group to the *wait*, *fg*, *bg*,
75835 and *kill* utilities. Using these utilities, the job can be identified by prefixing
75836 the job number with ' % '.

75837 *<status>* Unspecified.

75838 *<job-name>* Unspecified.

75839 When the shell notifies the user a job has been completed, it may remove the job's process
75840 ID from the list of those known in the current shell execution environment; see [Section](#)
75841 [2.9.3.1](#). Asynchronous notification shall not be enabled by default.

75842 **-C** (Uppercase C.) Prevent existing files from being overwritten by the shell's ' > ' redirection
75843 operator (see [Section 2.7.2](#)); the " > | " redirection operator shall override this *noclobber*
75844 option for an individual file.

75845 **-e** When this option is on, when any command fails (for any of the reasons listed in [Section](#) |
75846 [2.8.1](#) or by returning an exit status greater than zero), the shell immediately shall exit with |
75847 the following exceptions: |

75848 1. The failure of any individual command in a multi-command pipeline shall not cause |
75849 the shell to exit. Only the failure of the pipeline itself shall be considered. |

75850 2. The **-e** setting shall be ignored when executing the compound list following the |
75851 **while**, **until**, **if**, or **elif** reserved word, a pipeline beginning with the **!** reserved word, |
75852 or any command of an AND-OR list other than the last. |

75853 3. If the exit status of a compound command other than a subshell command was the |
75854 result of a failure while **-e** was being ignored, then **-e** shall not apply to this |
75855 command. |

75856 This requirement applies to the shell environment and each subshell environment |
75857 separately. For example, in: |

75858 `set -e; (false; echo one) | cat; echo two` |

75859 the *false* command causes the subshell to exit without executing `echo one`; however, `echo` |
75860 `two` is executed because the exit status of the pipeline `(false; echo one) | cat` is |
75861 zero. |

75862 **-f** The shell shall disable pathname expansion.

75863 **-h** Locate and remember utilities invoked by functions as those functions are defined (the
75864 utilities are normally located when the function is executed).

75865 **-m** This option shall be supported if the implementation supports the User Portability Utilities
75866 option. All jobs shall be run in their own process groups. Immediately before the shell issues
75867 a prompt after completion of the background job, a message reporting the exit status of the
75868 background job shall be written to standard error. If a foreground job stops, the shell shall
75869 write a message to standard error to that effect, formatted as described by the *jobs* utility. In
75870 addition, if a job changes status other than exiting (for example, if it stops for input or
75871 output or is stopped by a SIGSTOP signal), the shell shall write a similar message
75872 immediately prior to writing the next prompt. This option is enabled by default for
75873 interactive shells.

- 75874 **-n** The shell shall read commands but does not execute them; this can be used to check for
 75875 shell script syntax errors. An interactive shell may ignore this option.
- 75876 **-o** Write the current settings of the options to standard output in an unspecified format.
- 75877 **+o** Write the current option settings to standard output in a format that is suitable for reinput
 75878 to the shell as commands that achieve the same options settings.
- 75879 **-o option**
 75880 This option is supported if the system supports the User Portability Utilities option. It shall
 75881 set various options, many of which shall be equivalent to the single option letters. The
 75882 following values of *option* shall be supported:
- 75883 *allexport* Equivalent to **-a**.
- 75884 *errexit* Equivalent to **-e**.
- 75885 *ignoreeof* Prevent an interactive shell from exiting on end-of-file. This setting prevents
 75886 accidental logouts when <control>-D is entered. A user shall explicitly *exit* to
 75887 leave the interactive shell.
- 75888 *monitor* Equivalent to **-m**. This option is supported if the system supports the User
 75889 Portability Utilities option.
- 75890 *noclobber* Equivalent to **-C** (uppercase C).
- 75891 *noglob* Equivalent to **-f**.
- 75892 *noexec* Equivalent to **-n**.
- 75893 *nolog* Prevent the entry of function definitions into the command history; see
 75894 [Command History List](#).
- 75895 *notify* Equivalent to **-b**.
- 75896 *nounset* Equivalent to **-u**.
- 75897 *verbose* Equivalent to **-v**.
- 75898 *vi* Allow shell command line editing using the built-in *vi* editor. Enabling *vi*
 75899 mode shall disable any other command line editing mode provided as an
 75900 implementation extension.
- 75901 It need not be possible to set *vi* mode on for certain block-mode terminals.
- 75902 *xtrace* Equivalent to **-x**.
- 75903 **-u** When the shell tries to expand an unset parameter other than the '@' and '*' special
 75904 parameters, it shall write a message to standard error and shall not execute the command
 75905 containing the expansion, but for the purposes of setting the '?' special parameter and the
 75906 exit status of the shell the command shall be treated as having been executed and returned
 75907 an exit status of between 1 and 125 inclusive. A non-interactive shell shall immediately exit.
 75908 An interactive shell shall not exit.
- 75909 **-v** The shell shall write its input to standard error as it is read.
- 75910 **-x** The shell shall write to standard error a trace for each command after it expands the
 75911 command and before it executes it. It is unspecified whether the command that turns
 75912 tracing off is traced.
- 75913 The default for all these options shall be off (unset) unless stated otherwise in the description of
 75914 the option or unless the shell was invoked with them on; see *sh*.

75915 The remaining arguments shall be assigned in order to the positional parameters. The special
 75916 parameter '`#`' shall be set to reflect the number of positional parameters. All positional
 75917 parameters shall be unset before any new values are assigned.

75918 If the first argument is '`-`', the results are unspecified.

75919 The special argument "`--`" immediately following the *set* command name can be used to
 75920 delimit the arguments if the first argument begins with '`+`' or '`-`', or to prevent inadvertent
 75921 listing of all shell variables when there are no arguments. The command *set --* without *argument*
 75922 shall unset all positional parameters and set the special parameter '`#`' to zero.

75923 OPTIONS

75924 See the DESCRIPTION.

75925 OPERANDS

75926 See the DESCRIPTION.

75927 STDIN

75928 Not used.

75929 INPUT FILES

75930 None.

75931 ENVIRONMENT VARIABLES

75932 None.

75933 ASYNCHRONOUS EVENTS

75934 Default.

75935 STDOUT

75936 See the DESCRIPTION.

75937 STDERR

75938 The standard error shall be used only for diagnostic messages.

75939 OUTPUT FILES

75940 None.

75941 EXTENDED DESCRIPTION

75942 None.

75943 EXIT STATUS

75944 Zero.

75945 CONSEQUENCES OF ERRORS

75946 Default.

75947 APPLICATION USAGE

75948 Application writers should avoid relying on *set -e* within functions. For example, in the
 75949 following script:

```
75950 set -e
75951 start() {
75952     some_server
75953     echo some_server started successfully
75954 }
75955 start || echo >&2 some_server failed
```

75956 the *-e* setting is ignored within the function body (because the function is a command in an
 75957 AND-OR list other than the last). Therefore, if `some_server` fails, the function carries on to

echo "some_server started successfully", and the exit status of the function is zero (which means "some_server failed" is not output).

EXAMPLES

Write out all variables and their values:

```
set
```

Set \$1, \$2, and \$3 and set "\$#" to 3:

```
set c a b
```

Turn on the `-x` and `-v` options:

```
set -xv
```

Unset all positional parameters:

```
set --
```

Set \$1 to the value of *x*, even if it begins with `'-'` or `'+'`:

```
set -- "$x"
```

Set the positional parameters to the expansion of *x*, even if *x* expands with a leading `'-'` or `'+'`:

```
set -- $x
```

RATIONALE

The `set --` form is listed specifically in the SYNOPSIS even though this usage is implied by the Utility Syntax Guidelines. The explanation of this feature removes any ambiguity about whether the `set --` form might be misinterpreted as being equivalent to `set` without any options or arguments. The functionality of this form has been adopted from the KornShell. In System V, `set --` only unsets parameters if there is at least one argument; the only way to unset all parameters is to use *shift*. Using the KornShell version should not affect System V scripts because there should be no reason to issue it without arguments deliberately; if it were issued as, for example:

```
set -- "$@"
```

and there were in fact no arguments resulting from `"$@"`, unsetting the parameters would have no result.

The `set +` form in early proposals was omitted as being an unnecessary duplication of `set` alone and not widespread historical practice.

The *noclobber* option was changed to allow `set -C` as well as the `set -o noclobber` option. The single-letter version was added so that the historical `"$-"` paradigm would not be broken; see [Section 2.5.2](#).

The description of the `-e` option is intended to match the behavior of the 1988 version of the KornShell.

The `-h` flag is related to command name hashing. See [hash](#).

The following `set` flags were omitted intentionally with the following rationale:

-k The `-k` flag was originally added by the author of the Bourne shell to make it easier for users of pre-release versions of the shell. In early versions of the Bourne shell the construct `set name=value` had to be used to assign values to shell variables. The problem with `-k` is that the behavior affects parsing, virtually precluding writing any compilers. To explain the behavior of `-k`, it is necessary to describe the parsing algorithm, which is implementation-defined. For example:

75999 set -k; echo name=value

76000 and:

76001 set -k
76002 echo name=value

76003 behave differently. The interaction with functions is even more complex. What is more, the
76004 **-k** flag is never needed, since the command line could have been reordered.

76005 **-t** The **-t** flag is hard to specify and almost never used. The only known use could be done
76006 with here-documents. Moreover, the behavior with *ksh* and *sh* differs. The reference page
76007 says that it exits after reading and executing one command. What is one command? If the
76008 input is *date;date*, *sh* executes both *date* commands while *ksh* does only the first.

76009 Consideration was given to rewriting *set* to simplify its confusing syntax. A specific suggestion
76010 was that the *unset* utility should be used to unset options instead of using the non-*getopt()*-able
76011 +*option* syntax. However, the conclusion was reached that the historical practice of using +*option*
76012 was satisfactory and that there was no compelling reason to modify such widespread historical
76013 practice.

76014 The **-o** option was adopted from the KornShell to address user needs. In addition to its
76015 generally friendly interface, **-o** is needed to provide the *vi* command line editing mode, for
76016 which historical practice yields no single-letter option name. (Although it might have been
76017 possible to invent such a letter, it was recognized that other editing modes would be developed
76018 and **-o** provides ample name space for describing such extensions.)

76019 Historical implementations are inconsistent in the format used for **-o** option status reporting.
76020 The **+o** format without an option-argument was added to allow portable access to the options
76021 that can be saved and then later restored using, for instance, a dot script.

76022 Historically, *sh* did trace the command *set +x*, but *ksh* did not.

76023 The *ignoreeof* setting prevents accidental logouts when the end-of-file character (typically
76024 <control>-D) is entered. A user shall explicitly *exit* to leave the interactive shell.

76025 The *set -m* option was added to apply only to the UPE because it applies primarily to interactive
76026 use, not shell script applications.

76027 The ability to do asynchronous notification became available in the 1988 version of the
76028 KornShell. To have it occur, the user had to issue the command:

76029 trap "jobs -n" CLD

76030 The C shell provides two different levels of an asynchronous notification capability. The
76031 environment variable *notify* is analogous to what is done in *set -b* or *set -o notify*. When set, it
76032 notifies the user immediately of background job completions. When unset, this capability is
76033 turned off.

76034 The other notification ability comes through the built-in utility *notify*. The syntax is:

76035 notify [%job ...]

76036 By issuing *notify* with no operands, it causes the C shell to notify the user asynchronously when
76037 the state of the current job changes. If given operands, *notify* asynchronously informs the user of
76038 changes in the states of the specified jobs.

76039 To add asynchronous notification to the POSIX shell, neither the KornShell extensions to *trap*,
76040 nor the C shell *notify* environment variable seemed appropriate (*notify* is not a proper POSIX
76041 environment variable name).

76042	The <i>set</i> -b option was selected as a compromise.	
76043	The <i>notify</i> built-in was considered to have more functionality than was required for simple asynchronous notification.	
76044		
76045	Historically, some shells applied the -u option to all parameters including <code>\$@</code> and <code>\$*</code> . The standard developers felt that this was a misfeature since it is normal and common for <code>\$@</code> and <code>\$*</code> to be used in shell scripts regardless of whether they were passed any arguments. Treating these uses as an error when no arguments are passed reduces the value of -u for its intended purpose of finding spelling mistakes in variable names and uses of unset positional parameters.	+
76046		+
76047		+
76048		+
76049		+
76050	FUTURE DIRECTIONS	
76051	None.	
76052	SEE ALSO	
76053	Section 2.14, <i>hash</i>	
76054	XBD Section 4.22, Section 12.2	
76055	CHANGE HISTORY	
76056	Issue 6	
76057	The obsolescent <i>set</i> command name followed by <code>'-'</code> has been removed.	
76058	The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:	
76059		
76060	The <i>nolog</i> option is added to <i>set</i> -o.	
76061	IEEE PASC Interpretation 1003.2 #167 is applied, clarifying that the options default also takes into account the description of the option.	
76062		
76063	IEEE Std 1003.1-2001/Cor 1-2002, item XCU/TC1/D6/5 is applied so that the reference page sections use terms as described in the Utility Description Defaults (Section 1.4). No change in behavior is intended.	
76064		
76065		
76066	IEEE Std 1003.1-2001/Cor 1-2002, item XCU/TC1/D6/8 is applied, changing the square brackets in the example in RATIONALE to be in bold, which is the typeface used for optional items.	
76067		
76068		
76069	Issue 7	
76070	Austin Group Interpretation 1003.1-2001 #027 is applied, clarifying the behavior if the first argument is <code>'-'</code> .	
76071		
76072	SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.	
76073	XSI shading is removed from the -h functionality.	
76074	POSIX.1-2008, Technical Corrigendum 1, XCU/TC1-2008/0046 [52], XCU/TC1-2008/0047 [155,280], XCU/TC1-2008/0048 [52], XCU/TC1-2008/0049 [52], and XCU/TC1-2008/0050 [155,430] are applied.	+
76075		+
76076		+

76077 **NAME**

76078 shift — shift positional parameters

76079 **SYNOPSIS**76080 shift [*n*]76081 **DESCRIPTION**

76082 The positional parameters shall be shifted. Positional parameter 1 shall be assigned the value of
76083 parameter (1+*n*), parameter 2 shall be assigned the value of parameter (2+*n*), and so on. The
76084 parameters represented by the numbers "\$#" down to "\$#-*n*+1" shall be unset, and the
76085 parameter '#' is updated to reflect the new number of positional parameters.

76086 The value *n* shall be an unsigned decimal integer less than or equal to the value of the special
76087 parameter '#'. If *n* is not given, it shall be assumed to be 1. If *n* is 0, the positional and special
76088 parameters are not changed.

76089 **OPTIONS**

76090 None.

76091 **OPERANDS**

76092 See the DESCRIPTION.

76093 **STDIN**

76094 Not used.

76095 **INPUT FILES**

76096 None.

76097 **ENVIRONMENT VARIABLES**

76098 None.

76099 **ASYNCHRONOUS EVENTS**

76100 Default.

76101 **STDOUT**

76102 Not used.

76103 **STDERR**

76104 The standard error shall be used only for diagnostic messages.

76105 **OUTPUT FILES**

76106 None.

76107 **EXTENDED DESCRIPTION**

76108 None.

76109 **EXIT STATUS**

76110 If the *n* operand is invalid or is greater than "\$#", this may be considered a syntax error and a |
76111 non-interactive shell may exit; if the shell does not exit in this case, a non-zero exit status shall be |
76112 returned. Otherwise, zero shall be returned.

76113 **CONSEQUENCES OF ERRORS**

76114 Default.

76115 **APPLICATION USAGE**

76116 None.

76117 **EXAMPLES**76118 **\$** set a b c d e76119 **\$** shift 276120 **\$** echo \$*76121 **c d e**76122 **RATIONALE**

76123 None.

76124 **FUTURE DIRECTIONS**

76125 None.

76126 **SEE ALSO**76127 [Section 2.14](#)76128 **CHANGE HISTORY**76129 **Issue 6**

76130 IEEE Std 1003.1-2001/Cor 1-2002, item XCU/TC1/D6/5 is applied so that the reference page
76131 sections use terms as described in the Utility Description Defaults ([Section 1.4](#)). No change in
76132 behavior is intended.

76133 **Issue 7**

76134 POSIX.1-2008, Technical Corrigendum 1, XCU/TC1-2008/0051 [459] is applied.

+

76135 **NAME**

76136 times — write process times

76137 **SYNOPSIS**

76138 times

76139 **DESCRIPTION**76140 The *times* utility shall write the accumulated user and system times for the shell and for all of its
76141 child processes, in the following POSIX locale format:

```
76142       "%dm%fs %dm%fs\n%dm%fs %dm%fs\n", <shell user minutes>,  
76143       <shell user seconds>, <shell system minutes>,  
76144       <shell system seconds>, <children user minutes>,  
76145       <children user seconds>, <children system minutes>,  
76146       <children system seconds>
```

76147 The four pairs of times shall correspond to the members of the **<sys/times.h> tms** structure
76148 (defined in XBD [Chapter 13](#)) as returned by *times()*: *tms_utime*, *tms_stime*, *tms_cutime*, and
76149 *tms_cstime*, respectively.76150 **OPTIONS**

76151 None.

76152 **OPERANDS**

76153 None.

76154 **STDIN**

76155 Not used.

76156 **INPUT FILES**

76157 None.

76158 **ENVIRONMENT VARIABLES**

76159 None.

76160 **ASYNCHRONOUS EVENTS**

76161 Default.

76162 **STDOUT**

76163 See the DESCRIPTION.

76164 **STDERR**

76165 The standard error shall be used only for diagnostic messages.

76166 **OUTPUT FILES**

76167 None.

76168 **EXTENDED DESCRIPTION**

76169 None.

76170 **EXIT STATUS**

76171 Zero.

76172 **CONSEQUENCES OF ERRORS**

76173 Default.

76174 **APPLICATION USAGE**

76175 None.

76176 **EXAMPLES**76177 **\$ times**76178 **0m0.43s 0m1.11s**76179 **8m44.18s 1m43.23s**76180 **RATIONALE**76181 The *times* special built-in from the Single UNIX Specification is now required for all conforming
76182 shells.76183 **FUTURE DIRECTIONS**

76184 None.

76185 **SEE ALSO**76186 [Section 2.14](#)76187 XBD [<sys/times.h>](#)76188 **CHANGE HISTORY**76189 **Issue 6**76190 IEEE Std 1003.1-2001/Cor 1-2002, item XCU/TC1/D6/9 is applied, changing text in the
76191 DESCRIPTION from: "Write the accumulated user and system times for the shell and for all of
76192 its child processes ..." to: "The *times* utility shall write the accumulated user and system times for
76193 the shell and for all of its child processes ...".

NAME

trap — trap signals

SYNOPSIS

```
trap n [condition...]
trap [action condition...]
```

DESCRIPTION

If the first operand is an unsigned decimal integer, the shell shall treat all operands as conditions, and shall reset each condition to the default value. Otherwise, if there are operands, the first is treated as an action and the remaining as conditions.

If *action* is `'-'`, the shell shall reset each *condition* to the default value. If *action* is null (`" "`), the shell shall ignore each specified *condition* if it arises. Otherwise, the argument *action* shall be read and executed by the shell when one of the corresponding conditions arises. The action of *trap* shall override a previous action (either default action or one explicitly set). The value of `"$?"` after the *trap* action completes shall be the value it had before *trap* was invoked.

The condition can be EXIT, 0 (equivalent to EXIT), or a signal specified using a symbolic name, without the SIG prefix, as listed in the tables of signal names in the `<signal.h>` header defined in XBD [Chapter 13](#); for example, HUP, INT, QUIT, TERM. Implementations may permit names with the SIG prefix or ignore case in signal names as an extension. Setting a trap for SIGKILL or SIGSTOP produces undefined results.

The environment in which the shell executes a *trap* on EXIT shall be identical to the environment immediately after the last command executed before the *trap* on EXIT was taken.

Each time *trap* is invoked, the *action* argument shall be processed in a manner equivalent to:

```
eval action
```

Signals that were ignored on entry to a non-interactive shell cannot be trapped or reset, although no error need be reported when attempting to do so. An interactive shell may reset or catch signals ignored on entry. Traps shall remain in place for a given shell until explicitly changed with another *trap* command.

When a subshell is entered, traps that are not being ignored shall be set to the default actions, except in the case of a command substitution containing only a single *trap* command, when the traps need not be altered. Implementations may check for this case using only lexical analysis; for example, if `'trap'` and `$( trap -- )` do not alter the traps in the subshell, cases such as assigning `var=trap` and then using `$( $var )` may still alter them. This does not imply that the *trap* command cannot be used within the subshell to set new traps.

The *trap* command with no operands shall write to standard output a list of commands associated with each condition. If the command is executed in a subshell, the implementation does not perform the optional check described above for a command substitution containing only a single *trap* command, and no *trap* commands with operands have been executed since entry to the subshell, the list shall contain the commands that were associated with each condition immediately before the subshell environment was entered. Otherwise, the list shall contain the commands currently associated with each condition. The format shall be:

```
"trap -- %s %s ...\n", <action>, <condition> ...
```

The shell shall format the output, including the proper use of quoting, so that it is suitable for reinput to the shell as commands that achieve the same trapping results. For example:

```
save_traps=$(trap)
...
eval "$save_traps"
```

76240 XSI XSI-conformant systems also allow numeric signal numbers for the conditions corresponding to
 76241 the following signal names:

76242 1 SIGHUP
 76243 2 SIGINT
 76244 3 SIGQUIT
 76245 6 SIGABRT
 76246 9 SIGKILL
 76247 14 SIGALRM
 76248 15 SIGTERM

76249 The *trap* special built-in shall conform to XBD [Section 12.2](#).

76250 OPTIONS

76251 None.

76252 OPERANDS

76253 See the DESCRIPTION.

76254 STDIN

76255 Not used.

76256 INPUT FILES

76257 None.

76258 ENVIRONMENT VARIABLES

76259 None.

76260 ASYNCHRONOUS EVENTS

76261 Default.

76262 STDOUT

76263 See the DESCRIPTION.

76264 STDERR

76265 The standard error shall be used only for diagnostic messages.

76266 OUTPUT FILES

76267 None.

76268 EXTENDED DESCRIPTION

76269 None.

76270 EXIT STATUS

76271 XSI If the trap name **or number** is invalid, a non-zero exit status shall be returned; otherwise, zero
 76272 XSI shall be returned. For both interactive and non-interactive shells, invalid signal names **or**
 76273 **numbers** shall not be considered a syntax error and do not cause the shell to abort.

76274 CONSEQUENCES OF ERRORS

76275 Default.

APPLICATION USAGE

None.

EXAMPLES

Write out a list of all traps and actions:

```
trap
```

Set a trap so the *logout* utility in the directory referred to by the *HOME* environment variable executes when the shell terminates:

```
trap '"$HOME"/logout' EXIT
```

or:

```
trap '"$HOME"/logout' 0
```

Unset traps on INT, QUIT, TERM, and EXIT:

```
trap - INT QUIT TERM EXIT
```

RATIONALE

Implementations may permit lowercase signal names as an extension. Implementations may also accept the names with the SIG prefix; no known historical shell does so. The *trap* and *kill* utilities in this volume of POSIX.1-2008 are now consistent in their omission of the SIG prefix for signal names. Some *kill* implementations do not allow the prefix, and *kill -l* lists the signals without prefixes.

Trapping SIGKILL or SIGSTOP is syntactically accepted by some historical implementations, but it has no effect. Portable POSIX applications cannot attempt to trap these signals.

The output format is not historical practice. Since the output of historical *trap* commands is not portable (because numeric signal values are not portable) and had to change to become so, an opportunity was taken to format the output in a way that a shell script could use to save and then later reuse a trap if it wanted.

The KornShell uses an **ERR** trap that is triggered whenever *set -e* would cause an exit. This is allowable as an extension, but was not mandated, as other shells have not used it.

The text about the environment for the EXIT trap invalidates the behavior of some historical versions of interactive shells which, for example, close the standard input before executing a trap on 0. For example, in some historical interactive shell sessions the following trap on 0 would always print "--":

```
trap 'read foo; echo "--$foo--" 0
```

The command:

```
trap 'eval " $cmd"' 0
```

causes the contents of the shell variable *cmd* to be executed as a command when the shell exits. Using:

```
trap '$cmd' 0
```

does not work correctly if *cmd* contains any special characters such as quoting or redirections. Using:

```
trap " $cmd" 0
```

also works (the leading <space> character protects against unlikely cases where *cmd* is a decimal integer or begins with '-'), but it expands the *cmd* variable when the *trap* command is executed,

76317 not when the exit action is executed.

76318 **FUTURE DIRECTIONS**

76319 None.

76320 **SEE ALSO**

76321 [Section 2.14](#)

76322 XBD [Section 12.2, <signal.h>](#)

76323 **CHANGE HISTORY**

76324 **Issue 6**

76325 XSI-conforming implementations provide the mapping of signal names to numbers given above
76326 (previously this had been marked obsolescent). Other implementations need not provide this
76327 optional mapping.

76328 IEEE Std 1003.1-2001/Cor 1-2002, item XCU/TC1/D6/5 is applied so that the reference page
76329 sections use terms as described in the Utility Description Defaults ([Section 1.4](#)). No change in
76330 behavior is intended.

76331 **Issue 7**

76332 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

76333 Austin Group Interpretation 1003.1-2001 #116 is applied.

76334 POSIX.1-2008, Technical Corrigendum 1, XCU/TC1-2008/0052 [53,268,440], +
76335 XCU/TC1-2008/0053 [53,268,440], XCU/TC1-2008/0054 [163], XCU/TC1-2008/0055 [163], and +
76336 XCU/TC1-2008/0056 [163] are applied.

76337 **NAME**

76338 unset — unset values and attributes of variables and functions

76339 **SYNOPSIS**76340 unset [-fv] *name*...76341 **DESCRIPTION**76342 Each variable or function specified by *name* shall be unset.76343 If **-v** is specified, *name* refers to a variable name and the shell shall unset it and remove it from
76344 the environment. Read-only variables cannot be unset.76345 If **-f** is specified, *name* refers to a function and the shell shall unset the function definition.76346 If neither **-f** nor **-v** is specified, *name* refers to a variable; if a variable by that name does not
76347 exist, it is unspecified whether a function by that name, if any, shall be unset.76348 Unsetting a variable or function that was not previously set shall not be considered an error and
76349 does not cause the shell to abort.76350 The *unset* special built-in shall support XBD [Section 12.2](#).

76351 Note that:

76352 VARIABLE=

76353 is not equivalent to an *unset* of **VARIABLE**; in the example, **VARIABLE** is set to " ". Also, the
76354 variables that can be *unset* should not be misinterpreted to include the special parameters (see
76355 [Section 2.5.2](#)).76356 **OPTIONS**

76357 See the DESCRIPTION.

76358 **OPERANDS**

76359 See the DESCRIPTION.

76360 **STDIN**

76361 Not used.

76362 **INPUT FILES**

76363 None.

76364 **ENVIRONMENT VARIABLES**

76365 None.

76366 **ASYNCHRONOUS EVENTS**

76367 Default.

76368 **STDOUT**

76369 Not used.

76370 **STDERR**

76371 The standard error shall be used only for diagnostic messages.

76372 **OUTPUT FILES**

76373 None.

76374 **EXTENDED DESCRIPTION**

76375 None.

76376 **EXIT STATUS**76377 0 All *name* operands were successfully unset.76378 >0 At least one *name* could not be unset.76379 **CONSEQUENCES OF ERRORS**

76380 Default.

76381 **APPLICATION USAGE**

76382 None.

76383 **EXAMPLES**76384 Unset *VISUAL* variable:

76385 unset -v VISUAL

76386 Unset the functions **foo** and **bar**:

76387 unset -f foo bar

76388 **RATIONALE**76389 Consideration was given to omitting the **-f** option in favor of an *unfunction* utility, but the
76390 standard developers decided to retain historical practice.76391 The **-v** option was introduced because System V historically used one name space for both
76392 variables and functions. When *unset* is used without options, System V historically unset either a
76393 function or a variable, and there was no confusion about which one was intended. A portable
76394 POSIX application can use *unset* without an option to unset a variable, but not a function; the **-f**
76395 option must be used.76396 **FUTURE DIRECTIONS**

76397 None.

76398 **SEE ALSO**76399 [Section 2.14](#)76400 XBD [Section 12.2](#)76401 **CHANGE HISTORY**76402 **Issue 6**76403 IEEE Std 1003.1-2001/Cor 1-2002, item XCU/TC1/D6/5 is applied so that the reference page
76404 sections use terms as described in the Utility Description Defaults ([Section 1.4](#)). No change in
76405 behavior is intended.76406 **Issue 7**

76407 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

76408
76409

Chapter 3
Batch Environment Services

76410 OB BE
76411
76412
76413
76414
76415

This chapter describes the services and utilities that shall be implemented on all systems that claim conformance to the Batch Environment Services and Utilities option. The functionality described in this section shall be provided on implementations that support the Batch Environment Services and Utilities option (and the rest of this section is not further shaded for this option).
Note that the Batch Environment Services and Utilities option is marked obsolescent in Issue 7.

76416 **3.1 General Concepts**

76417 **3.1.1 Batch Client-Server Interaction**

76418 Batch jobs are created and managed by batch servers. A batch client interacts with a batch server
76419 to access batch services on behalf of the user. In order to use batch services, a user must have
76420 access to a batch client.

76421 A batch server is a computational entity, such as a daemon process, that provides batch services.
76422 Batch servers route, queue, modify, and execute batch jobs on behalf of batch clients.

76423 The batch utilities described in this volume of POSIX.1-2008 (and listed in [Table 3-1](#)) are clients
76424 of batch services; they allow users to perform actions on the job such as creating, modifying, and
76425 deleting batch jobs from a shell command line. Although these batch utilities may be said to
76426 accomplish certain services, they actually obtain services on behalf of a user by means of
76427 requests to batch servers.

76428 **Table 3-1** Batch Utilities

76429	<i>qalter</i>	<i>qmove</i>	<i>qrls</i>	<i>qstat</i>
76430	<i>qdel</i>	<i>qmsg</i>	<i>qselect</i>	<i>qsub</i>
76431	<i>qhold</i>	<i>qrerun</i>	<i>qsig</i>	

76432 Client-server interaction takes place by means of the batch requests defined in this chapter.
76433 Because direct access to batch jobs and queues is limited to batch servers, clients and servers of
76434 different implementations can interoperate, since dependencies on private structures for batch
76435 jobs and queues are limited to batch servers. Also, batch servers may be clients of other batch
76436 servers.

3.1.2 Batch Queues

Two types of batch queue are described: routing queues and execution queues. When a batch job is placed in a routing queue, it is a candidate for routing. A batch job is removed from routing queues under the following conditions:

- The batch job has been routed to another queue.
- The batch job has been deleted from the batch queue.
- The batch job has been aborted.

When a batch job is placed in an execution queue, it is a candidate for execution.

A batch job is removed from an execution queue under the following conditions:

- The batch job has been executed and exited.
- The batch job has been aborted.
- The batch job has been deleted from the batch queue.
- The batch job has been moved to another queue.

Access to a batch queue is limited to the batch server that manages the batch queue. Clients never access a batch queue or a batch job directly, either to read or write information; all client access to batch queues or jobs takes place through batch servers.

3.1.3 Batch Job Creation

When a batch server creates a batch job on behalf of a client, it shall assign a batch job identifier to the job. A batch job identifier consists of both a sequence number that is unique among the sequence numbers issued by that server and the name of the server. Since the batch server name is unique within a name space, the job identifier is likewise unique within the name space.

The batch server that creates a batch job shall return the batch server-assigned job identifier to the client that requested the job creation. If the batch server routes or moves the job to another server, it sends the job identifier with the job. Once assigned, the job identifier of a batch job shall never change.

3.1.4 Batch Job Tracking

Since a batch job may be moved after creation, the batch server name component of the job identifier need not indicate the location of the job. An implementation may provide a batch job tracking mechanism, in which case the user generally does not need to know the location of the job. However, an implementation need not provide a batch job tracking mechanism, in which case the user must find routed jobs by probing the possible destinations.

3.1.5 Batch Job Routing

To route a batch job, a batch server either moves the job to some other queue that is managed by the batch server, or requests that some other batch server accept the job.

Each routing queue has one or more queues to which it can route batch jobs. The batch server administrator creates routing queues.

A batch server may route a batch job from a routing queue to another routing queue. Batch servers shall prevent or otherwise handle cases of circular routing paths. As a deferred service, a batch server routes jobs from the routing queues that it manages. The algorithm by which a batch server selects a batch queue to which to route a batch job is implementation-defined.

A batch job need not be eligible for routing to all the batch queues fed by the routing queue from which it is routed. A batch server that has been asked to accept the job may reject the request if the job requires resources that are unavailable to that batch server, or if the client is not authorized to access the batch server.

Batch servers may route high-priority jobs before low-priority jobs, but, on other than overloaded systems, the effect may be imperceptible to the user. If all the batch servers fed by a routing queue reject requests to accept the job for reasons that are permanent, the batch server that manages the job shall abort the job. If all or some rejections are temporary, the batch server should try to route the job again at some later point.

The reasons for rejecting a batch job are implementation-defined.

The reasons for which the routing should be retried later and the reasons for which the job should be aborted are also implementation-defined.

3.1.6 Batch Job Execution

To execute a batch job is to create a session leader (a process) that runs the shell program indicated by the *Shell_Path* attribute of the job. The script shall be passed to the program as its standard input. An implementation may pass the script to the program by other implementation-defined means. At the time a batch job begins execution, it is defined to enter the RUNNING state. The primary program that is executed by a batch job is typically, though not necessarily, a shell program.

A batch server shall execute eligible jobs as a deferred service—no client request is necessary once the batch job is created and eligible. However, the attributes of a batch job, such as the job hold type, may render the job ineligible. A batch server shall scan the execution queues that it manages for jobs that are eligible for execution. The algorithm by which the batch server selects eligible jobs for execution is implementation-defined.

As part of creating the process for the batch job, the batch server shall open the standard output and standard error streams of the session.

The attributes of a batch job may indicate that the batch server executing the job shall send mail to a list of users at the time it begins execution of the job.

76505 3.1.7 Batch Job Exit

76506 When the session leader of an executing job terminates, the job exits. As part of exiting a batch
76507 job, the batch server that manages the job shall remove the job from the batch queue in which it
76508 resides. The server shall transfer output files of the job to a location described by the attributes of
76509 the job.

76510 The attributes of a batch job may indicate that the batch server managing the job shall send mail
76511 to a list of users at the time the job exits.

76512 3.1.8 Batch Job Abort

76513 A batch server shall abort jobs for which a required deferred service cannot be performed. The
76514 attributes of a batch job may indicate that the batch server that aborts the job shall send mail to a
76515 list of users at the time it aborts the job.

76516 3.1.9 Batch Authorization

76517 Clients, such as the batch environment utilities (marked BE), access batch services by means of
76518 requests to one or more batch servers. To acquire the services of any given batch server, the user
76519 identifier under which the client runs must be authorized to use that batch server.

76520 The user with an associated user name that creates a batch job shall own the job and can perform
76521 actions such as read, modify, delete, and move.

76522 A user identifier of the same value at a different host need not be the same user. For example,
76523 user name *smith* at host **alpha** may or may not represent the same person as user name *smith* at
76524 host **beta**. Likewise, the same person may have access to different user names on different hosts.

76525 An implementation may optionally provide an authorization mechanism that permits one user
76526 name to access jobs under another user name.

76527 A process on a client host may be authorized to run processes under multiple user names at a
76528 batch server host. Where appropriate, the utilities defined in this volume of POSIX.1-2008
76529 provide a means for a user to choose from among such user names when creating or modifying
76530 a batch job.

76531 3.1.10 Batch Administration

76532 The processing of a batch job by a batch server is affected by the attributes of the job. The
76533 processing of a batch job may also be affected by the attributes of the batch queue in which the
76534 job resides and by the status of the batch server that manages the job. See also XBD [Chapter 3](#)
76535 (on page 33) for batch definitions.

3.1.11 Batch Notification

Whereas batch servers are persistent entities, clients are often transient. For example, the *qsub* utility creates a batch job and exits. For this reason, batch servers notify users of batch job events by sending mail to the user that owns the job, or to other designated users.

3.2 Batch Services

The presence of Batch Environment Services and Utilities option services is indicated by the configuration variable `POSIX2_PBS`. A conforming batch server provides services as defined in this section.

A batch server shall provide batch services in two ways:

1. The batch server provides a service at the request of a client.
2. The batch server provides a deferred service as a result of a change in conditions monitored by the batch server.

If a batch server cannot complete a request, it shall reject the request. If a batch server cannot complete a deferred service for a batch job, the batch server shall abort the batch job. [Table 3-2](#) is a summary of environment variables that shall be supported by an implementation of the batch server and utilities.

Table 3-2 Environment Variable Summary

Variable	Description
<i>PBS_DPREFIX</i>	Defines the directive prefix (see <i>qsub</i>)
<i>PBS_ENVIRONMENT</i>	Batch Job is batch or interactive (see Section 3.2.2.1)
<i>PBS_JOBID</i>	The <i>job_identifier</i> attribute of job (see Section 3.2.3.8)
<i>PBS_JOBNAME</i>	The <i>job_name</i> attribute of job (see Section 3.2.3.8)
<i>PBS_O_HOME</i>	Defines the <i>HOME</i> of the batch client (see <i>qsub</i>)
<i>PBS_O_HOST</i>	Defines the host name of the batch client (see <i>qsub</i>)
<i>PBS_O_LANG</i>	Defines the <i>LANG</i> of the batch client (see <i>qsub</i>)
<i>PBS_O_LOGNAME</i>	Defines the <i>LOGNAME</i> of the batch client (see <i>qsub</i>)
<i>PBS_O_MAIL</i>	Defines the <i>MAIL</i> of the batch client (see <i>qsub</i>)
<i>PBS_O_PATH</i>	Defines the <i>PATH</i> of the batch client (see <i>qsub</i>)
<i>PBS_O_QUEUE</i>	Defines the submit queue of the batch client (see <i>qsub</i>)
<i>PBS_O_SHELL</i>	Defines the <i>SHELL</i> of the batch client (see <i>qsub</i>)
<i>PBS_O_TZ</i>	Defines the <i>TZ</i> of the batch client (see <i>qsub</i>)
<i>PBS_O_WORKDIR</i>	Defines the working directory of the batch client (see <i>qsub</i>)
<i>PBS_QUEUE</i>	Defines the initial execution queue (see Section 3.2.2.1)

3.2.1 Batch Job States

A batch job shall always be in one of the following states: QUEUED, RUNNING, HELD, WAITING, EXITING, or TRANSITING. The state of a batch job determines the types of requests that the batch server that manages the batch job can accept for the batch job. A batch server shall change the state of a batch job either in response to service requests from clients or as a result of deferred services, such as job execution or job routing.

A batch job that is in the QUEUED state resides in a queue but is still pending either execution or routing, depending on the queue type.

A batch server that queues a batch job in a routing queue shall put the batch job in the QUEUED state. A batch server that puts a batch job in an execution queue, but has not yet executed the batch job, shall put the batch job in the QUEUED state. A batch job that resides in an execution queue and is executing is defined to be in the RUNNING state. While a batch job is in the RUNNING state, a session leader is associated with the batch job.

A batch job that resides in an execution queue, but is ineligible to run because of a hold attribute, is defined to be in the HELD state.

A batch job that is not held, but must wait until a future date and time before executing, is defined to be in the WAITING state.

When the session leader associated with a running job exits, the batch job shall be placed in the EXITING state.

A batch job for which the session leader has terminated is defined to be in the EXITING state, and the batch server that manages such a batch job cannot accept job modification requests that affect the batch job. While a batch job is in the EXITING state, the batch server that manages the batch job is staging output files and notifying clients of job completion. Once a batch job has exited, it no longer exists as an object managed by a batch server.

A batch job that is being moved from a routing queue to another queue is defined to be in the TRANSITING state.

When a batch job in a routing queue has been selected to be moved to a new destination, then the batch job shall be in either the QUEUED state or the TRANSITING state, depending on the batch server implementation.

Batch jobs with either an *Execution_Time* attribute value set in the future or a *Hold_Types* attribute of value not equal to NO_HOLD, or both, may be routed or held in the routing queue. The treatment of jobs with the *Execution_Time* or *Hold_Types* attributes in a routing queue is implementation-defined.

When a batch job in a routing queue has not been selected to be moved to a new destination and the batch job has a *Hold_Types* attribute value of other than NO_HOLD, then the job should be in the HELD state.

Note: The effect of a hold upon a batch job in a routing queue is implementation-defined. The implementation should use the state that matches whether the batch job can route with a hold or not.

When a batch job in a routing queue has not been selected to be moved to a new destination and the batch job has:

- A *Hold_Types* attribute value of NO_HOLD
- An *Execution_Time* attribute in the past

then the batch job shall be in the QUEUED state.

When a batch job in a routing queue has not been selected to be moved to a new destination and the batch job has:

- A *Hold_Types* attribute value of NO_HOLD
- An *Execution_Time* attribute in the future

then the batch job may be in the WAITING state.

Note: The effect of a future execution time upon a batch job in a routing queue is implementation-defined. The implementation should use the state that matches whether the batch job can route with a hold or not.

Table 3-3 describes the next state of a batch job, given the current state of the batch job and the type of request. Table 3-4 (on page 2405) describes the response of a batch server to a request, given the current state of the batch job and the type of request.

3.2.2 Deferred Batch Services

This section describes the deferred services performed by batch servers: job execution, job routing, job exit, job abort, and the rerunning of jobs after a restart.

3.2.2.1 Batch Job Execution

To execute a batch job is to create a session leader (a process) that runs the shell program indicated by the *Shell_Path_List* attribute of the batch job. The script is passed to the program as its standard input. An implementation may pass the script to the program by other implementation-defined means. At the time a batch job begins execution, it is defined to enter the RUNNING state.

Table 3-3 Next State Table

Request Type	Current State						
	X	Q	R	H	W	E	T
Queue Batch Job Request	Q	e	e	e	e	e	e
Modify Batch Job Request	e	Q	R	H	W	e	T
Delete Batch Job Request	e	X	E	X	X	E	X
Batch Job Message Request	e	Q	R	H	W	E	T
Rerun Batch Job Request	e	e	Q	e	e	e	e
Signal Batch Job Request	e	e	R	H	W	e	e
Batch Job Status Request	e	Q	R	H	W	E	T
Batch Queue Status Request	X	Q	R	H	W	E	T
Server Status Request	X	Q	R	H	W	E	T
Select Batch Jobs Request	X	Q	R	H	W	E	T
Move Batch Job Request	e	Q	R	H	W	e	T
Hold Batch Job Request	e	H	R/H	H	H	e	T
Release Batch Job Request	e	Q	R	Q/W/H	W	e	T
Server Shutdown Request	X	Q	Q	H	W	E	T
Locate Batch Job Request	e	Q	R	H	W	E	T

76651 **Legend**

76652 X Nonexistent

76653 Q QUEUED

76654 R RUNNING

76655 H HELD

76656 W WAITING

76657 E EXITING

76658 T TRANSITING

76659 e Error

76660 A batch server that has an execution queue containing jobs is said to own the queue and manage
 76661 the batch jobs in that queue. A batch server that has been started shall execute the batch jobs in
 76662 the execution queues owned by the batch server. The batch server shall schedule for execution
 76663 those jobs in the execution queues that are in the QUEUED state. The algorithm for scheduling
 76664 jobs is implementation-defined.

76665 A batch server that executes a batch job shall create, in the environment of the session leader of
 76666 the batch job, an environment variable named *PBS_ENVIRONMENT*, the value of which is the
 76667 string *PBS_BATCH* encoded in the portable character set.

76668 A batch server that executes a batch job shall create, in the environment of the session leader of
 76669 the batch job, an environment variable named *PBS_QUEUE*, the value of which is the name of
 76670 the execution queue of the batch job encoded in the portable character set.

76671 To rerun a batch job is to requeue a batch job that is currently executing and then kill the session
 76672 leader of the executing job by sending a SIGKILL prior to completion; see [Section 3.2.3.11](#) (on
 76673 page 2417). A batch server that reruns a batch job shall append the standard output and
 76674 standard error files of the batch job to the corresponding files of the previous execution, if they
 76675 exist, with appropriate annotation. If either file does not exist, that file shall be created as in
 76676 normal execution.

76677

Table 3-4 Results/Output Table

76678

76679

76680

76681

76682

76683

76684

76685

76686

76687

76688

76689

76690

76691

76692

76693

76694

Request Type	Current State						
	X	Q	R	H	W	E	T
<i>Queue Batch Job Request</i>	O	e	e	e	e	e	e
<i>Modify Batch Job Request</i>	e	O	e	O	O	e	e
<i>Delete Batch Job Request</i>	e	O	O	O	O	e	O
<i>Batch Job Message Request</i>	e	e	O	e	e	e	e
<i>Rerun Batch Job Request</i>	e	e	O	e	e	e	e
<i>Signal Batch Job Request</i>	e	e	O	e	e	e	e
<i>Batch Job Status Request</i>	e	O	O	O	O	O	O
<i>Batch Queue Status Request</i>	O	O	O	O	O	O	O
<i>Server Status Request</i>	O	O	O	O	O	O	O
<i>Select Batch Job Request</i>	e	O	O	O	O	O	O
<i>Move Batch Job Request</i>	e	O	O	O	O	e	e
<i>Hold Batch Job Request</i>	e	O	O	O	O	e	e
<i>Release Batch Job Request</i>	e	O	e	O	O	e	e
<i>Server Shutdown Request</i>	O	O	e	O	O	e	e
<i>Locate Batch Job Request</i>	e	O	O	O	O	O	O

76695

Legend

76696

O OK

76697

e Error message

76698

76699

The execution of a batch job by a batch server shall be controlled by job, queue, and server attributes, as defined in this section.

76700

Account_Name Attribute

76701

76702

76703

Batch accounting is an optional feature of batch servers. If a batch server implements accounting, the statements in this section apply and the configuration variable `POSIX2_PBS_ACCOUNTING` shall be set to 1.

76704

76705

A batch server that executes a batch job shall charge the account named in the *Account_Name* attribute of the batch job for resources consumed by the batch job.

76706

76707

If the *Account_Name* attribute of the batch job is absent from the batch job attribute list or is altered while the batch job is in execution, the batch server action is implementation-defined.

76708

Checkpoint Attribute

76709

76710

76711

Batch checkpointing is an optional feature of batch servers. If a batch server implements checkpointing, the statements in this section apply and the configuration variable `POSIX2_PBS_CHECKPOINT` shall be set to 1.

76712

76713

76714

76715

76716

There are two attributes associated with the checkpointing feature: *Checkpoint* and *Minimum_Cpu_Interval*. *Checkpoint* is a batch job attribute, while *Minimum_Cpu_Interval* is a queue attribute. An implementation that does not support checkpointing shall support the *Checkpoint* job attribute to the extent that the batch server shall maintain and pass this attribute to other servers.

76717

76718

76719

The behavior of a batch server that executes a batch job for which the value of the *Checkpoint* attribute is `CHECKPOINT_UNSPECIFIED` is implementation-defined. A batch server that executes a batch job for which the value of the *Checkpoint* attribute is `NO_CHECKPOINT` shall

76720 not checkpoint the batch job.

76721 A batch server that executes a batch job for which the value of the *Checkpoint* attribute is
76722 CHECKPOINT_AT_SHUTDOWN shall checkpoint the batch job only when the batch server
76723 accepts a request to shut down during the time when the batch job is in the RUNNING state.

76724 A batch server that executes a batch job for which the value of the *Checkpoint* attribute is
76725 CHECKPOINT_AT_MIN_CPU_INTERVAL shall checkpoint the batch job at the interval
76726 specified by the *Minimum_Cpu_Interval* attribute of the queue for which the batch job has been
76727 selected. The *Minimum_Cpu_Interval* attribute shall be specified in units of CPU minutes.

76728 A batch server that executes a batch job for which the value of the *Checkpoint* attribute is an
76729 unsigned integer shall checkpoint the batch job at an interval that is the value of either the
76730 *Checkpoint* attribute, or the *Minimum_Cpu_Interval* attribute of the queue for which the batch job
76731 has been selected, whichever is greater. Both intervals shall be in units of CPU minutes. When
76732 the *Minimum_Cpu_Interval* attribute is greater than the *Checkpoint* attribute, the batch job shall
76733 write a warning message to the standard error stream of the batch job.

76734 **Error_Path Attribute**

76735 The *Error_Path* attribute of a running job cannot be changed by a *Modify Batch Job Request*. When
76736 the *Join_Path* attribute of the batch job is set to the value FALSE and the *Keep_Files* attribute of
76737 the batch job does not contain the value KEEP_STD_ERROR, a batch server that executes a batch
76738 job shall perform one of the following actions:

- 76739 • Set the standard error stream of the session leader of the batch job to the path described by
76740 the value of the *Error_Path* attribute of the batch job.
- 76741 • Buffer the standard error of the session leader of the batch job until completion of the batch
76742 job, and when the batch job exits return the contents to the destination described by the
76743 value of the *Error_Path* attribute of the batch job.

76744 Applications shall not rely on having access to the standard error of a batch job prior to the
76745 completion of the batch job.

76746 When the *Error_Path* attribute does not specify a host name, then the batch server shall retain the
76747 standard error of the batch job on the host of execution.

76748 When the *Error_Path* attribute does specify a host name and the *Keep_Files* attribute does not
76749 contain the value KEEP_STD_ERROR, then the final destination of the standard error of the
76750 batch job shall be on the host whose host name is specified.

76751 If the path indicated by the value of the *Error_Path* attribute of the batch job is a relative path, the
76752 batch server shall expand the path relative to the home directory of the user on the host to which
76753 the file is being returned.

76754 When the batch server buffers the standard error of the batch job and the file cannot be opened
76755 for write upon completion of the batch job, then the server shall place the standard error in an
76756 implementation-defined location and notify the user of the location via mail. It shall be possible
76757 for the user to process this mail using the *mailx* utility.

76758 If a batch server that does not buffer the standard error cannot open the standard error path of
76759 the batch job for write access, then the batch server shall abort the batch job.

Execution_Time Attribute

A batch server shall not execute a batch job before the time represented by the value of the *Execution_Time* attribute of the batch job. The *Execution_Time* attribute is defined in seconds since the Epoch.

Hold_Types Attribute

A batch server shall support the following hold types:

- s Can be set or released by a user with at least a privilege level of batch administrator (SYSTEM).
- o Can be set or released by a user with at least a privilege level of batch operator (OPERATOR).
- u Can be set or released by the user with at least a privilege level of user, where the user is defined in the *Job_Owner* attribute (USER).
- n Indicates that none of the *Hold_Types* attributes are set (NO_HOLD).

An implementation may define other hold types. Any additional hold types, how they are specified, their internal representation, their behavior, and how they affect the behavior of other utilities are implementation-defined.

The value of the *Hold_Types* attribute shall be the union of the valid hold types ('s', 'o', 'u', and any implementation-defined hold types), or 'n'.

A batch server shall not execute a batch job if the *Hold_Types* attribute of the batch job has a value other than NO_HOLD. If the *Hold_Types* attribute of the batch job has a value other than NO_HOLD, the batch job shall be in the HELD state.

Job_Owner Attribute

The *Job_Owner* attribute consists of a pair of user name and host name values of the form:

`username@hostname`

A batch server that accepts a *Queue Batch Job Request* shall set the *Job_Owner* attribute to a string that is the *username@hostname* of the user who submitted the job.

Join_Path Attribute

A batch server that executes a batch job for which the value of the *Join_Path* attribute is TRUE shall ignore the value of the *Error_Path* attribute and merge the standard error of the batch job with the standard output of the batch job.

Keep_Files Attribute

A batch server that executes a batch job for which the value of the *Keep_Files* attribute includes the value KEEP_STD_OUTPUT shall retain the standard output of the batch job on the host where execution occurs. The standard output shall be retained in the home directory of the user under whose user ID the batch job is executed and the filename shall be the default filename for the standard output as defined under the `-o` option of the *qsub* utility. The *Output_Path* attribute is not modified.

A batch server that executes a batch job for which the value of the *Keep_Files* attribute includes the value KEEP_STD_ERROR shall retain the standard error of the batch job on the host where execution occurs. The standard error shall be retained in the home directory of the user under whose user ID the batch job is executed and the filename shall be the default filename for

76801 standard error as defined under the `-e` option of the *qsub* utility. The *Error_Path* attribute is not
 76802 modified.

76803 A batch server that executes a batch job for which the value of the *Keep_Files* attribute includes
 76804 values other than `KEEP_STD_OUTPUT` and `KEEP_STD_ERROR` shall retain these other files on
 76805 the host where execution occurs. These files (with implementation-defined names) shall be
 76806 retained in the home directory of the user under whose user identifier the batch job is executed.

76807 **Mail_Points and Mail_Users Attributes**

76808 A batch server that executes a batch job for which one of the values of the *Mail_Points* attribute is
 76809 the value `MAIL_AT_BEGINNING` shall send a mail message to each user account listed in the
 76810 *Mail_Users* attribute of the batch job.

76811 The mail message shall contain at least the batch job identifier, queue, and server at which the
 76812 batch job currently resides, and the *Job_Owner* attribute.

76813 **Output_Path Attribute**

76814 The *Output_Path* attribute of a running job cannot be changed by a *Modify Batch Job Request*.
 76815 When the *Keep_Files* attribute of the batch job does not contain the value `KEEP_STD_OUTPUT`, a
 76816 batch server that executes a batch job shall either:

- 76817 • Set the standard output stream of the session leader of the batch job to the destination
 76818 described by the value of the *Output_Path* attribute of the batch job.

76819 or:

- 76820 • Buffer the standard output of the session leader of the batch job until completion of the
 76821 batch job, and when the batch job exits return the contents to the destination described by
 76822 the value of the *Output_Path* attribute of the batch job.

76823 When the *Output_Path* attribute does not specify a host name, then the batch server shall retain
 76824 the standard output of the batch job on the host of execution.

76825 When the *Keep_Files* attribute does not contain the value `KEEP_STD_OUTPUT` and the
 76826 *Output_Path* attribute does specify a host name, then the final destination of the standard output
 76827 of the batch job shall be on the host specified.

76828 If the path specified in the *Output_Path* attribute of the batch job is a relative path, the batch
 76829 server shall expand the path relative to the home directory of the user on the host to which the
 76830 file is being returned.

76831 Whether or not the batch server buffers the standard output of the batch job until completion of
 76832 the batch job is implementation-defined. Applications shall not rely on having access to the
 76833 standard output of a batch job prior to the completion of the batch job.

76834 When the batch server does buffer the standard output of the batch job and the file cannot be
 76835 opened for write upon completion of the batch job, then the batch server shall place the standard
 76836 output in an implementation-defined location and notify the user of the location via mail. It shall
 76837 be possible for the user to process this mail using the *mailx* utility.

76838 If a batch server that does not buffer the standard output cannot open the standard output path
 76839 of the batch job for write access, then the batch server shall abort the batch job.

Priority Attribute

A batch server implementation may choose to preferentially execute a batch job based on the *Priority* attribute. The interpretation of the batch job *Priority* attribute by a batch server is implementation-defined. If an implementation uses the *Priority* attribute, it shall interpret larger values of the *Priority* attribute to mean the batch job shall be preferentially selected for execution.

Rerunable Attribute

A batch job that began execution but did not complete, because the batch server either shut down or terminated abnormally, shall be requeued if the *Rerunable* attribute of the batch job has the value TRUE.

If a batch job, which was requeued after beginning execution but prior to completion, has a valid checkpoint file and the batch server supports checkpointing, then the batch job shall be restarted from the last valid checkpoint.

If the batch job cannot be restarted from a checkpoint, then when a batch job has a *Rerunable* attribute value of TRUE and was requeued after beginning execution but prior to completion, the batch server shall place the batch job into execution at the beginning of the job.

When a batch job has a *Rerunable* attribute value other than TRUE and was requeued after beginning execution but prior to completion, and the batch job cannot be restarted from a checkpoint, then the batch server shall abort the batch job.

Resource_List Attribute

A batch server that executes a batch job shall establish the resource limits of the session leader of the batch job according to the values of the *Resource_List* attribute of the batch job. Resource limits shall be enforced by an implementation-defined method.

Shell_Path_List Attribute

The *Shell_Path_List* job attribute consists of a list of pairs of pathname and host name values. The host name component can be omitted, in which case the pathname serves as the default pathname when a batch server cannot find the name of the host on which it is running in the list.

A batch server that executes a batch job shall select, from the value of the *Shell_Path_List* attribute of the batch job, a pathname where the shell to execute the batch job shall be found. The batch server shall select the pathname, in order of preference, according to the following methods:

- Select the pathname that contains the name of the host on which the batch server is running.
- Select the pathname for which the host name has been omitted.
- Select the pathname for the login shell of the user under which the batch job is to execute.

If the shell path value selected is an invalid pathname, the batch server shall abort the batch job.

If the value of the selected pathname from the *Shell_Path_List* attribute of the batch job represents a partial path, the batch server shall expand the path relative to a path that is implementation-defined.

The batch server that executes the batch job shall execute the program that was selected from the *Shell_Path_List* attribute of the batch job. The batch server shall pass the path to the script of the batch job as the first argument to the shell program.

User_List Attribute

The *User_List* job attribute consists of a list of pairs of user name and host name values. The host name component can be omitted, in which case the user name serves as a default when a batch server cannot find the name of the host on which it is running in the list.

A batch server that executes a batch job shall select, from the value of the *User_List* attribute of the batch job, a user name under which to create the session leader. The server shall select the user name, in order of preference, according to the following methods:

- Select the user name of a value that contains the name of the host on which the batch server executes.
- Select the user name of a value for which the host name has been omitted.
- Select the user name from the *Job_Owner* attribute of the batch job.

Variable_List Attribute

A batch server that executes a batch job shall create, in the environment of the session leader of the batch job, each environment variable listed in the *Variable_List* attribute of the batch job, and set the value of each such environment variable to that of the corresponding variable in the variable list.

3.2.2.2 Batch Job Routing

To route a batch job is to select a queue from a list and move the batch job to that queue.

A batch server that has routing queues, which have been started, shall route the jobs in the routing queues owned by the batch server. A batch server may delay the routing of a batch job. The algorithm for selecting a batch job and the queue to which it will be routed is implementation-defined.

When a routing queue has multiple possible destinations specified, then the precedence of the destinations is implementation-defined.

A batch server that routes a batch job to a queue at another server shall move the batch job into the target queue with a *Queue Batch Job Request*.

If the target server rejects the *Queue Batch Job Request*, the routing server shall retry routing the batch job or abort the batch job. A batch server that retries failed routings shall provide a means for the batch administrator to specify the number of retries and the minimum period of time between retries. The means by which an administrator specifies the number of retries and the delay between retries is implementation-defined. When the number of retries specified by the batch administrator has been exhausted, the batch server shall abort the batch job and perform the functions of *Batch Job Exit*; see [Section 3.2.2.3](#).

3.2.2.3 Batch Job Exit

For each job in the EXITING state, the batch server that exited the batch job shall perform the following deferred services in the order specified:

1. If buffering standard error, move that file into the location specified by the *Error_Path* attribute of the batch job.
2. If buffering standard output, move that file into the location specified by the *Output_Path* attribute of the batch job.

3. If the *Mail_Points* attribute of the batch job includes MAIL_AT_EXIT, send mail to the users listed in the *Mail_Users* attribute of the batch job. The mail message shall contain at least the batch job identifier, queue, and server at which the batch job currently resides, and the *Job_Owner* attribute.
4. Remove the batch job from the queue.

If a batch server that buffers the standard error output cannot return the standard error file to the standard error path at the time the batch job exits, the batch server shall do one of the following:

- Mail the standard error file to the batch job owner.
- Save the standard error file and mail the location and name of the file where the standard error is stored to the batch job owner.
- Save the standard error file and notify the user by other implementation-defined means.

If a batch server that buffers the standard output cannot return the standard output file to the standard output path at the time the batch job exits, the batch server shall do one of the following:

- Mail the standard output file to the batch job owner.
- Save the standard output file and mail the location and name of the file where the standard output is stored to the batch job owner.
- Save the standard output file and notify the user by other implementation-defined means.

At the conclusion of job exit processing, the batch job is no longer managed by a batch server.

3.2.2.4 Batch Server Restart

A batch server that has been either shutdown or terminated abnormally, and has returned to operation, is said to have “restarted”.

Upon restarting, a batch server shall requeue those jobs managed by the batch server that were in the RUNNING state at the time the batch server shut down and for which the *Rerunable* attribute of the batch job has the value TRUE.

Queues are defined to be non-volatile. A batch server shall store the content of queues that it controls in such a way that server and system shutdowns do not erase the content of the queues.

3.2.2.5 Batch Job Abort

A batch server that cannot perform a deferred service for a batch job shall abort the batch job.

A batch server that aborts a batch job shall perform the following services:

- Delete the batch job from the queue in which it resides.
- If the *Mail_Points* attribute of the batch job includes the value MAIL_AT_ABORT, send mail to the users listed in the value of the *Mail_Users* attribute of the job. The mail message shall contain at least the batch job identifier, queue, and server at which the batch job currently resides, the *Job_Owner* attribute, and the reason for the abort.
- If the batch job was in the RUNNING state, terminate the session leader of the executing job by sending the session leader a SIGKILL, place the batch job in the EXITING state, and perform the actions of *Batch Job Exit*.

3.2.3 Requested Batch Services

This section describes the services provided by batch servers in response to requests from clients. Table 3-5 summarizes the current set of batch service requests and for each gives its type (deferred or not) and whether it is an optional function.

Table 3-5 Batch Services Summary

Batch Service	Deferred	Optional
<i>Batch Job Execution</i>	Yes	No
<i>Batch Job Routing</i>	Yes	No
<i>Batch Job Exit</i>	Yes	No
<i>Batch Server Restart</i>	Yes	No
<i>Batch Job Abort</i>	Yes	No
<i>Delete Batch Job Request</i>	No	No
<i>Hold Batch Job Request</i>	No	No
<i>Batch Job Message Request</i>	No	Yes
<i>Batch Job Status Request</i>	No	No
<i>Locate Batch Job Request</i>	No	Yes
<i>Modify Batch Job Request</i>	No	No
<i>Move Batch Job Request</i>	No	No
<i>Queue Batch Job Request</i>	No	No
<i>Batch Queue Status Request</i>	No	No
<i>Release Batch Job Request</i>	No	No
<i>Rerun Batch Job Request</i>	No	No
<i>Select Batch Jobs Request</i>	No	No
<i>Server Shutdown Request</i>	No	No
<i>Server Status Request</i>	No	No
<i>Signal Batch Job Request</i>	No	No
<i>Track Batch Job Request</i>	No	Yes

If a request is rejected because the batch client is not authorized to perform the action, the batch server shall return the same status as when the batch job does not exist.

3.2.3.1 Delete Batch Job Request

A batch job is defined to have been deleted when it has been removed from the queue in which it resides and not instantiated in another queue. A client requests that the server that manages a batch job delete the batch job. Such a request is called a *Delete Batch Job Request*.

A batch server shall reject a *Delete Batch Job Request* if any of the following statements are true:

- The user of the batch client is not authorized to delete the designated job.
- The designated job is not managed by the batch server.
- The designated job is in a state inconsistent with the delete request.

A batch server may reject a *Delete Batch Job Request* for other implementation-defined reasons. The method used to determine whether the user of a client is authorized to perform the requested action is implementation-defined.

A batch server requested to delete a batch job shall delete the batch job if the batch job exists and is not in the EXITING state.

A batch server that deletes a batch job in the RUNNING state shall send a SIGKILL signal to the

77003 session leader of the batch job. It is implementation-defined whether additional signals are sent
77004 to the session leader of the job prior to sending the SIGKILL signal.

77005 A batch server that deletes a batch job in the RUNNING state shall place the batch job in the
77006 EXITING state after it has killed the session leader of the batch job and shall perform the actions
77007 of *Batch Job Exit*.

77008 3.2.3.2 *Hold Batch Job Request*

77009 A batch client can request that the batch server add one or more holds to a batch job. Such a
77010 request is called a *Hold Batch Job Request*.

77011 A batch server shall reject a *Hold Batch Job Request* if any of the following statements are true:

- 77012 • The batch server does not support one or more of the requested holds to be added to the
77013 batch job.
- 77014 • The user of the batch client is not authorized to add one or more of the requested holds to
77015 the batch job.
- 77016 • The batch server does not manage the specified job.
- 77017 • The designated job is in the EXITING state.

77018 A batch server may reject a *Hold Batch Job Request* for other implementation-defined reasons. The
77019 method used to determine whether the user of a client is authorized to perform the requested
77020 action is implementation-defined.

77021 A batch server that accepts a *Hold Batch Job Request* for a batch job in the RUNNING state shall
77022 place a hold on the batch job. The effects, if any, the hold will have on a batch job in the
77023 RUNNING state are implementation-defined.

77024 A batch server that accepts a *Hold Batch Job Request* shall add each type of hold listed in the *Hold*
77025 *Batch Job Request*, that is not already present, to the value of the *Hold_Types* attribute of the batch
77026 job.

77027 3.2.3.3 *Batch Job Message Request*

77028 *Batch Job Message Request* is an optional feature of batch servers. If an implementation supports
77029 *Batch Job Message Request*, the statements in this section apply and the configuration variable
77030 POSIX2_PBS_MESSAGE shall be set to 1.

77031 A batch client can request that a batch server write a message into certain output files of a batch
77032 job. Such a request is called a *Batch Job Message Request*.

77033 A batch server shall reject a *Batch Job Message Request* if any of the following statements are true:

- 77034 • The batch server does not support sending messages to jobs.
- 77035 • The user of the batch client is not authorized to post a message to the designated job.
- 77036 • The designated job does not exist on the batch server.
- 77037 • The designated job is not in the RUNNING state.

77038 A batch server may reject a *Batch Job Message Request* for other implementation-defined reasons.
77039 The method used to determine whether the user of a client is authorized to perform the
77040 requested action is implementation-defined.

77041 A batch server that accepts a *Batch Job Message Request* shall write the message sent by the batch
77042 client into the files indicated by the batch client.

77043 3.2.3.4 *Batch Job Status Request*

77044 A batch client can request that a batch server respond with the status and attributes of a batch
77045 job. Such a request is called a *Batch Job Status Request*.

77046 A batch server shall reject a *Batch Job Status Request* if any of the following statements are true:

- 77047 • The user of the batch client is not authorized to query the status of the designated job.
- 77048 • The designated job is not managed by the batch server.

77049 A batch server may reject a *Batch Job Status Request* for other implementation-defined reasons.
77050 The method used to determine whether the user of a client is authorized to perform the
77051 requested action is implementation-defined.

77052 A batch server that accepts a *Batch Job Status Request* shall return a *Batch Job Status Message* to the
77053 batch client.

77054 A batch server may return other information in response to a *Batch Job Status Request*.

77055 3.2.3.5 *Locate Batch Job Request*

77056 *Locate Batch Job Request* is an optional feature of batch servers. If an implementation supports
77057 *Locate Batch Job Request*, the statements in this section apply and the configuration variable
77058 POSIX2_PBS_LOCATE shall be set to 1.

77059 A batch client can ask a batch server to respond with the location of a batch job that was created
77060 by the batch server. Such a request is called a *Locate Batch Job Request*.

77061 A batch server that accepts a *Locate Batch Job Request* shall return a *Batch Job Location Message* to
77062 the batch client.

77063 A batch server may reject a *Locate Batch Job Request* for a batch job that was not created by that
77064 server.

77065 A batch server may reject a *Locate Batch Job Request* for a batch job that is no longer managed by
77066 that server; that is, for a batch job that is not in a queue owned by that server.

77067 A batch server may reject a *Locate Batch Job Request* for other implementation-defined reasons.

77068 3.2.3.6 *Modify Batch Job Request*

77069 Batch clients modify (alter) the attributes of a batch job by making a request to the server that
77070 manages the batch job. Such a request is called a *Modify Batch Job Request*.

77071 A batch server shall reject a *Modify Batch Job Request* if any of the following statements are true:

- 77072 • The user of the batch client is not authorized to make the requested modification to the
77073 batch job.
- 77074 • The designated job is not managed by the batch server.
- 77075 • The requested modification is inconsistent with the state of the batch job.
- 77076 • An unrecognized resource is requested for a batch job in an execution queue.

77077 A batch server may reject a *Modify Batch Job Request* for other implementation-defined reasons.
77078 The method used to determine whether the user of a client is authorized to perform the
77079 requested action is implementation-defined.

77080 A batch server that accepts a *Modify Batch Job Request* shall modify all the specified attributes of
77081 the batch job. A batch server that rejects a *Modify Batch Job Request* shall modify none of the

77082 attributes of the batch job.

77083 If the servicing by a batch server of an otherwise valid request would result in no change, then
77084 the batch server shall indicate successful completion of the request.

77085 3.2.3.7 *Move Batch Job Request*

77086 A batch client can request that a batch server move a batch job to another destination. Such a
77087 request is called a *Move Batch Job Request*.

77088 A batch server shall reject a *Move Batch Job Request* if any of the following statements are true:

- 77089 • The user of the batch client is not authorized to remove the designated job from the queue
77090 in which the batch job resides.
- 77091 • The user of the batch client is not authorized to move the designated job to the destination.
- 77092 • The designated job is not managed by the batch server.
- 77093 • The designated job is in the EXITING state.
- 77094 • The destination is inaccessible.

77095 A batch server can reject a *Move Batch Job Request* for other implementation-defined reasons. The
77096 method used to determine whether the user of a client is authorized to perform the requested
77097 action is implementation-defined.

77098 A batch server that accepts a *Move Batch Job Request* shall perform the following services:

- 77099 • Queue the designated job at the destination.
- 77100 • Remove the designated job from the queue in which the batch job resides.

77101 If the destination resides on another batch server, the batch server shall queue the batch job at
77102 the destination by sending a *Queue Batch Job Request* to the other server. If the *Queue Batch Job*
77103 *Request* fails, the batch server shall reject the *Move Batch Job Request*. If the *Queue Batch Job*
77104 *Request* succeeds, the batch server shall remove the batch job from its queue.

77105 The batch server shall not modify any attributes of the batch job.

77106 3.2.3.8 *Queue Batch Job Request*

77107 A batch queue is controlled by one and only one batch server. A batch server is said to own the
77108 queues that it controls. Batch clients make requests of batch servers to have jobs queued. Such a
77109 request is called a *Queue Batch Job Request*.

77110 A batch server requested to queue a batch job for which the queue is not specified shall select an
77111 implementation-defined queue for the batch job. Such a queue is called the “default queue” of
77112 the batch server. The implementation shall provide the means for a batch administrator to
77113 specify the default queue. The queue, whether specified or defaulted, is called the “target
77114 queue”.

77115 A batch server shall reject a *Queue Batch Job Request* if any of the following statements are true:

- 77116 • The client is not authorized to create a batch job in the target queue.
- 77117 • The request specifies a queue that does not exist on the batch server.
- 77118 • The target queue is an execution queue and the batch server cannot satisfy a resource
77119 requirement of the batch job.

- 77120 • The target queue is an execution queue and an unrecognized resource is requested.
- 77121 • The target queue is an execution queue, the batch server does not support checkpointing,
- 77122 and the value of the *Checkpoint* attribute of the batch job is not NO_CHECKPOINT.
- 77123 • The job requires access to a user identifier that the batch client is not authorized to access.

77124 A batch server may reject a *Queue Batch Job Request* for other implementation-defined reasons.

77125 A batch server that accepts a *Queue Batch Job Request* for a batch job for which the
 77126 PBS_O_QUEUE value is missing from the value of the *Variable_List* attribute of the batch job
 77127 shall add that variable to the list and set the value to the name of the target queue. Once set, no
 77128 server shall change the value of PBS_O_QUEUE, even if the batch job is moved to another
 77129 queue.

77130 A batch server that accepts a *Queue Batch Job Request* for a batch job for which the PBS_JOBID
 77131 value is missing from the value of the *Variable_List* attribute shall add that variable to the list and
 77132 set the value to the batch job identifier assigned by the server in the format:

77133 `sequence_number.server`

77134 A batch server that accepts a *Queue Batch Job Request* for a batch job for which the
 77135 PBS_JOBNAME value is missing from the value of the *Variable_List* attribute of the batch job
 77136 shall add that variable to the list and set the value to the *Job_Name* attribute of the batch job.

77137 3.2.3.9 *Batch Queue Status Request*

77138 A batch client can request that a batch server respond with the status and attributes of a queue.
 77139 Such a request is called a *Batch Queue Status Request*.

77140 A batch server shall reject a *Batch Queue Status Request* if any of the following statements are
 77141 true:

- 77142 • The user of the batch client is not authorized to query the status of the designated queue.
- 77143 • The designated queue does not exist on the batch server.

77144 A batch server may reject a *Batch Queue Status Request* for other implementation-defined reasons.
 77145 The method used to determine whether the user of a client is authorized to perform the
 77146 requested action is implementation-defined.

77147 A batch server that accepts a *Batch Queue Status Request* shall return a *Batch Queue Status Reply* to
 77148 the batch client.

77149 3.2.3.10 *Release Batch Job Request*

77150 A batch client can request that the server remove one or more holds from a batch job. Such a
 77151 request is called a *Release Batch Job Request*.

77152 A batch server shall reject a *Release Batch Job Request* if any of the following statements are true:

- 77153 • The user of the batch client is not authorized to remove one or more of the requested holds
- 77154 from the batch job.
- 77155 • The batch server does not manage the specified job.

77156 A batch server may reject a *Release Batch Job Request* for other implementation-defined reasons.
 77157 The method used to determine whether the user of a client is authorized to perform the
 77158 requested action is implementation-defined.

77159 A batch server that accepts a *Release Batch Job Request* shall remove each type of hold listed in the

77160 *Release Batch Job Request*, that is present, from the value of the *Hold_Types* attribute of the batch
77161 job.

77162 3.2.3.11 *Rerun Batch Job Request*

77163 To rerun a batch job is to kill the session leader of the batch job and leave the batch job eligible
77164 for re-execution. A batch client can request that a batch server rerun a batch job. Such a request
77165 is called *Rerun Batch Job Request*.

77166 A batch server shall reject a *Rerun Batch Job Request* if any of the following statements are true:

- 77167 • The user of the batch client is not authorized to rerun the designated job.
- 77168 • The *Rerunable* attribute of the designated job has the value FALSE.
- 77169 • The designated job is not in the RUNNING state.
- 77170 • The batch server does not manage the designated job.

77171 A batch server may reject a *Rerun Batch Job Request* for other implementation-defined reasons.
77172 The method used to determine whether the user of a client is authorized to perform the
77173 requested action is implementation-defined.

77174 A batch server that rejects a *Rerun Batch Job Request* shall in no way modify the execution of the
77175 batch job.

77176 A batch server that accepts a request to rerun a batch job shall perform the following services:

- 77177 • Requeue the batch job in the execution queue in which it was executing.
- 77178 • Send a SIGKILL signal to the process group of the session leader of the batch job.

77179 An implementation may indicate to the batch job owner that the batch job has been rerun.
77180 Whether and how the batch job owner is notified that a batch job is rerun is implementation-
77181 defined.

77182 A batch server that reruns a batch job may send other implementation-defined signals to the
77183 session leader of the batch job prior to sending the SIGKILL signal.

77184 A batch server may preferentially select a rerun job for execution. Whether rerun jobs shall be
77185 selected for execution before other jobs is implementation-defined.

77186 3.2.3.12 *Select Batch Jobs Request*

77187 A batch client can request from a batch server a list of jobs managed by that server that match a
77188 list of selection criteria. Such a request is called a *Select Batch Jobs Request*. All the batch jobs
77189 managed by the batch server that receives the request are candidates for selection.

77190 A batch server that accepts a *Select Batch Jobs Request* shall return a list of zero or more job
77191 identifiers that correspond to jobs that meet the selection criteria.

77192 If the batch client is not authorized to query the status of a batch job, the batch server shall not
77193 select the batch job.

77194 3.2.3.13 *Server Shutdown Request*

77195 A batch server is defined to have shut down when it does not respond to requests from clients
 77196 and does not perform deferred services for jobs. A batch client can request that a batch server
 77197 shut down. Such a request is called a *Server Shutdown Request*.

77198 A batch server shall reject a *Server Shutdown Request* from a client that is not authorized to shut
 77199 down the batch server. The method used to determine whether the user of a client is authorized
 77200 to perform the requested action is implementation-defined.

77201 A batch server may reject a *Server Shutdown Request* for other implementation-defined reasons.
 77202 The reasons for which a *Server Shutdown Request* may be rejected are implementation-defined.

77203 At server shutdown, a batch server shall do, in order of preference, one of the following:

- 77204 • If checkpointing is implemented and the batch job is checkpointable, then checkpoint the
 77205 batch job and requeue it.
- 77206 • If the batch job is rerunnable, then requeue the batch job to be rerun (restarted from the
 77207 beginning).
- 77208 • Abort the batch job.

77209 3.2.3.14 *Server Status Request*

77210 A batch client can request that a batch server respond with the status and attributes of the batch
 77211 server. Such a request is called a *Server Status Request*.

77212 A batch server shall reject a *Server Status Request* if the following statement is true:

- 77213 • The user of the batch client is not authorized to query the status of the designated server.

77214 A batch server may reject a *Server Status Request* for other implementation-defined reasons. The
 77215 method used to determine whether the user of a client is authorized to perform the requested
 77216 action is implementation-defined.

77217 A batch server that accepts a *Server Status Request* shall return a *Server Status Reply* to the batch
 77218 client.

77219 3.2.3.15 *Signal Batch Job Request*

77220 A batch client can request that a batch server signal the session leader of a batch job. Such a
 77221 request is called a *Signal Batch Job Request*.

77222 A batch server shall reject a *Signal Batch Job Request* if any of the following statements are true:

- 77223 • The user of the batch client is not authorized to signal the batch job.
- 77224 • The job is not in the RUNNING state.
- 77225 • The batch server does not manage the designated job.
- 77226 • The requested signal is not supported by the implementation.

77227 A batch server may reject a *Signal Batch Job Request* for other implementation-defined reasons.
 77228 The method used to determine whether the user of a client is authorized to perform the
 77229 requested action is implementation-defined.

77230 A batch server that accepts a request to signal a batch job shall send the signal requested by the
 77231 batch client to the process group of the session leader of the batch job.

77232 3.2.3.16 *Track Batch Job Request*

77233 *Track Batch Job Request* is an optional feature of batch servers. If an implementation supports
 77234 *Track Batch Job Request*, the statements in this section apply and the configuration variable
 77235 POSIX2_PBS_TRACK shall be set to 1.

77236 *Track Batch Job Request* provides a method for tracking the current location of a batch job. Clients
 77237 may use the tracking information to determine the batch server that should receive a batch
 77238 server request.

77239 If *Track Batch Job Request* is supported by a batch server, then when the batch server queues a
 77240 batch job as a result of a *Queue Batch Job Request*, and the batch server is not the batch server that
 77241 created the batch job, the batch server shall send a *Track Batch Job Request* to the batch server that
 77242 created the job.

77243 If *Track Batch Job Request* is supported by a batch server, then the *Track Batch Job Request* may also
 77244 be sent to other servers as a backup to the primary server. The method by which backup servers
 77245 are specified is implementation-defined.

77246 If *Track Batch Job Request* is supported by a batch server that receives a *Track Batch Job Request*,
 77247 then the batch server shall record the current location of the batch job as contained in the
 77248 request.

77249 **3.3 Common Behavior for Batch Environment Utilities**77250 **3.3.1 Batch Job Identifier**

77251 A utility shall recognize *job_identifiers* of the format:

77252 [sequence_number] [.server_name] [@server]

77253 where:

77254 *sequence_number* An integer that, when combined with *server_name*, provides a batch job
 77255 identifier that is unique within the batch system.

77256 *server_name* The name of the batch server to which the batch job was originally submitted.

77257 *server* The name of the batch server that is currently managing the batch job.

77258 If the application omits the batch *server_name* portion of a batch job identifier, a utility shall use
 77259 the name of a default batch server.

77260 If the application omits the batch *server* portion of a batch job identifier, a utility shall use:

- 77261 • The batch server indicated by *server_name*, if present
- 77262 • The name of the default batch server
- 77263 • The name of the batch server that is currently managing the batch job

77264 If only @*server* is specified, then the status of all jobs owned by the user on the requested server
 77265 is listed.

77266 The means by which a utility determines the default batch server is implementation-defined.

77267 If the application presents the batch *server* portion of a batch job identifier to a utility, the utility
 77268 shall send the request to the specified server.

A strictly conforming application shall use the syntax described for the job identifier. Whenever a batch job identifier is specified whose syntax is not recognized by an implementation, then a message for each error that occurs shall be written to standard error and the utility shall exit with an exit status greater than zero.

When a batch job identifier is supplied as an argument to a batch utility and the *server_name* portion of the batch job identifier is omitted, then the utility shall use the name of the default batch server.

When a batch job identifier is supplied as an argument to a batch utility and the batch *server* portion of the batch job identifier is omitted, then the utility shall use either:

- The name of the default batch server
- or:
- The name of the batch server that is currently managing the batch job

When a batch job identifier is supplied as an argument to a batch utility and the batch *server* portion of the batch job identifier is specified, then the utility shall send the required *Batch Server Request* to the specified server.

3.3.2 Destination

The utility shall recognize a *destination* of the format:

[*queue*] [*@server*]

where:

queue The name of a valid execution or routing queue at the batch server denoted by *@server*, defined as a string of up to 15 alphanumeric characters in the portable character set (see XBD [Section 6.1](#), on page 125) where the first character is alphabetic.

server The name of a batch server, defined as a string of alphanumeric characters in the portable character set.

If the application omits the batch *server* portion of a destination, then the utility shall use either:

- The name of the default batch server
- or:
- The name of the batch server that is currently managing the batch job

The means by which a utility determines the default batch server is implementation-defined.

If the application omits the *queue* portion of a destination, then the utility shall use the name of the default queue at the batch server chosen. The means by which a batch server determines its default queue is implementation-defined. If a destination is specified in the *queue@server* form, then the utility shall use the specified queue at the specified server.

A strictly conforming application shall use the syntax described for a destination. Whenever a destination is specified whose syntax is not recognized by an implementation, then a message shall be written to standard error and the utility shall exit with an exit status greater than zero.

3.3.3 Multiple Keyword-Value Pairs

For each option that can have multiple keyword-value pair arguments, the following rules shall apply. Examples of options that can have list-oriented option-arguments are `-u value@keyword` and `-l keyword=value`.

1. If a batch utility is presented with a list-oriented option-argument for which a keyword has a corresponding value that begins with a single or double-quote, then the utility shall stop interpreting the input stream for delimiters until a second single or double-quote, respectively, is encountered. This feature allows some flexibility for a <comma> (',') or <equals-sign> ('=') to be part of the value string for a particular keyword; for example:

```
keywd1='val1,val2',keywd2="val3,val4"
```

Note: This may require the user to escape the quotes as in the following command:

```
foo -xkeywd1=\'val1,val2\',keywd2=\"val3,val4\"
```

2. If a batch server is presented with a list-oriented attribute that has a keyword that was encountered earlier in the list, then the later entry for that keyword shall replace the earlier entry.
3. If a batch server is presented with a list-oriented attribute that has a keyword without any corresponding value of the form *keyword=* or *@keyword* and the same keyword was encountered earlier in the list, then the prior entry for that keyword shall be ignored by the batch server.
4. If a batch utility is expecting a list-oriented option-argument entry of the form *keyword=value*, but is presented with an entry of the form *keyword* without any corresponding *value*, then the entry shall be treated as though a default value of NULL was assigned (that is, *keyword=NULL*) for entry parsing purposes. The utility shall include only the keyword, not the NULL value, in the associated job attribute.
5. If a batch utility is expecting a list-oriented option-argument entry of the form *value@keyword*, but is presented with an entry of the form *value* without any corresponding *keyword*, then the entry shall be treated as though a keyword of NULL was assigned (that is, *value@NULL*) for entry parsing purposes. The utility shall include only the value, not the NULL keyword, in the associated job attribute.
6. A batch server shall accept a list-oriented attribute that has multiple occurrences of the same keyword, interpreting the keywords, in order, with the last value encountered taking precedence over prior instances of the same keyword. This rule allows, but does not require, a batch utility to preprocess the attribute to remove duplicate keywords.
7. If a batch utility is presented with multiple list-oriented option-arguments on the command line or in script directives, or both, for a single option, then the utility shall concatenate, in order, any command line keyword and value pairs to the end of any directive keyword and value pairs separated by a single <comma> to produce a single string that is an equivalent, valid option-argument. The resulting string shall be assigned to the associated attribute of the batch job (after optionally removing duplicate entries as described in item 6).

77346

Chapter 4

77347

Utilities

77348

This chapter contains the definitions of the utilities, as follows:

77349

- Mandatory utilities that are present on every conformant system

77350

- Optional utilities that are present only on systems supporting the associated option; see [Section 1.7.1](#) (on page 7) for information on the options in this volume of POSIX.1-2008

77351

NAME

admin — create and administer SCCS files (**DEVELOPMENT**)

SYNOPSIS

```

admin -i[name] [-n] [-a login] [-d flag] [-e login] [-f flag]
      [-m mrlist] [-r rel] [-t[name] [-y[comment]]] newfile

admin -n [-a login] [-d flag] [-e login] [-f flag] [-m mrlist]
      [-t[name]] [-y[comment]] newfile...

admin [-a login] [-d flag] [-m mrlist] [-r rel] [-t[name]] file...

admin -h file...

admin -z file...
```

DESCRIPTION

The *admin* utility shall create new SCCS files or change parameters of existing ones. If a named file does not exist, it shall be created, and its parameters shall be initialized according to the specified options. Parameters not initialized by an option shall be assigned a default value. If a named file does exist, parameters corresponding to specified options shall be changed, and other parameters shall be left as is.

All SCCS filenames supplied by the application shall be of the form *s.filename*. New SCCS files shall be given read-only permission mode. Write permission in the parent directory is required to create a file. All writing done by *admin* shall be to a temporary *x-file*, named *x.filename* (see *get*) created with read-only mode if *admin* is creating a new SCCS file, or created with the same mode as that of the SCCS file if the file already exists. After successful execution of *admin*, the SCCS file shall be removed (if it exists), and the *x-file* shall be renamed with the name of the SCCS file. This ensures that changes are made to the SCCS file only if no errors occur.

The *admin* utility shall also use a transient lock file (named *z.filename*), which is used to prevent simultaneous updates to the SCCS file; see *get*.

OPTIONS

The *admin* utility shall conform to XBD [Section 12.2](#) (on page 215), except that the *-i*, *-t*, and *-y* options have optional option-arguments. These optional option-arguments shall not be presented as separate arguments. The following options are supported:

- n** Create a new SCCS file. When *-n* is used without *-i*, the SCCS file shall be created with control information but without any file data.
- i[name]** Specify the *name* of a file from which the text for a new SCCS file shall be taken. The text constitutes the first delta of the file (see the *-r* option for the delta numbering scheme). If the *-i* option is used, but the *name* option-argument is omitted, the text shall be obtained by reading the standard input. If this option is omitted, the SCCS file shall be created with control information but without any file data. The *-i* option implies the *-n* option.
- r SID** Specify the SID of the initial delta to be inserted. This SID shall be a trunk SID; that is, the branch and sequence numbers shall be zero or missing. The level number is optional, and defaults to 1.
- t[name]** Specify the *name* of a file from which descriptive text for the SCCS file shall be taken. In the case of existing SCCS files (neither *-i* nor *-n* is specified):

77394		• A -t option without a <i>name</i> option-argument shall cause the removal of
77395		descriptive text (if any) currently in the SCCS file.
77396		• A -t option with a <i>name</i> option-argument shall cause the text (if any) in the
77397		named file to replace the descriptive text (if any) currently in the SCCS file.
77398	-f flag	Specify a <i>flag</i> , and, possibly, a value for the <i>flag</i> , to be placed in the SCCS file.
77399		Several -f options may be supplied on a single <i>admin</i> command line.
77400		Implementations shall recognize the following flags and associated values:
77401	b	Allow use of the -b option on a <i>get</i> command to create branch deltas.
77402	cceil	Specify the highest release (that is, ceiling), a number less than or equal to
77403		9999, which may be retrieved by a <i>get</i> command for editing. The default
77404		value for an unspecified c flag shall be 9999.
77405	ffloor	Specify the lowest release (that is, floor), a number greater than 0 but less
77406		than 9999, which may be retrieved by a <i>get</i> command for editing. The
77407		default value for an unspecified f flag shall be 1.
77408	dSID	Specify the default delta number (SID) to be used by a <i>get</i> command.
77409	istr	Treat the “No ID keywords” message issued by <i>get</i> or <i>delta</i> as a fatal error.
77410		In the absence of this flag, the message is only a warning. The message is
77411		issued if no SCCS identification keywords (see <i>get</i>) are found in the text
77412		retrieved or stored in the SCCS file. If a value is supplied, the application
77413		shall ensure that the keywords exactly match the given string; however,
77414		the string shall contain a keyword, and no embedded <newline>
77415		characters.
77416	j	Allow concurrent <i>get</i> commands for editing on the same SID of an SCCS
77417		file. This allows multiple concurrent updates to the same version of the
77418		SCCS file.
77419	llist	Specify a <i>list</i> of releases to which deltas can no longer be made (that is, <i>get</i>
77420		-e against one of these locked releases fails). Conforming applications
77421		shall use the following syntax to specify a <i>list</i> . Implementations may
77422		accept additional forms as an extension:
77423		<list> ::= a <range-list>
77424		<range-list> ::= <range> <range-list>, <range>
77425		<range> ::= <SID>
77426		The character <i>a</i> in the <i>list</i> shall be equivalent to specifying all releases for
77427		the named SCCS file. The non-terminal <SID> in range shall be the delta
77428		number of an existing delta associated with the SCCS file.
77429	n	Cause <i>delta</i> to create a null delta in each of those releases (if any) being
77430		skipped when a delta is made in a new release (for example, in making
77431		delta 5.1 after delta 2.7, releases 3 and 4 are skipped). These null deltas
77432		shall serve as anchor points so that branch deltas may later be created
77433		from them. The absence of this flag shall cause skipped releases to be
77434		nonexistent in the SCCS file, preventing branch deltas from being created
77435		from them in the future. During the initial creation of an SCCS file, the n
77436		flag may be ignored; that is, if the -r option is used to set the release
77437		number of the initial SID to a value greater than 1, null deltas need not be
77438		created for the “skipped” releases.

77439	qtext	Substitute user-definable <i>text</i> for all occurrences of the %Q% keyword in the SCCS file text retrieved by <i>get</i> .
77440		
77441	mmod	Specify the module name of the SCCS file substituted for all occurrences of the %M% keyword in the SCCS file text retrieved by <i>get</i> . If the m flag is not specified, the value assigned shall be the name of the SCCS file with the leading ' .' removed.
77442		
77443		
77444		
77445	ttype	Specify the <i>type</i> of module in the SCCS file substituted for all occurrences of the %Y% keyword in the SCCS file text retrieved by <i>get</i> .
77446		
77447	vpgm	Cause <i>delta</i> to prompt for modification request (MR) numbers as the reason for creating a delta. The optional value specifies the name of an MR number validation program. (If this flag is set when creating an SCCS file, the application shall ensure that the m option is also used even if its value is null.)
77448		
77449		
77450		
77451		
77452	-d flag	Remove (delete) the specified <i>flag</i> from an SCCS file. Several -d options may be supplied on a single <i>admin</i> command. See the -f option for allowable <i>flag</i> names. (The l list flag gives a <i>list</i> of releases to be unlocked. See the -f option for further description of the l flag and the syntax of a <i>list</i> .)
77453		
77454		
77455		
77456	-a login	Specify a <i>login</i> name, or numerical group ID, to be added to the list of users who may make deltas (changes) to the SCCS file. A group ID shall be equivalent to specifying all <i>login</i> names common to that group ID. Several -a options may be used on a single <i>admin</i> command line. As many <i>logins</i> , or numerical group IDs, as desired may be on the list simultaneously. If the list of users is empty, then anyone may add deltas. If <i>login</i> or group ID is preceded by a '!', the users so specified shall be denied permission to make deltas.
77457		
77458		
77459		
77460		
77461		
77462		
77463	-e login	Specify a <i>login</i> name, or numerical group ID, to be erased from the list of users allowed to make deltas (changes) to the SCCS file. Specifying a group ID is equivalent to specifying all <i>login</i> names common to that group ID. Several -e options may be used on a single <i>admin</i> command line.
77464		
77465		
77466		
77467	-y[comment]	Insert the <i>comment</i> text into the SCCS file as a comment for the initial delta in a manner identical to that of <i>delta</i> . In the POSIX locale, omission of the -y option shall result in a default comment line being inserted in the form:
77468		
77469		
77470		"date and time created %s %s by %s", <date>, <time>, <login>
77471		where <date> is expressed in the format of the <i>date</i> utility's %Y/%m/%d conversion specification, <time> in the format of the <i>date</i> utility's %T conversion specification format, and <login> is the login name of the user creating the file.
77472		
77473		
77474	-m mrlist	Insert the list of modification request (MR) numbers into the SCCS file as the reason for creating the initial delta in a manner identical to <i>delta</i> . The application shall ensure that the v flag is set and the MR numbers are validated if the v flag has a value (the name of an MR number validation program). A diagnostic message shall be written if the v flag is not set or MR validation fails.
77475		
77476		
77477		
77478		
77479	-h	Check the structure of the SCCS file and compare the newly computed checksum with the checksum that is stored in the SCCS file. If the newly computed checksum does not match the checksum in the SCCS file, a diagnostic message shall be written.
77480		
77481		
77482		

77483 **-z** Recompute the SCCS file checksum and store it in the first line of the SCCS file (see
 77484 the **-h** option above). Note that use of this option on a truly corrupted file may
 77485 prevent future detection of the corruption.

77486 OPERANDS

77487 The following operands shall be supported:

77488 *file* A pathname of an existing SCCS file or a directory. If *file* is a directory, the *admin*
 77489 utility shall behave as though each file in the directory were specified as a named
 77490 file, except that non-SCCS files (last component of the pathname does not begin
 77491 with **s**.) and unreadable files shall be silently ignored.

77492 *newfile* A pathname of an SCCS file to be created.

77493 If exactly one *file* or *newfile* operand appears, and it is **'-'**, the standard input shall be read; each
 77494 line of the standard input shall be taken to be the name of an SCCS file to be processed. Non-
 77495 SCCS files and unreadable files shall be silently ignored.

77496 STDIN

77497 The standard input shall be a text file used only if **-i** is specified without an option-argument or
 77498 if a *file* or *newfile* operand is specified as **'-'**. If the first character of any standard input line is
 77499 <SOH> in the POSIX locale, the results are unspecified.

77500 INPUT FILES

77501 The existing SCCS files shall be text files of an unspecified format.

77502 The application shall ensure that the file named by the **-i** option's *name* option-argument shall
 77503 be a text file; if the first character of any line in this file is <SOH> in the POSIX locale, the results
 77504 are unspecified. If this file contains more than 99 999 lines, the number of lines recorded in the
 77505 header for this file shall be 99 999 for this delta.

77506 ENVIRONMENT VARIABLES

77507 The following environment variables shall affect the execution of *admin*:

77508 **LANG** Provide a default value for the internationalization variables that are unset or null.
 77509 (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization
 77510 variables used to determine the values of locale categories.)

77511 **LC_ALL** If set to a non-empty string value, override the values of all the other
 77512 internationalization variables.

77513 **LC_CTYPE** Determine the locale for the interpretation of sequences of bytes of text data as
 77514 characters (for example, single-byte as opposed to multi-byte characters in
 77515 arguments and input files).

77516 **LC_MESSAGES**

77517 Determine the locale that should be used to affect the format and contents of
 77518 diagnostic messages written to standard error and the contents of the default **-y**
 77519 comment.

77520 **NLSPATH** Determine the location of message catalogs for the processing of **LC_MESSAGES**.

77521 ASYNCHRONOUS EVENTS

77522 Default.

77523 STDOUT

77524 Not used.

77525 STDERR

77526 The standard error shall be used only for diagnostic messages.

77527 OUTPUT FILES

77528 Any SCCS files created shall be text files of an unspecified format. During processing of a *file*, a
77529 locking *z-file*, as described in *get* (on page 2788), may be created and deleted.

77530 EXTENDED DESCRIPTION

77531 None.

77532 EXIT STATUS

77533 The following exit values shall be returned:

77534 0 Successful completion.

77535 >0 An error occurred.

77536 CONSEQUENCES OF ERRORS

77537 Default.

77538 APPLICATION USAGE

77539 It is recommended that directories containing SCCS files be writable by the owner only, and that
77540 SCCS files themselves be read-only. The mode of the directories should allow only the owner to
77541 modify SCCS files contained in the directories. The mode of the SCCS files prevents any
77542 modification at all except by SCCS commands.

77543 EXAMPLES

77544 None.

77545 RATIONALE

77546 None.

77547 FUTURE DIRECTIONS

77548 None.

77549 SEE ALSO

77550 *delta, get, prs, what*

77551 XBD [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 215)

77552 CHANGE HISTORY

77553 First released in Issue 2.

77554 Issue 6

77555 The normative text is reworded to avoid use of the term “must” for application requirements,
77556 and to emphasize the term “shall” for implementation requirements.

77557 The grammar is updated.

77558 The Open Group Base Resolution bwg2001-007 is applied, adding new text to the INPUT FILES
77559 section warning that the maximum lines recorded in the file is 99 999.

77560 The Open Group Base Resolution bwg2001-009 is applied, amending the description of the **-h**
77561 option.

77562 **NAME**

77563 alias — define or display aliases

77564 **SYNOPSIS**77565 alias [*alias-name*[=*string*] . . .]77566 **DESCRIPTION**

77567 The *alias* utility shall create or redefine alias definitions or write the values of existing alias
 77568 definitions to standard output. An alias definition provides a string value that shall replace a
 77569 command name when it is encountered; see [Section 2.3.1](#) (on page 2322).

77570 An alias definition shall affect the current shell execution environment and the execution
 77571 environments of the subshells of the current shell. When used as specified by this volume of
 77572 POSIX.1-2008, the alias definition shall not affect the parent process of the current shell nor any
 77573 utility environment invoked by the shell; see [Section 2.12](#) (on page 2353).

77574 **OPTIONS**

77575 None.

77576 **OPERANDS**

77577 The following operands shall be supported:

77578 *alias-name* Write the alias definition to standard output.

77579 *alias-name=string*
 77580 Assign the value of *string* to the alias *alias-name*.

77581 If no operands are given, all alias definitions shall be written to standard output.

77582 **STDIN**

77583 Not used.

77584 **INPUT FILES**

77585 None.

77586 **ENVIRONMENT VARIABLES**77587 The following environment variables shall affect the execution of *alias*:

77588 *LANG* Provide a default value for the internationalization variables that are unset or null.
 77589 (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization
 77590 variables used to determine the values of locale categories.)

77591 *LC_ALL* If set to a non-empty string value, override the values of all the other
 77592 internationalization variables.

77593 *LC_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as
 77594 characters (for example, single-byte as opposed to multi-byte characters in
 77595 arguments).

77596 *LC_MESSAGES*
 77597 Determine the locale that should be used to affect the format and contents of
 77598 diagnostic messages written to standard error.

77599 XSI *NLSPATH* Determine the location of message catalogs for the processing of *LC_MESSAGES*.77600 **ASYNCHRONOUS EVENTS**

77601 Default.

STDOUT

The format for displaying aliases (when no operands or only *name* operands are specified) shall be:

```
"%s=%s\n", name, value
```

The *value* string shall be written with appropriate quoting so that it is suitable for reinput to the shell. See the description of shell quoting in [Section 2.2](#) (on page 2320).

STDERR

The standard error shall be used only for diagnostic messages.

OUTPUT FILES

None.

EXTENDED DESCRIPTION

None.

EXIT STATUS

The following exit values shall be returned:

0 Successful completion.

>0 One of the *name* operands specified did not have an alias definition, or an error occurred.

CONSEQUENCES OF ERRORS

Default.

APPLICATION USAGE

None.

EXAMPLES

1. Create a short alias for a commonly used *ls* command:

```
alias lf="ls -CF"
```

2. Create a simple “redo” command to repeat previous entries in the command history file:

```
alias r='fc -s'
```

3. Use 1K units for *du*:

```
alias du=du\ -k
```

4. Set up *nohup* so that it can deal with an argument that is itself an alias name:

```
alias nohup="nohup "
```

RATIONALE

The *alias* description is based on historical KornShell implementations. Known differences exist between that and the C shell. The KornShell version was adopted to be consistent with all the other KornShell features in this volume of POSIX.1-2008, such as command line editing.

Since *alias* affects the current shell execution environment, it is generally provided as a shell regular built-in.

Historical versions of the KornShell have allowed aliases to be exported to scripts that are invoked by the same shell. This is triggered by the *alias -x* flag; it is allowed by this volume of POSIX.1-2008 only when an explicit extension such as *-x* is used. The standard developers considered that aliases were of use primarily to interactive users and that they should normally not affect shell scripts called by those users; functions are available to such scripts.

77642 Historical versions of the KornShell had not written aliases in a quoted manner suitable for
77643 reentry to the shell, but this volume of POSIX.1-2008 has made this a requirement for all similar
77644 output. Therefore, consistency was chosen over this detail of historical practice.

77645 **FUTURE DIRECTIONS**

77646 None.

77647 **SEE ALSO**

77648 [Section 2.9.5](#) (on page 2346)

77649 XBD [Chapter 8](#) (on page 173)

77650 **CHANGE HISTORY**

77651 First released in Issue 4.

77652 **Issue 6**

77653 This utility is marked as part of the User Portability Utilities option.

77654 The APPLICATION USAGE section is added.

77655 **Issue 7**

77656 The *alias* utility is moved from the User Portability Utilities option to the Base. User Portability
77657 Utilities is now an option for interactive utilities.

77658 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

77659 The first example is changed to remove the creation of an alias for a standard utility that alters
77660 its behavior to be non-conforming.

77661 NAME

77662 ar — create and maintain library archives

77663 SYNOPSIS

77664 SD ar -d [-v] archive file...

77665 XSI ar -m [-v] archive file...

77666 ar -m -a [-v] posname archive file...

77667 ar -m -b [-v] posname archive file...

77668 ar -m -i [-v] posname archive file...

77669 XSI ar -p [-v] [-s] archive [file...]

77670 XSI ar -q [-cv] archive file...

77671 ar -r [-cuv] archive file...

77672 XSI ar -r -a [-cuv] posname archive file...

77673 ar -r -b [-cuv] posname archive file...

77674 ar -r -i [-cuv] posname archive file...

77675 XSI ar -t [-v] [-s] archive [file...]

77676 XSI ar -x [-v] [-sCT] archive [file...]

77677 DESCRIPTION

77678 The *ar* utility is part of the Software Development Utilities option.

77679 The *ar* utility can be used to create and maintain groups of files combined into an archive. Once
 77680 an archive has been created, new files can be added, and existing files in an archive can be
 77681 extracted, deleted, or replaced. When an archive consists entirely of valid object files, the
 77682 implementation shall format the archive so that it is usable as a library for link editing (see *c99*
 77683 and *fort77*). When some of the archived files are not valid object files, the suitability of the
 77684 XSI archive for library use is undefined. If an archive consists entirely of printable files, the entire
 77685 archive shall be printable.

77686 When *ar* creates an archive, it creates administrative information indicating whether a symbol
 77687 table is present in the archive. When there is at least one object file that *ar* recognizes as such in
 77688 the archive, an archive symbol table shall be created in the archive and maintained by *ar*; it is
 77689 used by the link editor to search the archive. Whenever the *ar* utility is used to create or update
 77690 the contents of such an archive, the symbol table shall be rebuilt. The *-s* option shall force the
 77691 symbol table to be rebuilt.

77692 All *file* operands can be pathnames. However, files within archives shall be named by a filename,
 77693 which is the last component of the pathname used when the file was entered into the archive.
 77694 The comparison of *file* operands to the names of files in archives shall be performed by
 77695 comparing the last component of the operand to the name of the file in the archive.

77696 It is unspecified whether multiple files in the archive may be identically named. In the case of
 77697 XSI such files, however, each *file* and *posname* operand shall match only the first file in the archive
 77698 having a name that is the same as the last component of the operand.

77699 OPTIONS

77700 The *ar* utility shall conform to XBD [Section 12.2](#) (on page 215), except for Guideline 9.

77701 The following options shall be supported:

77702	XSI	-a	Position new files in the archive after the file named by the <i>posname</i> operand.
77703	XSI	-b	Position new files in the archive before the file named by the <i>posname</i> operand.
77704		-c	Suppress the diagnostic message that is written to standard error by default when the archive <i>archive</i> is created.
77706	XSI	-C	Prevent extracted files from replacing like-named files in the file system. This option is useful when -T is also used, to prevent truncated filenames from replacing files with the same prefix.
77708			
77709		-d	Delete one or more <i>files</i> from <i>archive</i> .
77710	XSI	-i	Position new files in the archive before the file in the archive named by the <i>posname</i> operand (equivalent to -b).
77711			
77712	XSI	-m	Move the named files in the archive. The -a , -b , or -i options with the <i>posname</i> operand indicate the position; otherwise, move the names files in the archive to the end of the archive.
77713			
77714			
77715		-p	Write the contents of the <i>files</i> in the archive named by <i>file</i> operands from <i>archive</i> to the standard output. If no <i>file</i> operands are specified, the contents of all files in the archive shall be written in the order of the archive.
77716			
77717			
77718	XSI	-q	Append the named files to the end of the archive. In this case <i>ar</i> does not check whether the added files are already in the archive. This is useful to bypass the searching otherwise done when creating a large archive piece by piece.
77719			
77720			
77721		-r	Replace or add <i>files</i> to <i>archive</i> . If the archive named by <i>archive</i> does not exist, a new archive shall be created and a diagnostic message shall be written to standard error (unless the -c option is specified). If no <i>files</i> are specified and the <i>archive</i> exists, the results are undefined. Files that replace existing files in the archive shall not change the order of the archive. Files that do not replace existing files in the archive shall be appended to the archive unless a -a , -b , or -i option specifies another position.
77722			
77723			
77724			
77725			
77726	XSI		
77727	XSI	-s	Force the regeneration of the archive symbol table even if <i>ar</i> is not invoked with an option that modifies the archive contents. This option is useful to restore the archive symbol table after it has been stripped; see <i>strip</i> .
77728			
77729			
77730		-t	Write a table of contents of <i>archive</i> to the standard output. Only the files specified by the <i>file</i> operands shall be included in the written list. If no <i>file</i> operands are specified, all files in <i>archive</i> shall be included in the order of the archive.
77731			
77732			
77733	XSI	-T	Allow filename truncation of extracted files whose archive names are longer than the file system can support. By default, extracting a file with a name that is too long shall be an error; a diagnostic message shall be written and the file shall not be extracted.
77734			
77735			
77736			
77737		-u	Update older files in the archive. When used with the -r option, files in the archive shall be replaced only if the corresponding <i>file</i> has a modification time that is at least as new as the modification time of the file in the archive.
77738			
77739			

77740 **-v** Give verbose output. When used with the option characters **-d**, **-r**, or **-x**, write a
 77741 detailed file-by-file description of the archive creation and maintenance activity, as
 77742 described in the STDOUT section.

77743 When used with **-p**, write the name of the file in the archive to the standard output
 77744 before writing the file in the archive itself to the standard output, as described in
 77745 the STDOUT section.

77746 When used with **-t**, include a long listing of information about the files in the
 77747 archive, as described in the STDOUT section.

77748 **-x** Extract the files in the archive named by the *file* operands from *archive*. The
 77749 contents of the archive shall not be changed. If no *file* operands are given, all files
 77750 in the archive shall be extracted. The modification time of each file extracted shall
 77751 be set to the time the file is extracted from the archive.

77752 OPERANDS

77753 The following operands shall be supported:

77754 *archive* A pathname of the archive.

77755 *file* A pathname. Only the last component shall be used when comparing against the
 77756 names of files in the archive. If two or more *file* operands have the same last
 77757 pathname component (basename), the results are unspecified. The
 77758 implementation's archive format shall not truncate valid filenames of files added
 77759 to or replaced in the archive.

77760 XSI *posname* The name of a file in the archive, used for relative positioning; see options **-m** and
 77761 **-r**.

77762 STDIN

77763 Not used.

77764 INPUT FILES

77765 The archive named by *archive* shall be a file in the format created by *ar -r*.

77766 ENVIRONMENT VARIABLES

77767 The following environment variables shall affect the execution of *ar*:

77768 *LANG* Provide a default value for the internationalization variables that are unset or null.
 77769 (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization
 77770 variables used to determine the values of locale categories.)

77771 *LC_ALL* If set to a non-empty string value, override the values of all the other
 77772 internationalization variables.

77773 *LC_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as
 77774 characters (for example, single-byte as opposed to multi-byte characters in
 77775 arguments and input files).

77776 *LC_MESSAGES*

77777 Determine the locale that should be used to affect the format and contents of
 77778 diagnostic messages written to standard error.

77779 *LC_TIME* Determine the format and content for date and time strings written by *ar -tv*.

77780 XSI *NLSPATH* Determine the location of message catalogs for the processing of *LC_MESSAGES*.

77781 *TMPPDIR* Determine the pathname that overrides the default directory for temporary files, if
77782 any.

77783 *TZ* Determine the timezone used to calculate date and time strings written by *ar -tv*.
77784 If *TZ* is unset or null, an unspecified default timezone shall be used.

77785 ASYNCHRONOUS EVENTS

77786 Default.

77787 STDOUT

77788 If the *-d* option is used with the *-v* option, the standard output format shall be:

77789 "d - %s\n", <file>

77790 where *file* is the operand specified on the command line.

77791 If the *-p* option is used with the *-v* option, *ar* shall precede the contents of each file with:

77792 "\n<%s>\n\n", <file>

77793 where *file* is the operand specified on the command line, if *file* operands were specified, and the
77794 name of the file in the archive if they were not.

77795 If the *-r* option is used with the *-v* option:

- 77796 • If *file* is already in the archive, the standard output format shall be:

77797 "r - %s\n", <file>

77798 where <file> is the operand specified on the command line.

- 77799 • If *file* is not already in the archive, the standard output format shall be:

77800 "a - %s\n", <file>

77801 where <file> is the operand specified on the command line.

77802 If the *-t* option is used, *ar* shall write the names of the files in the archive to the standard output
77803 in the format:

77804 "%s\n", <file>

77805 where *file* is the operand specified on the command line, if *file* operands were specified, or the
77806 name of the file in the archive if they were not.

77807 If the *-t* option is used with the *-v* option, the standard output format shall be:

77808 "%s %u/%u %u %s %d %d:%d %d %s\n", <member mode>, <user ID>,
77809 <group ID>, <number of bytes in member>,
77810 <abbreviated month>, <day-of-month>, <hour>,
77811 <minute>, <year>, <file>

77812 where:

77813 <file> Shall be the operand specified on the command line, if *file* operands were specified,
77814 or the name of the file in the archive if they were not.

77815 <member mode>

77816 Shall be formatted the same as the <file mode> string defined in the STDOUT
77817 section of *ls*, except that the first character, the <entry type>, is not used; the string
77818 represents the file mode of the file in the archive at the time it was added to or
77819 replaced in the archive.

77820 The following represent the last-modification time of a file when it was most recently added to
77821 or replaced in the archive:

77822 *<abbreviated month>*

77823 Equivalent to the format of the %b conversion specification format in *date*.

77824 *<day-of-month>*

77825 Equivalent to the format of the %e conversion specification format in *date*.

77826 *<hour>* Equivalent to the format of the %H conversion specification format in *date*.

77827 *<minute>* Equivalent to the format of the %M conversion specification format in *date*.

77828 *<year>* Equivalent to the format of the %Y conversion specification format in *date*.

77829 When *LC_TIME* does not specify the POSIX locale, a different format and order of presentation
77830 of these fields relative to each other may be used in a format appropriate in the specified locale.

77831 If the *-x* option is used with the *-v* option, the standard output format shall be:

77832 "x - %s\n", *<file>*

77833 where *file* is the operand specified on the command line, if *file* operands were specified, or the
77834 name of the file in the archive if they were not.

77835 **STDERR**

77836 The standard error shall be used only for diagnostic messages. The diagnostic message about
77837 creating a new archive when *-c* is not specified shall not modify the exit status.

77838 **OUTPUT FILES**

77839 Archives are files with unspecified formats.

77840 **EXTENDED DESCRIPTION**

77841 None.

77842 **EXIT STATUS**

77843 The following exit values shall be returned:

77844 0 Successful completion.

77845 >0 An error occurred.

77846 **CONSEQUENCES OF ERRORS**

77847 Default.

77848 **APPLICATION USAGE**

77849 None.

77850 **EXAMPLES**

77851 None.

77852 **RATIONALE**

77853 The archive format is not described. It is recognized that there are several known *ar* formats,
77854 which are not compatible. The *ar* utility is included, however, to allow creation of archives that
77855 are intended for use only on one machine. The archive is specified as a file, and it can be moved
77856 as a file. This does allow an archive to be moved from one machine to another machine that uses
77857 the same implementation of *ar*.

77858 Utilities such as *pax* (and its forebears *tar* and *cpio*) also provide portable “archives”. This is a not
77859 a duplication; the *ar* utility is included to provide an interface primarily for *make* and the
77860 compilers, based on a historical model.

In historical implementations, the **-q** option (available on XSI-conforming systems) is known to execute quickly because *ar* does not check on whether the added members are already in the archive. This is useful to bypass the searching otherwise done when creating a large archive piece-by-piece. These remarks may but need not remain true for a brand new implementation of this utility; hence, these remarks have been moved into the RATIONALE.

BSD implementations historically required applications to provide the **-s** option whenever the archive was supposed to contain a symbol table. As in this volume of POSIX.1-2008, System V historically creates or updates an archive symbol table whenever an object file is removed from, added to, or updated in the archive.

The OPERANDS section requires what might seem to be true without specifying it: the archive cannot truncate the filenames below [NAME_MAX]. Some historical implementations do so, however, causing unexpected results for the application. Therefore, this volume of POSIX.1-2008 makes the requirement explicit to avoid misunderstandings.

According to the System V documentation, the options **-dmpqrtx** are not required to begin with a <hyphen> ('-'). This volume of POSIX.1-2008 requires that a conforming application use the leading <hyphen>.

The archive format used by the 4.4 BSD implementation is documented in this RATIONALE as an example:

A file created by *ar* begins with the “magic” string “!**<arch>**\n”. The rest of the archive is made up of objects, each of which is composed of a header for a file, a possible filename, and the file contents. The header is portable between machine architectures, and, if the file contents are printable, the archive is itself printable.

The header is made up of six ASCII fields, followed by a two-character trailer. The fields are the object name (16 characters), the file last modification time (12 characters), the user and group IDs (each 6 characters), the file mode (8 characters), and the file size (10 characters). All numeric fields are in decimal, except for the file mode, which is in octal.

The modification time is the file *st_mtime* field. The user and group IDs are the file *st_uid* and *st_gid* fields. The file mode is the file *st_mode* field. The file size is the file *st_size* field. The two-byte trailer is the string “**<newline>**”.

Only the name field has any provision for overflow. If any filename is more than 16 characters in length or contains an embedded space, the string “**#1/**” followed by the ASCII length of the name is written in the name field. The file size (stored in the archive header) is incremented by the length of the name. The name is then written immediately following the archive header.

Any unused characters in any of these fields are written as <space> characters. If any fields are their particular maximum number of characters in length, there is no separation between the fields.

Objects in the archive are always an even number of bytes long; files that are an odd number of bytes long are padded with a <newline>, although the size in the header does not reflect this.

The *ar* utility description requires that (when all its members are valid object files) *ar* produce an object code library, which the linkage editor can use to extract object modules. If the linkage editor needs a symbol table to permit random access to the archive, *ar* must provide it; however, *ar* does not require a symbol table.

The BSD **-o** option was omitted. It is a rare conforming application that uses *ar* to extract object

code from a library with concern for its modification time, since this can only be of importance to *make*. Hence, since this functionality is not deemed important for applications portability, the modification time of the extracted files is set to the current time.

There is at least one known implementation (for a small computer) that can accommodate only object files for that system, disallowing mixed object and other files. The ability to handle any type of file is not only historical practice for most implementations, but is also a reasonable expectation.

Consideration was given to changing the output format of *ar -tv* to the same format as the output of *ls -l*. This would have made parsing the output of *ar* the same as that of *ls*. This was rejected in part because the current *ar* format is commonly used and changes would break historical usage. Second, *ar* gives the user ID and group ID in numeric format separated by a `<slash>`. Changing this to be the user name and group name would not be correct if the archive were moved to a machine that contained a different user database. Since *ar* cannot know whether the archive was generated on the same machine, it cannot tell what to report.

The text on the *-ur* option combination is historical practice—since one filename can easily represent two different files (for example, */a/foo* and */b/foo*), it is reasonable to replace the file in the archive even when the modification time in the archive is identical to that in the file system.

FUTURE DIRECTIONS

None.

SEE ALSO

c99, *date*, *fort77*, *pax*, *strip*

XBD [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 215), [<unistd.h>](#), description of `{POSIX_NO_TRUNC}`

CHANGE HISTORY

First released in Issue 2.

Issue 5

The FUTURE DIRECTIONS section is added.

Issue 6

This utility is marked as part of the Software Development Utilities option.

The STDOUT description is changed for the *-v* option to align with the IEEE P1003.2b draft standard.

The normative text is reworded to avoid use of the term “must” for application requirements.

The *TZ* entry is added to the ENVIRONMENT VARIABLES section.

IEEE PASC Interpretation 1003.2 #198 is applied, changing the description to consistently use “file” to refer to a file in the file system hierarchy, “archive” to refer to the archive being operated upon by the *ar* utility, and “file in the archive” to refer to a copy of a file that is contained in the archive.

IEEE Std 1003.1-2001/Cor 1-2002, item XCU/TC1/D6/10 is applied, making corrections to the SYNOPSIS. The change was needed since the *-a*, *-b*, and *-i* options are mutually-exclusive, and *posname* is required if any of these options is specified.

IEEE Std 1003.1-2001/Cor 1-2002, item XCU/TC1/D6/11 is applied, correcting the description of the two-byte trailer in RATIONALE which had missed out a backquote. The correct trailer is a backquote followed by a `<newline>`.

77949 **Issue 7**

77950 SD5-XCU-ERN-6 is applied, clarifying that Guideline 9 of the Utility Syntax Guidelines does not
77951 apply.

77952 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

77953 The description of the `-t` option is changed to say “Only the files specified ...”.

77954 **NAME**

77955 asa — interpret carriage-control characters

77956 **SYNOPSIS**77957 FR asa [*file...*]77958 **DESCRIPTION**77959 The *asa* utility shall write its input files to standard output, mapping carriage-control characters
77960 from the text files to line-printer control sequences in an implementation-defined manner.77961 The first character of every line shall be removed from the input, and the following actions are
77962 performed.

77963 If the character removed is:

77964 <space> The rest of the line is output without change.

77965 0 A <newline> is output, then the rest of the input line.

77966 1 One or more implementation-defined characters that causes an advance to the next
77967 page shall be output, followed by the rest of the input line.77968 + The <newline> of the previous line shall be replaced with one or more
77969 implementation-defined characters that causes printing to return to column
77970 position 1, followed by the rest of the input line. If the '+' is the first character in
77971 the input, it shall be equivalent to <space>.77972 The action of the *asa* utility is unspecified upon encountering any character other than those
77973 listed above as the first character in a line.77974 **OPTIONS**

77975 None.

77976 **OPERANDS**77977 *file* A pathname of a text file used for input. If no *file* operands are specified, the
77978 standard input shall be used.77979 **STDIN**77980 The standard input shall be used if no *file* operands are specified, and shall be used if a *file*
77981 operand is '-' and the implementation treats the '-' as meaning standard input. Otherwise,
77982 the standard input shall not be used. See the INPUT FILES section.77983 **INPUT FILES**

77984 The input files shall be text files.

77985 **ENVIRONMENT VARIABLES**77986 The following environment variables shall affect the execution of *asa*:77987 LANG Provide a default value for the internationalization variables that are unset or null.
77988 (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization
77989 variables used to determine the values of locale categories.)77990 LC_ALL If set to a non-empty string value, override the values of all the other
77991 internationalization variables.77992 LC_CTYPE Determine the locale for the interpretation of sequences of bytes of text data as
77993 characters (for example, single-byte as opposed to multi-byte characters in
77994 arguments and input files).

77995 **LC_MESSAGES**
 77996 Determine the locale that should be used to affect the format and contents of
 77997 diagnostic messages written to standard error.

77998 XSI **NLSPATH** Determine the location of message catalogs for the processing of *LC_MESSAGES*.

77999 **ASYNCHRONOUS EVENTS**
 78000 Default.

78001 **STDOUT**
 78002 The standard output shall be the text from the input file modified as described in the
 78003 DESCRIPTION section.

78004 **STDERR**
 78005 None.

78006 **OUTPUT FILES**
 78007 None.

78008 **EXTENDED DESCRIPTION**
 78009 None.

78010 **EXIT STATUS**
 78011 The following exit values shall be returned:
 78012 0 All input files were output successfully.
 78013 >0 An error occurred.

78014 **CONSEQUENCES OF ERRORS**
 78015 Default.

78016 **APPLICATION USAGE**
 78017 None.

78018 **EXAMPLES**

78019 1. The following command:
 78020 *asa file*
 78021 permits the viewing of *file* (created by a program using FORTRAN-style carriage-control
 78022 characters) on a terminal.

78023 2. The following command:
 78024 *a.out | asa | lp*
 78025 formats the FORTRAN output of **a.out** and directs it to the printer.

78026 **RATIONALE**
 78027 The *asa* utility is needed to map “standard” FORTRAN 77 output into a form acceptable to
 78028 contemporary printers. Usually, *asa* is used to pipe data to the *lp* utility; see *lp*.

78029 This utility is generally used only by FORTRAN programs. The standard developers decided to
 78030 retain *asa* to avoid breaking the historical large base of FORTRAN applications that put carriage-
 78031 control characters in their output files. There is no requirement that a system have a FORTRAN
 78032 compiler in order to run applications that need *asa*.

78033 Historical implementations have used an ASCII <form-feed> in response to a 1 and an ASCII
 78034 <carriage-return> in response to a '+'. It is suggested that implementations treat characters
 78035 other than 0, 1, and '+' as <space> in the absence of any compelling reason to do otherwise.

78036 However, the action is listed here as “unspecified”, permitting an implementation to provide
78037 extensions to access fast multiple-line slewing and channel seeking in a non-portable manner.

78038 **FUTURE DIRECTIONS**

78039 None.

78040 **SEE ALSO**

78041 *fort77, lp*

78042 XBD [Chapter 8](#) (on page 173)

78043 **CHANGE HISTORY**

78044 First released in Issue 4.

78045 **Issue 6**

78046 This utility is marked as part of the FORTRAN Runtime Utilities option.

78047 The normative text is reworded to avoid use of the term “must” for application requirements.

78048 **Issue 7**

78049 Austin Group Interpretation 1003.1-2001 #092 is applied.

78050 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

78051 NAME

78052 at — execute commands at a later time

78053 SYNOPSIS

78054 at [-m] [-f file] [-q queueName] -t time_arg

78055 at [-m] [-f file] [-q queueName] timespec...

78056 at -r at_job_id...

78057 at -l -q queueName

78058 at -l [at_job_id...]

78059 DESCRIPTION

78060 The *at* utility shall read commands from standard input and group them together as an *at-job*, to
78061 be executed at a later time.78062 The *at-job* shall be executed in a separate invocation of the shell, running in a separate process
78063 group with no controlling terminal, except that the environment variables, current working
78064 directory, file creation mask, and other implementation-defined execution-time attributes in
78065 effect when the *at* utility is executed shall be retained and used when the *at-job* is executed.78066 When the *at-job* is submitted, the *at_job_id* and scheduled time shall be written to standard error.
78067 The *at_job_id* is an identifier that shall be a string consisting solely of alphanumeric characters
78068 and the <period> character. The *at_job_id* shall be assigned by the system when the job is
78069 scheduled such that it uniquely identifies a particular job.78070 User notification and the processing of the job's standard output and standard error are
78071 described under the **-m** option.78072 XSI Users shall be permitted to use *at* if their name appears in the file **at.allow** which is located in an
78073 implementation-defined directory. If that file does not exist, the file **at.deny**, which is located in
78074 an implementation-defined directory, shall be checked to determine whether the user shall be
78075 denied access to *at*. If neither file exists, only a process with appropriate privileges shall be
78076 allowed to submit a job. If only **at.deny** exists and is empty, global usage shall be permitted. The
78077 **at.allow** and **at.deny** files shall consist of one user name per line.

78078 OPTIONS

78079 The *at* utility shall conform to XBD [Section 12.2](#) (on page 215).

78080 The following options shall be supported:

78081 **-f file** Specify the pathname of a file to be used as the source of the *at-job*, instead of
78082 standard input.78083 **-l** (The letter ell.) Report all jobs scheduled for the invoking user if no *at_job_id*
78084 operands are specified. If *at_job_ids* are specified, report only information for these
78085 jobs. The output shall be written to standard output.78086 **-m** Send mail to the invoking user after the *at-job* has run, announcing its completion.
78087 Standard output and standard error produced by the *at-job* shall be mailed to the
78088 user as well, unless redirected elsewhere. Mail shall be sent even if the job
78089 produces no output.78090 If **-m** is not used, the job's standard output and standard error shall be provided to
78091 the user by means of mail, unless they are redirected elsewhere; if there is no such
78092 output to provide, the implementation need not notify the user of the job's
78093 completion.

78094	-q <i>queuename</i>	
78095		Specify in which queue to schedule a job for submission. When used with the -l
78096		option, limit the search to that particular queue. By default, at-jobs shall be
78097		scheduled in queue <i>a</i> . In contrast, queue <i>b</i> shall be reserved for batch jobs; see
78098		<i>batch</i> . The meanings of all other <i>queuenames</i> are implementation-defined. If -q is
78099		specified along with either of the -t <i>time_arg</i> or <i>timespec</i> arguments, the results are
78100		unspecified.
78101	-r	Remove the jobs with the specified <i>at_job_id</i> operands that were previously
78102		scheduled by the <i>at</i> utility.
78103	-t <i>time_arg</i>	Submit the job to be run at the time specified by the <i>time</i> option-argument, which
78104		the application shall ensure has the format as specified by the <i>touch -t time</i> utility.

OPERANDS

78105	The following operands shall be supported:	
78106		
78107	<i>at_job_id</i>	The name reported by a previous invocation of the <i>at</i> utility at the time the job was
78108		scheduled.
78109	<i>timespec</i>	Submit the job to be run at the date and time specified. All of the <i>timespec</i> operands
78110		are interpreted as if they were separated by <space> characters and concatenated,
78111		and shall be parsed as described in the grammar at the end of this section. The date
78112		and time shall be interpreted as being in the timezone of the user (as determined
78113		by the <i>TZ</i> variable), unless a timezone name appears as part of <i>time</i> , below.
78114		In the POSIX locale, the following describes the three parts of the time specification
78115		string. All of the values from the <i>LC_TIME</i> categories in the POSIX locale shall be
78116		recognized in a case-insensitive manner.
78117	<i>time</i>	The time can be specified as one, two, or four digits. One-digit and
78118		two-digit numbers shall be taken to be hours; four-digit numbers to
78119		be hours and minutes. The time can alternatively be specified as two
78120		numbers separated by a <colon>, meaning <i>hour:minute</i> . An AM/PM
78121		indication (one of the values from the am_pm keywords in the
78122		<i>LC_TIME</i> locale category) can follow the time; otherwise, a 24-hour
78123		clock time shall be understood. A timezone name can also follow to
78124		further qualify the time. The acceptable timezone names are
78125		implementation-defined, except that they shall be case-insensitive
78126		and the string utc is supported to indicate the time is in Coordinated
78127		Universal Time. In the POSIX locale, the <i>time</i> field can also be one of
78128		the following tokens:
78129	midnight	Indicates the time 12:00 am (00:00).
78130	noon	Indicates the time 12:00 pm.
78131	now	Indicates the current day and time. Invoking <i>at <now></i>
78132		shall submit an at-job for potentially immediate
78133		execution (that is, subject only to unspecified
78134		scheduling delays).
78135	<i>date</i>	An optional <i>date</i> can be specified as either a month name (one of the
78136		values from the mon or abmon keywords in the <i>LC_TIME</i> locale
78137		category) followed by a day number (and possibly year number
78138		preceded by a comma), or a day of the week (one of the values from
78139		the day or abday keywords in the <i>LC_TIME</i> locale category). In the

78140 POSIX locale, two special days shall be recognized:

78141 **today** Indicates the current day.

78142 **tomorrow** Indicates the day following the current day.

78143 If no *date* is given, **today** shall be assumed if the given time is greater
 78144 than the current time, and **tomorrow** shall be assumed if it is less. If
 78145 the given month is less than the current month (and no year is given),
 78146 next year shall be assumed.

78147 *increment* The optional *increment* shall be a number preceded by a <plus-sign>
 78148 ('+') and suffixed by one of the following: **minutes**, **hours**, **days**,
 78149 **weeks**, **months**, or **years**. (The singular forms shall also be accepted.)
 78150 The keyword **next** shall be equivalent to an increment number of +1.
 78151 For example, the following are equivalent commands:

78152 at 2pm + 1 week
 78153 at 2pm next week

78154 The following grammar describes the precise format of *timespec* in the POSIX locale. The general
 78155 conventions for this style of grammar are described in [Section 1.3](#) (on page 2309). This formal
 78156 syntax shall take precedence over the preceding text syntax description. The longest possible
 78157 token or delimiter shall be recognized at a given point. When used in a *timespec*, white space
 78158 shall also delimit tokens.

```

78159 %token hr24clock_hr_min
78160 %token hr24clock_hour
78161 /*
78162     An hr24clock_hr_min is a one, two, or four-digit number. A one-digit
78163     or two-digit number constitutes an hr24clock_hour. An hr24clock_hour
78164     may be any of the single digits [0,9], or may be double digits, ranging
78165     from [00,23]. If an hr24clock_hr_min is a four-digit number, the
78166     first two digits shall be a valid hr24clock_hour, while the last two
78167     represent the number of minutes, from [00,59].
78168 */
78169 %token wallclock_hr_min
78170 %token wallclock_hour
78171 /*
78172     A wallclock_hr_min is a one, two-digit, or four-digit number.
78173     A one-digit or two-digit number constitutes a wallclock_hour.
78174     A wallclock_hour may be any of the single digits [1,9], or may
78175     be double digits, ranging from [01,12]. If a wallclock_hr_min
78176     is a four-digit number, the first two digits shall be a valid
78177     wallclock_hour, while the last two represent the number of
78178     minutes, from [00,59].
78179 */
78180 %token minute
78181 /*
78182     A minute is a one or two-digit number whose value can be [0,9]
78183     or [00,59].
78184 */
78185 %token day_number
78186 /*
```

```

78187      A day_number is a number in the range appropriate for the particular
78188      month and year specified by month_name and year_number, respectively.
78189      If no year_number is given, the current year is assumed if the given
78190      date and time are later this year. If no year_number is given and
78191      the date and time have already occurred this year and the month is
78192      not the current month, next year is the assumed year.
78193      */

78194      %token year_number
78195      /*
78196      A year_number is a four-digit number representing the year A.D., in
78197      which the at_job is to be run.
78198      */

78199      %token inc_number
78200      /*
78201      The inc_number is the number of times the succeeding increment
78202      period is to be added to the specified date and time.
78203      */

78204      %token timezone_name
78205      /*
78206      The name of an optional timezone suffix to the time field, in an
78207      implementation-defined format.
78208      */

78209      %token month_name
78210      /*
78211      One of the values from the mon or abmon keywords in the LC_TIME
78212      locale category.
78213      */

78214      %token day_of_week
78215      /*
78216      One of the values from the day or abday keywords in the LC_TIME
78217      locale category.
78218      */

78219      %token am_pm
78220      /*
78221      One of the values from the am_pm keyword in the LC_TIME locale
78222      category.
78223      */

78224      %start timespec
78225      %%
78226      timespec      : time
78227                    | time date
78228                    | time increment
78229                    | time date increment
78230                    | nowspec
78231                    ;
78232      nowspec       : "now"
78233                    | "now" increment
78234                    ;

```

```

78235         time          : hr24clock_hr_min
78236                       | hr24clock_hr_min timezone_name
78237                       | hr24clock_hour ":" minute
78238                       | hr24clock_hour ":" minute timezone_name
78239                       | wallclock_hr_min am_pm
78240                       | wallclock_hr_min am_pm timezone_name
78241                       | wallclock_hour ":" minute am_pm
78242                       | wallclock_hour ":" minute am_pm timezone_name
78243                       | "noon"
78244                       | "midnight"
78245                       ;

78246         date           : month_name day_number
78247                       | month_name day_number "," year_number
78248                       | day_of_week
78249                       | "today"
78250                       | "tomorrow"
78251                       ;

78252         increment      : "+" inc_number inc_period
78253                       | "next" inc_period
78254                       ;

78255         inc_period     : "minute" | "minutes"
78256                       | "hour" | "hours"
78257                       | "day" | "days"
78258                       | "week" | "weeks"
78259                       | "month" | "months"
78260                       | "year" | "years"
78261                       ;

```

78262 STDIN

78263 The standard input shall be a text file consisting of commands acceptable to the shell command
 78264 language described in [Chapter 2](#) (on page 2319). The standard input shall only be used if no `-f`
 78265 *file* option is specified.

78266 INPUT FILES

78267 See the STDIN section.

78268 XSI The text files **at.allow** and **at.deny**, which are located in an implementation-defined directory,
 78269 shall contain zero or more user names, one per line, of users who are, respectively, authorized or
 78270 denied access to the *at* and *batch* utilities.

78271 ENVIRONMENT VARIABLES

78272 The following environment variables shall affect the execution of *at*:

78273 78274 78275	LANG	Provide a default value for the internationalization variables that are unset or null. (See XBD Section 8.2 (on page 174) for the precedence of internationalization variables used to determine the values of locale categories.)
78276 78277	LC_ALL	If set to a non-empty string value, override the values of all the other internationalization variables.
78278 78279 78280	LC_CTYPE	Determine the locale for the interpretation of sequences of bytes of text data as characters (for example, single-byte as opposed to multi-byte characters in arguments and input files).

78281		<i>LC_MESSAGES</i>	
78282			Determine the locale that should be used to affect the format and contents of
78283			diagnostic messages written to standard error and informative messages written to
78284			standard output.
78285	XSI	<i>NLSPATH</i>	Determine the location of message catalogs for the processing of <i>LC_MESSAGES</i> .
78286		<i>LC_TIME</i>	Determine the format and contents for date and time strings written and accepted
78287			by <i>at</i> .
78288		<i>SHELL</i>	Determine a name of a command interpreter to be used to invoke the at-job. If the
78289			variable is unset or null, <i>sh</i> shall be used. If it is set to a value other than a name for
78290			<i>sh</i> , the implementation shall do one of the following: use that shell; use <i>sh</i> ; use the
78291			login shell from the user database; or any of the preceding accompanied by a
78292			warning diagnostic about which was chosen.
78293		<i>TZ</i>	Determine the timezone. The job shall be submitted for execution at the time
78294			specified by <i>timespec</i> or <i>-t time</i> relative to the timezone specified by the <i>TZ</i>
78295			variable. If <i>timespec</i> specifies a timezone, it shall override <i>TZ</i> . If <i>timespec</i> does not
78296			specify a timezone and <i>TZ</i> is unset or null, an unspecified default timezone shall
78297			be used.
78298		ASYNCHRONOUS EVENTS	
78299			Default.
78300		STDOUT	
78301			When standard input is a terminal, prompts of unspecified format for each line of the user input
78302			described in the STDIN section may be written to standard output.
78303			In the POSIX locale, the following shall be written to the standard output for each job when jobs
78304			are listed in response to the <i>-l</i> option:
78305			"%s\t%s\n", <i>at_job_id</i> , < <i>date</i> >
78306			where <i>date</i> shall be equivalent in format to the output of:
78307			date +"%a %b %e %T %Y"
78308			The date and time written shall be adjusted so that they appear in the timezone of the user (as
78309			determined by the <i>TZ</i> variable).
78310		STDERR	
78311			In the POSIX locale, the following shall be written to standard error when a job has been
78312			successfully submitted:
78313			"job %s at %s\n", <i>at_job_id</i> , < <i>date</i> >
78314			where <i>date</i> has the same format as that described in the STDOUT section. Neither this, nor
78315			warning messages concerning the selection of the command interpreter, shall be considered a
78316			diagnostic that changes the exit status.
78317			Diagnostic messages, if any, shall be written to standard error.
78318		OUTPUT FILES	
78319			None.
78320		EXTENDED DESCRIPTION	
78321			None.

EXIT STATUS

The following exit values shall be returned:

- 0 The *at* utility successfully submitted, removed, or listed a job or jobs.
- >0 An error occurred.

CONSEQUENCES OF ERRORS

The job shall not be scheduled, removed, or listed.

APPLICATION USAGE

The format of the *at* command line shown here is guaranteed only for the POSIX locale. Other cultures may be supported with substantially different interfaces, although implementations are encouraged to provide comparable levels of functionality.

Since the commands run in a separate shell invocation, running in a separate process group with no controlling terminal, open file descriptors, traps, and priority inherited from the invoking environment are lost.

Some implementations do not allow substitution of different shells using *SHELL*. System V systems, for example, have used the login shell value for the user in */etc/passwd*. To select reliably another command interpreter, the user must include it as part of the script, such as:

```
$ at 1800
myshell myscript
EOT
job ... at ...
$
```

EXAMPLES

1. This sequence can be used at a terminal:

```
at -m 0730 tomorrow
sort < file >outfile
EOT
```

2. This sequence, which demonstrates redirecting standard error to a pipe, is useful in a command procedure (the sequence of output redirection specifications is significant):

```
at now + 1 hour <<!
diff file1 file2 2>&1 >outfile | mailx mygroup
!
```

3. To have a job reschedule itself, *at* can be invoked from within the at-job. For example, this daily processing script named **my.daily** runs every day (although *crontab* is a more appropriate vehicle for such work):

```
# my.daily runs every day
daily processing
at now tomorrow < my.daily
```

4. The spacing of the three portions of the POSIX locale *timespec* is quite flexible as long as there are no ambiguities. Examples of various times and operand presentation include:

```
at 0815am Jan 24
at 8 :15amjan24
at now "+ 1day"
at 5 pm FRIday
at '17
```

```
78366         utc+
78367         30minutes'
```

78368 RATIONALE

78369 The *at* utility reads from standard input the commands to be executed at a later time. It may be
 78370 useful to redirect standard output and standard error within the specified commands.

78371 The **-t** *time* option was added as a new capability to support an internationalized way of
 78372 specifying a time for execution of the submitted job.

78373 Early proposals added a “jobname” concept as a way of giving submitted jobs names that are
 78374 meaningful to the user submitting them. The historical, system-specified *at_job_id* gives no
 78375 indication of what the job is. Upon further reflection, it was decided that the benefit of this was
 78376 not worth the change in historical interface. The *at* functionality is useful in simple
 78377 environments, but in large or complex situations, the functionality provided by the Batch
 78378 Services option is more suitable.

78379 The **-q** option historically has been an undocumented option, used mainly by the *batch* utility.

78380 The System V **-m** option was added to provide a method for informing users that an at-job had
 78381 completed. Otherwise, users are only informed when output to standard error or standard
 78382 output are not redirected.

78383 The behavior of *at* **<now>** was changed in an early proposal from being unspecified to
 78384 submitting a job for potentially immediate execution. Historical BSD *at* implementations support
 78385 this. Historical System V implementations give an error in that case, but a change to the System
 78386 V versions should have no backwards-compatibility ramifications.

78387 On BSD-based systems, a **-u** *user* option has allowed those with appropriate privileges to access
 78388 the work of other users. Since this is primarily a system administration feature and is not
 78389 universally implemented, it has been omitted. Similarly, a specification for the output format for
 78390 a user with appropriate privileges viewing the queues of other users has been omitted.

78391 The **-f** *file* option from System V is used instead of the BSD method of using the last operand as
 78392 the pathname. The BSD method is ambiguous—does:

```
78393 at 1200 friday
```

78394 mean the same thing if there is a file named **friday** in the current directory?

78395 The *at_job_id* is composed of a limited character set in historical practice, and it is mandated here
 78396 to invalidate systems that might try using characters that require shell quoting or that could not
 78397 be easily parsed by shell scripts.

78398 The *at* utility varies between System V and BSD systems in the way timezones are used. On
 78399 System V systems, the *TZ* variable affects the at-job submission times and the times displayed
 78400 for the user. On BSD systems, *TZ* is not taken into account. The BSD behavior is easily achieved
 78401 with the current specification. If the user wishes to have the timezone default to that of the
 78402 system, they merely need to issue the *at* command immediately following an unsetting or null
 78403 assignment to *TZ*. For example:

```
78404 TZ= at noon ...
```

78405 gives the desired BSD result.

78406 While the *yacc*-like grammar specified in the OPERANDS section is lexically unambiguous with
 78407 respect to the digit strings, a lexical analyzer would probably be written to look for and return
 78408 digit strings in those cases. The parser could then check whether the digit string returned is a
 78409 valid *day_number*, *year_number*, and so on, based on the context.

78410 **FUTURE DIRECTIONS**

78411 None.

78412 **SEE ALSO**78413 *batch*, *crontab*78414 XBD [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 215)78415 **CHANGE HISTORY**

78416 First released in Issue 2.

78417 **Issue 6**

78418 This utility is marked as part of the User Portability Utilities option.

78419 The following new requirements on POSIX implementations derive from alignment with the
78420 Single UNIX Specification:

- 78421
- If **-m** is not used, the job's standard output and standard error are provided to the user by
-
- 78422 mail.

78423 The effects of using the **-q** and **-t** options as defined in the IEEE P1003.2b draft standard are
78424 specified.

78425 The normative text is reworded to avoid use of the term “must” for application requirements.

78426 **Issue 7**78427 The *at* utility is moved from the User Portability Utilities option to the Base. User Portability
78428 Utilities is now an option for interactive utilities.78429 SD5-XCU-ERN-95 is applied, removing the references to fixed locations for the files referenced
78430 by the *at* utility.

78431 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

78432 **NAME**

78433 awk — pattern scanning and processing language

78434 **SYNOPSIS**78435 awk [-F *sepstring*] [-v *assignment*]... *program* [*argument*...] |78436 awk [-F *sepstring*] -f *progfile* [-f *progfile*]... [-v *assignment*]... |78437 [*argument*...]78438 **DESCRIPTION**

78439 The *awk* utility shall execute programs written in the *awk* programming language, which is
 78440 specialized for textual data manipulation. An *awk* program is a sequence of patterns and
 78441 corresponding actions. When input is read that matches a pattern, the action associated with that
 78442 pattern is carried out.

78443 Input shall be interpreted as a sequence of records. By default, a record is a line, less its
 78444 terminating <newline>, but this can be changed by using the **RS** built-in variable. Each record of
 78445 input shall be matched in turn against each pattern in the program. For each pattern matched,
 78446 the associated action shall be executed.

78447 The *awk* utility shall interpret each input record as a sequence of fields where, by default, a field
 78448 is a string of non-<blank> non-<newline> characters. This default <blank> and <newline> field
 78449 delimiter can be changed by using the **FS** built-in variable or the **-F *sepstring*** option. The *awk*
 78450 utility shall denote the first field in a record \$1, the second \$2, and so on. The symbol \$0 shall
 78451 refer to the entire record; setting any other field causes the re-evaluation of \$0. Assigning to \$0
 78452 shall reset the values of all other fields and the **NF** built-in variable.

78453 **OPTIONS**78454 The *awk* utility shall conform to XBD [Section 12.2](#) (on page 215).

78455 The following options shall be supported:

78456 **-F *sepstring*** Define the input field separator. This option shall be equivalent to: |78457 **-v FS=*sepstring*** |

78458 except that if **-F *sepstring*** and **-v FS=*sepstring*** are both used, it is unspecified |
 78459 whether the **FS** assignment resulting from **-F *sepstring*** is processed in command |
 78460 line order or is processed after the last **-v FS=*sepstring***. See the description of the |
 78461 **FS** built-in variable, and how it is used, in the EXTENDED DESCRIPTION section.

78462 **-f *progfile*** Specify the pathname of the file *progfile* containing an *awk* program. A pathname of |
 78463 '-' shall denote the standard input. If multiple instances of this option are |
 78464 specified, the concatenation of the files specified as *progfile* in the order specified
 78465 shall be the *awk* program. The *awk* program can alternatively be specified in the
 78466 command line as a single argument.

78467 **-v *assignment***

78468 The application shall ensure that the *assignment* argument is in the same form as an
 78469 *assignment* operand. The specified variable assignment shall occur prior to
 78470 executing the *awk* program, including the actions associated with **BEGIN** patterns
 78471 (if any). Multiple occurrences of this option can be specified.

78472 **OPERANDS**

78473 The following operands shall be supported:

78474 *program* If no **-f** option is specified, the first operand to *awk* shall be the text of the *awk*
 78475 program. The application shall supply the *program* operand as a single argument to
 78476 *awk*. If the text does not end in a <newline>, *awk* shall interpret the text as if it did.

78477 *argument* Either of the following two types of *argument* can be intermixed:

78478 *file* A pathname of a file that contains the input to be read, which is

78479 matched against the set of patterns in the program. If no *file* operands

78480 are specified, or if a *file* operand is *'-'*, the standard input shall be

78481 used.

78482 *assignment* An operand that begins with an <underscore> or alphabetic

78483 character from the portable character set (see the table in XBD [Section](#)

78484 [6.1](#), on page 125), followed by a sequence of underscores, digits, and

78485 alphabetic characters from the portable character set, followed by the *'='*

78486 character, shall specify a variable assignment rather than a pathname.

78487 The characters before the *'='* represent the name of an *awk* variable;

78488 if that name is an *awk* reserved word (see [Grammar](#), on page 2470)

78489 the behavior is undefined. The characters following the <equals-

78490 sign> shall be interpreted as if they appeared in the *awk* program

78491 preceded and followed by a double-quote (*'"*) character, as a

78492 **STRING** token (see [Grammar](#), on page 2470), except that if the last

78493 character is an unescaped <backslash>, it shall be interpreted as a

78494 literal <backslash> rather than as the first character of the sequence

78495 *"\"*". The variable shall be assigned the value of that **STRING**

78496 token and, if appropriate, shall be considered a *numeric string* (see

78497 [Expressions in awk](#), on page 2455), the variable shall also be assigned

78498 its numeric value. Each such variable assignment shall occur just

78499 prior to the processing of the following *file*, if any. Thus, an

78500 assignment before the first *file* argument shall be executed after the

78501 **BEGIN** actions (if any), while an assignment after the last *file*

78502 argument shall occur before the **END** actions (if any). If there are no

78503 *file* arguments, assignments shall be executed before processing the

78504 standard input.

78505 STDIN

78506 The standard input shall be used only if no *file* operands are specified, or if a *file* operand is *'-'*, +

78507 or if a *progfile* option-argument is *'-'*; see the INPUT FILES section. If the *awk* program contains

78508 no actions and no patterns, but is otherwise a valid *awk* program, standard input and any *file*

78509 operands shall not be read and *awk* shall exit with a return status of zero.

78510 INPUT FILES

78511 Input files to the *awk* program from any of the following sources shall be text files:

- 78512 • Any *file* operands or their equivalents, achieved by modifying the *awk* variables **ARGV**
- 78513 and **ARGC**
- 78514 • Standard input in the absence of any *file* operands
- 78515 • Arguments to the **getline** function

78516 Whether the variable **RS** is set to a value other than a <newline> or not, for these files,

78517 implementations shall support records terminated with the specified separator up to

78518 {**LINE_MAX**} bytes and may support longer records.

78519 If *-f progfile* is specified, the application shall ensure that the files named by each of the *progfile*

78520 option-arguments are text files and their concatenation, in the same order as they appear in the

78521 arguments, is an *awk* program.

78522 **ENVIRONMENT VARIABLES**78523 The following environment variables shall affect the execution of *awk*:

78524 **LANG** Provide a default value for the internationalization variables that are unset or null.
 78525 (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization
 78526 variables used to determine the values of locale categories.)

78527 **LC_ALL** If set to a non-empty string value, override the values of all the other
 78528 internationalization variables.

78529 **LC_COLLATE**

78530 Determine the locale for the behavior of ranges, equivalence classes, and multi-
 78531 character collating elements within regular expressions and in comparisons of
 78532 string values.

78533 **LC_CTYPE** Determine the locale for the interpretation of sequences of bytes of text data as
 78534 characters (for example, single-byte as opposed to multi-byte characters in
 78535 arguments and input files), the behavior of character classes within regular
 78536 expressions, the identification of characters as letters, and the mapping of
 78537 uppercase and lowercase characters for the **toupper** and **tolower** functions.

78538 **LC_MESSAGES**

78539 Determine the locale that should be used to affect the format and contents of
 78540 diagnostic messages written to standard error.

78541 **LC_NUMERIC**

78542 Determine the radix character used when interpreting numeric input, performing
 78543 conversions between numeric and string values, and formatting numeric output.
 78544 Regardless of locale, the <period> character (the decimal-point character of the
 78545 POSIX locale) is the decimal-point character recognized in processing *awk*
 78546 programs (including assignments in command line arguments).

78547 XSI **NLSPATH** Determine the location of message catalogs for the processing of **LC_MESSAGES**.

78548 **PATH** Determine the search path when looking for commands executed by *system(expr)*,
 78549 or input and output pipes; see XBD [Chapter 8](#) (on page 173).

78550 In addition, all environment variables shall be visible via the *awk* variable **ENVIRON**.78551 **ASYNCHRONOUS EVENTS**

78552 Default.

78553 **STDOUT**78554 The nature of the output files depends on the *awk* program.78555 **STDERR**

78556 The standard error shall be used only for diagnostic messages.

78557 **OUTPUT FILES**78558 The nature of the output files depends on the *awk* program.

EXTENDED DESCRIPTION

Overall Program Structure

An *awk* program is composed of pairs of the form:

```
pattern { action }
```

Either the pattern or the action (including the enclosing brace characters) can be omitted.

A missing pattern shall match any record of input, and a missing action shall be equivalent to:

```
{ print }
```

Execution of the *awk* program shall start by first executing the actions associated with all **BEGIN** patterns in the order they occur in the program. Then each *file* operand (or standard input if no files were specified) shall be processed in turn by reading data from the file until a record separator is seen (<newline> by default). Before the first reference to a field in the record is evaluated, the record shall be split into fields, according to the rules in [Regular Expressions](#) (on page 2461), using the value of **FS** that was current at the time the record was read. Each pattern in the program then shall be evaluated in the order of occurrence, and the action associated with each pattern that matches the current record executed. The action for a matching pattern shall be executed before evaluating subsequent patterns. Finally, the actions associated with all **END** patterns shall be executed in the order they occur in the program.

Expressions in awk

Expressions describe computations used in *patterns* and *actions*. In the following table, valid expression operations are given in groups from highest precedence first to lowest precedence last, with equal-precedence operators grouped between horizontal lines. In expression evaluation, where the grammar is formally ambiguous, higher precedence operators shall be evaluated before lower precedence operators. In this table *expr*, *expr1*, *expr2*, and *expr3* represent any expression, while *lvalue* represents any entity that can be assigned to (that is, on the left side of an assignment operator). The precise syntax of expressions is given in [Grammar](#) (on page 2470).

Table 4-1 Expressions in Decreasing Precedence in *awk*

Syntax	Name	Type of Result	Associativity
(<i>expr</i>)	Grouping	Type of <i>expr</i>	N/A
<i>\$expr</i>	Field reference	String	N/A
<i>lvalue</i> ++	Post-increment	Numeric	N/A
<i>lvalue</i> --	Post-decrement	Numeric	N/A
++ <i>lvalue</i>	Pre-increment	Numeric	N/A
-- <i>lvalue</i>	Pre-decrement	Numeric	N/A
<i>expr</i> ^ <i>expr</i>	Exponentiation	Numeric	Right
! <i>expr</i>	Logical not	Numeric	N/A
+ <i>expr</i>	Unary plus	Numeric	N/A
- <i>expr</i>	Unary minus	Numeric	N/A
<i>expr</i> * <i>expr</i>	Multiplication	Numeric	Left
<i>expr</i> / <i>expr</i>	Division	Numeric	Left
<i>expr</i> % <i>expr</i>	Modulus	Numeric	Left

Syntax	Name	Type of Result	Associativity
<i>expr</i> + <i>expr</i>	Addition	Numeric	Left
<i>expr</i> - <i>expr</i>	Subtraction	Numeric	Left
<i>expr</i> <i>expr</i>	String concatenation	String	Left
<i>expr</i> < <i>expr</i>	Less than	Numeric	None
<i>expr</i> <= <i>expr</i>	Less than or equal to	Numeric	None
<i>expr</i> != <i>expr</i>	Not equal to	Numeric	None
<i>expr</i> == <i>expr</i>	Equal to	Numeric	None
<i>expr</i> > <i>expr</i>	Greater than	Numeric	None
<i>expr</i> >= <i>expr</i>	Greater than or equal to	Numeric	None
<i>expr</i> ~ <i>expr</i>	ERE match	Numeric	None
<i>expr</i> !~ <i>expr</i>	ERE non-match	Numeric	None
<i>expr</i> in array	Array membership	Numeric	Left
(<i>index</i>) in array	Multi-dimension array membership	Numeric	Left
<i>expr</i> && <i>expr</i>	Logical AND	Numeric	Left
<i>expr</i> <i>expr</i>	Logical OR	Numeric	Left
<i>expr1</i> ? <i>expr2</i> : <i>expr3</i>	Conditional expression	Type of selected <i>expr2</i> or <i>expr3</i>	Right
<i>lvalue</i> ^= <i>expr</i>	Exponentiation assignment	Numeric	Right
<i>lvalue</i> %= <i>expr</i>	Modulus assignment	Numeric	Right
<i>lvalue</i> *= <i>expr</i>	Multiplication assignment	Numeric	Right
<i>lvalue</i> /= <i>expr</i>	Division assignment	Numeric	Right
<i>lvalue</i> += <i>expr</i>	Addition assignment	Numeric	Right
<i>lvalue</i> -= <i>expr</i>	Subtraction assignment	Numeric	Right
<i>lvalue</i> = <i>expr</i>	Assignment	Type of <i>expr</i>	Right

Each expression shall have either a string value, a numeric value, or both. Except as stated for specific contexts, the value of an expression shall be implicitly converted to the type needed for the context in which it is used. A string value shall be converted to a numeric value either by the equivalent of the following calls to functions defined by the ISO C standard:

```
setlocale(LC_NUMERIC, "");
numeric_value = atof(string_value);
```

or by converting the initial portion of the string to type **double** representation as follows:

The input string is decomposed into two parts: an initial, possibly empty, sequence of white-space characters (as specified by *isspace()*) and a subject sequence interpreted as a floating-point constant.

The expected form of the subject sequence is an optional '+' or '-' sign, then a non-empty sequence of digits optionally containing a <period>, then an optional exponent part. An exponent part consists of 'e' or 'E', followed by an optional sign, followed by one or more decimal digits.

The sequence starting with the first digit or the <period> (whichever occurs first) is interpreted as a floating constant of the C language, and if neither an exponent part nor a <period> appears, a <period> is assumed to follow the last digit in the string. If the subject sequence begins with a minus-sign, the value resulting from the conversion is negated.

A numeric value that is exactly equal to the value of an integer (see [Section 1.1.2](#), on page 2305) shall be converted to a string by the equivalent of a call to the **sprintf** function (see [String](#)

Functions, on page 2467) with the string "%d" as the *fmt* argument and the numeric value being converted as the first and only *expr* argument. Any other numeric value shall be converted to a string by the equivalent of a call to the **sprintf** function with the value of the variable **CONVFMT** as the *fmt* argument and the numeric value being converted as the first and only *expr* argument. The result of the conversion is unspecified if the value of **CONVFMT** is not a floating-point format specification. This volume of POSIX.1-2008 specifies no explicit conversions between numbers and strings. An application can force an expression to be treated as a number by adding zero to it, or can force it to be treated as a string by concatenating the null string (" ") to it.

A string value shall be considered a *numeric string* if it comes from one of the following:

1. Field variables
2. Input from the *getline()* function
3. **FILENAME**
4. **ARGV** array elements
5. **ENVIRON** array elements
6. Array elements created by the *split()* function
7. A command line variable assignment
8. Variable assignment from another numeric string variable

and an implementation-dependent condition corresponding to either case (a) or (b) below is met.

- a. After the equivalent of the following calls to functions defined by the ISO C standard, *string_value_end* would differ from *string_value*, and any characters before the terminating null character in *string_value_end* would be <blank> characters:

```
char *string_value_end;
setlocale(LC_NUMERIC, "");
numeric_value = strtod (string_value, &string_value_end);
```

- b. After all the following conversions have been applied, the resulting string would lexically be recognized as a **NUMBER** token as described by the lexical conventions in **Grammar** (on page 2470):

- All leading and trailing <blank> characters are discarded.
- If the first non-<blank> is '+' or '-', it is discarded.
- Each occurrence of the decimal point character from the current locale is changed to a <period>.

In case (a) the numeric value of the *numeric string* shall be the value that would be returned by the *strtod()* call. In case (b) if the first non-<blank> is '-', the numeric value of the *numeric string* shall be the negation of the numeric value of the recognized **NUMBER** token; otherwise, the numeric value of the *numeric string* shall be the numeric value of the recognized **NUMBER** token. Whether or not a string is a *numeric string* shall be relevant only in contexts where that term is used in this section.

When an expression is used in a Boolean context, if it has a numeric value, a value of zero shall be treated as false and any other value shall be treated as true. Otherwise, a string value of the null string shall be treated as false and any other value shall be treated as true. A Boolean context shall be one of the following:

- The first subexpression of a conditional expression
- An expression operated on by logical NOT, logical AND, or logical OR
- The second expression of a **for** statement
- The expression of an **if** statement
- The expression of the **while** clause in either a **while** or **do...while** statement
- An expression used as a pattern (as in Overall Program Structure)

All arithmetic shall follow the semantics of floating-point arithmetic as specified by the ISO C standard (see [Section 1.1.2](#), on page 2305).

The value of the expression:

```
expr1 ^ expr2
```

shall be equivalent to the value returned by the ISO C standard function call:

```
pow(expr1, expr2)
```

The expression:

```
lvalue ^= expr
```

shall be equivalent to the ISO C standard expression:

```
lvalue = pow(lvalue, expr)
```

except that lvalue shall be evaluated only once. The value of the expression:

```
expr1 % expr2
```

shall be equivalent to the value returned by the ISO C standard function call:

```
fmod(expr1, expr2)
```

The expression:

```
lvalue %= expr
```

shall be equivalent to the ISO C standard expression:

```
lvalue = fmod(lvalue, expr)
```

except that lvalue shall be evaluated only once.

Variables and fields shall be set by the assignment statement:

```
lvalue = expression
```

and the type of *expression* shall determine the resulting variable type. The assignment includes the arithmetic assignments ("**+=**", "**-=**", "***=**", "**/=**", "**%=**", "**^=**", "**++**", "**--**") all of which shall produce a numeric result. The left-hand side of an assignment and the target of increment and decrement operators can be one of a variable, an array with index, or a field selector.

The *awk* language supplies arrays that are used for storing numbers or strings. Arrays need not be declared. They shall initially be empty, and their sizes shall change dynamically. The subscripts, or element identifiers, are strings, providing a type of associative array capability. An array name followed by a subscript within square brackets can be used as an lvalue and thus as an expression, as described in the grammar; see [Grammar](#) (on page 2470). Unsubscripted array names can be used in only the following contexts:

- A parameter in a function definition or function call
- The **NAME** token following any use of the keyword **in** as specified in the grammar (see [Grammar](#), on page 2470); if the name used in this context is not an array name, the behavior is undefined

A valid array *index* shall consist of one or more <comma>-separated expressions, similar to the way in which multi-dimensional arrays are indexed in some programming languages. Because *awk* arrays are really one-dimensional, such a <comma>-separated list shall be converted to a single string by concatenating the string values of the separate expressions, each separated from the other by the value of the **SUBSEP** variable. Thus, the following two index operations shall be equivalent:

```
var[expr1, expr2, ... exprn]
```

```
var[expr1 SUBSEP expr2 SUBSEP ... SUBSEP exprn]
```

The application shall ensure that a multi-dimensioned *index* used with the **in** operator is parenthesized. The **in** operator, which tests for the existence of a particular array element, shall not cause that element to exist. Any other reference to a nonexistent array element shall automatically create it.

Comparisons (with the '<', '<=', '!=', '==', '>', and '>=' operators) shall be made numerically if both operands are numeric, if one is numeric and the other has a string value that is a numeric string, or if one is numeric and the other has the uninitialized value. Otherwise, operands shall be converted to strings as required and a string comparison shall be made using the locale-specific collation sequence. The value of the comparison expression shall be 1 if the relation is true, or 0 if the relation is false.

Variables and Special Variables

Variables can be used in an *awk* program by referencing them. With the exception of function parameters (see [User-Defined Functions](#), on page 2469), they are not explicitly declared. Function parameter names shall be local to the function; all other variable names shall be global. The same name shall not be used as both a function parameter name and as the name of a function or a special *awk* variable. The same name shall not be used both as a variable name with global scope and as the name of a function. The same name shall not be used within the same scope both as a scalar variable and as an array. Uninitialized variables, including scalar variables, array elements, and field variables, shall have an uninitialized value. An uninitialized value shall have both a numeric value of zero and a string value of the empty string. Evaluation of variables with an uninitialized value, to either string or numeric, shall be determined by the context in which they are used.

Field variables shall be designated by a '\$' followed by a number or numerical expression. The effect of the field number *expression* evaluating to anything other than a non-negative integer is unspecified; uninitialized variables or string values need not be converted to numeric values in this context. New field variables can be created by assigning a value to them. References to nonexistent fields (that is, fields after **NF**), shall evaluate to the uninitialized value. Such references shall not create new fields. However, assigning to a nonexistent field (for example, **\$(NF+2)=5**) shall increase the value of **NF**; create any intervening fields with the uninitialized value; and cause the value of **\$0** to be recomputed, with the fields being separated by the value of **OFS**. Each field variable shall have a string value or an uninitialized value when created. Field variables shall have the uninitialized value when created from **\$0** using **FS** and the variable does not contain any characters. If appropriate, the field variable shall be considered a numeric string (see [Expressions in awk](#), on page 2455).

78772	Implementations shall support the following other special variables that are set by <i>awk</i> :	
78773	ARGC	The number of elements in the ARGV array.
78774	ARGV	An array of command line arguments, excluding options and the <i>program</i>
78775		argument, numbered from zero to ARGC –1.
78776		The arguments in ARGV can be modified or added to; ARGC can be altered. As
78777		each input file ends, <i>awk</i> shall treat the next non-null element of ARGV , up to the
78778		current value of ARGC –1, inclusive, as the name of the next input file. Thus,
78779		setting an element of ARGV to null means that it shall not be treated as an input
78780		file. The name '–' indicates the standard input. If an argument matches the format
78781		of an <i>assignment</i> operand, this argument shall be treated as an <i>assignment</i> rather
78782		than a <i>file</i> argument.
78783	CONVFMT	The printf format for converting numbers to strings (except for output statements,
78784		where OFMT is used); "% . 6g" by default.
78785	ENVIRON	An array representing the value of the environment, as described in the <i>exec</i>
78786		functions defined in the System Interfaces volume of POSIX.1-2008. The indices of
78787		the array shall be strings consisting of the names of the environment variables, and
78788		the value of each array element shall be a string consisting of the value of that
78789		variable. If appropriate, the environment variable shall be considered a <i>numeric</i>
78790		<i>string</i> (see Expressions in awk , on page 2455); the array element shall also have its
78791		numeric value.
78792		In all cases where the behavior of <i>awk</i> is affected by environment variables
78793		(including the environment of any commands that <i>awk</i> executes via the system
78794		function or via pipeline redirections with the print statement, the printf statement,
78795		or the getline function), the environment used shall be the environment at the time
78796		<i>awk</i> began executing; it is implementation-defined whether any modification of
78797		ENVIRON affects this environment.
78798	FILENAME	A pathname of the current input file. Inside a BEGIN action the value is
78799		undefined. Inside an END action the value shall be the name of the last input file
78800		processed.
78801	FNR	The ordinal number of the current record in the current file. Inside a BEGIN action
78802		the value shall be zero. Inside an END action the value shall be the number of the
78803		last record processed in the last file processed.
78804	FS	Input field separator regular expression; a <space> by default.
78805	NF	The number of fields in the current record. Inside a BEGIN action, the use of NF is
78806		undefined unless a getline function without a <i>var</i> argument is executed previously.
78807		Inside an END action, NF shall retain the value it had for the last record read,
78808		unless a subsequent, redirected, getline function without a <i>var</i> argument is
78809		performed prior to entering the END action.
78810	NR	The ordinal number of the current record from the start of input. Inside a BEGIN
78811		action the value shall be zero. Inside an END action the value shall be the number
78812		of the last record processed.
78813	OFMT	The printf format for converting numbers to strings in output statements (see
78814		Output Statements , on page 2465); "% . 6g" by default. The result of the conversion
78815		is unspecified if the value of OFMT is not a floating-point format specification.

78816	OFS	The print statement output field separator; <space> by default.
78817	ORS	The print statement output record separator; a <newline> by default.
78818	RLENGTH	The length of the string matched by the match function.
78819	RS	The first character of the string value of RS shall be the input record separator; a <newline> by default. If RS contains more than one character, the results are unspecified. If RS is null, then records are separated by sequences consisting of a <newline> plus one or more blank lines, leading or trailing blank lines shall not result in empty records at the beginning or end of the input, and a <newline> shall always be a field separator, no matter what the value of FS is.
78820		
78821		
78822		
78823		
78824		
78825	RSTART	The starting position of the string matched by the match function, numbering from 1. This shall always be equivalent to the return value of the match function.
78826		
78827	SUBSEP	The subscript separator string for multi-dimensional arrays; the default value is implementation-defined.
78828		

78829 **Regular Expressions**

78830 The *awk* utility shall make use of the extended regular expression notation (see XBD [Section 9.4](#),
 78831 on page 188) except that it shall allow the use of C-language conventions for escaping special
 78832 characters within the EREs, as specified in the table in XBD [Chapter 5](#) (on page 121) ('\\',
 78833 '\\a', '\\b', '\\f', '\\n', '\\r', '\\t', '\\v') and the following table; these escape sequences
 78834 shall be recognized both inside and outside bracket expressions. Note that records need not be
 78835 separated by <newline> characters and string constants can contain <newline> characters, so
 78836 even the "\\n" sequence is valid in *awk* EREs. Using a <slash> character within an ERE requires
 78837 the escaping shown in the following table.

Table 4-2 Escape Sequences in *awk*

Escape Sequence	Description	Meaning
\ "	<backslash> <quotation-mark>	<quotation-mark> character
\ /	<backslash> <slash>	<slash> character
\ ddd	A <backslash> character followed by the longest sequence of one, two, or three octal-digit characters (01234567). If all of the digits are 0 (that is, representation of the NUL character), the behavior is undefined.	The character whose encoding is represented by the one, two, or three-digit octal integer. Multi-byte characters require multiple, concatenated escape sequences of this type, including the leading <backslash> for each byte.
\ c	A <backslash> character followed by any character not described in this table or in the table in XBD Chapter 5 (on page 121) ('\\', '\a', '\b', '\f', '\n', '\r', '\t', '\v').	Undefined

A regular expression can be matched against a specific field or string by using one of the two regular expression matching operators, `'~'` and `"!~"`. These operators shall interpret their right-hand operand as a regular expression and their left-hand operand as a string. If the regular expression matches the string, the `'~'` expression shall evaluate to a value of 1, and the `"!~"` expression shall evaluate to a value of 0. (The regular expression matching operation is as defined by the term `matched` in XBD Section 9.1 (on page 181), where a match occurs on any part of the string unless the regular expression is limited with the <circumflex> or <dollar-sign> special characters.) If the regular expression does not match the string, the `'~'` expression shall evaluate to a value of 0, and the `"!~"` expression shall evaluate to a value of 1. If the right-hand operand is any expression other than the lexical token **ERE**, the string value of the expression shall be interpreted as an extended regular expression, including the escape conventions described above. Note that these same escape conventions shall also be applied in determining the value of a string literal (the lexical token **STRING**), and thus shall be applied a second time when a string literal is used in this context.

When an **ERE** token appears as an expression in any context other than as the right-hand of the `'~'` or `"!~"` operator or as one of the built-in function arguments described below, the value of the resulting expression shall be the equivalent of:

```
$0 ~ /ere/
```

The *ere* argument to the **gsub**, **match**, **sub** functions, and the *fs* argument to the **split** function (see [String Functions](#), on page 2467) shall be interpreted as extended regular expressions. These can be either **ERE** tokens or arbitrary expressions, and shall be interpreted in the same manner as the right-hand side of the `'~'` or `"!~"` operator.

An extended regular expression can be used to separate fields by assigning a string containing the expression to the built-in variable **FS**, either directly or as a consequence of using the `-F sepstring` option. The default value of the **FS** variable shall be a single <space>. The following describes **FS** behavior:

1. If **FS** is a null string, the behavior is unspecified.

2. If **FS** is a single character:

- a. If **FS** is <space>, skip leading and trailing <blank> and <newline> characters; fields shall be delimited by sets of one or more <blank> or <newline> characters.
- b. Otherwise, if **FS** is any other character *c*, fields shall be delimited by each single occurrence of *c*.

3. Otherwise, the string value of **FS** shall be considered to be an extended regular expression. Each occurrence of a sequence matching the extended regular expression shall delimit fields.

Except for the '`~`' and '`!~`' operators, and in the **gsub**, **match**, **split**, and **sub** built-in functions, ERE matching shall be based on input records; that is, record separator characters (the first character of the value of the variable **RS**, <newline> by default) cannot be embedded in the expression, and no expression shall match the record separator character. If the record separator is not <newline>, <newline> characters embedded in the expression can be matched. For the '`~`' and '`!~`' operators, and in those four built-in functions, ERE matching shall be based on text strings; that is, any character (including <newline> and the record separator) can be embedded in the pattern, and an appropriate pattern shall match any character. However, in all *awk* ERE matching, the use of one or more NUL characters in the pattern, input record, or text string produces undefined results.

Patterns

A *pattern* is any valid *expression*, a range specified by two expressions separated by a comma, or one of the two special patterns **BEGIN** or **END**.

Special Patterns

The *awk* utility shall recognize two special patterns, **BEGIN** and **END**. Each **BEGIN** pattern shall be matched once and its associated action executed before the first record of input is read—except possibly by use of the **getline** function (see [Input/Output and General Functions](#), on page 2468) in a prior **BEGIN** action—and before command line assignment is done. Each **END** pattern shall be matched once and its associated action executed after the last record of input has been read. These two patterns shall have associated actions.

BEGIN and **END** shall not combine with other patterns. Multiple **BEGIN** and **END** patterns shall be allowed. The actions associated with the **BEGIN** patterns shall be executed in the order specified in the program, as are the **END** actions. An **END** pattern can precede a **BEGIN** pattern in a program.

If an *awk* program consists of only actions with the pattern **BEGIN**, and the **BEGIN** action contains no **getline** function, *awk* shall exit without reading its input when the last statement in the last **BEGIN** action is executed. If an *awk* program consists of only actions with the pattern **END** or only actions with the patterns **BEGIN** and **END**, the input shall be read before the statements in the **END** actions are executed.

Expression Patterns

An expression pattern shall be evaluated as if it were an expression in a Boolean context. If the result is true, the pattern shall be considered to match, and the associated action (if any) shall be executed. If the result is false, the action shall not be executed.

Pattern Ranges

A pattern range consists of two expressions separated by a comma; in this case, the action shall be performed for all records between a match of the first expression and the following match of the second expression, inclusive. At this point, the pattern range can be repeated starting at input records subsequent to the end of the matched range.

Actions

An action is a sequence of statements as shown in the grammar in [Grammar](#) (on page 2470). Any single statement can be replaced by a statement list enclosed in curly braces. The application shall ensure that statements in a statement list are separated by <newline> or <semicolon> characters. Statements in a statement list shall be executed sequentially in the order that they appear.

The *expression* acting as the conditional in an **if** statement shall be evaluated and if it is non-zero or non-null, the following statement shall be executed; otherwise, if **else** is present, the statement following the **else** shall be executed.

The **if**, **while**, **do...while**, **for**, **break**, and **continue** statements are based on the ISO C standard (see [Section 1.1.2](#), on page 2305), except that the Boolean expressions shall be treated as described in [Expressions in awk](#) (on page 2455), and except in the case of:

```
for (variable in array)
```

which shall iterate, assigning each *index* of *array* to *variable* in an unspecified order. The results of adding new elements to *array* within such a **for** loop are undefined. If a **break** or **continue** statement occurs outside of a loop, the behavior is undefined.

The **delete** statement shall remove an individual array element. Thus, the following code deletes an entire array:

```
for (index in array)
    delete array[index]
```

The **next** statement shall cause all further processing of the current input record to be abandoned. The behavior is undefined if a **next** statement appears or is invoked in a **BEGIN** or **END** action.

The **exit** statement shall invoke all **END** actions in the order in which they occur in the program source and then terminate the program without reading further input. An **exit** statement inside an **END** action shall terminate the program without further execution of **END** actions. If an expression is specified in an **exit** statement, its numeric value shall be the exit status of *awk*, unless subsequent errors are encountered or a subsequent **exit** statement with an expression is executed.

Output Statements

Both **print** and **printf** statements shall write to standard output by default. The output shall be written to the location specified by *output_redirection* if one is supplied, as follows:

```
> expression
>> expression
| expression
```

In all cases, the *expression* shall be evaluated to produce a string that is used as a pathname into which to write (for '**>**' or '**>>**') or as a command to be executed (for '**|**'). Using the first two forms, if the file of that name is not currently open, it shall be opened, creating it if necessary and using the first form, truncating the file. The output then shall be appended to the file. As long as the file remains open, subsequent calls in which *expression* evaluates to the same string value shall simply append output to the file. The file remains open until the **close** function (see [Input/Output and General Functions](#), on page 2468) is called with an expression that evaluates to the same string value.

The third form shall write output onto a stream piped to the input of a command. The stream shall be created if no stream is currently open with the value of *expression* as its command name. The stream created shall be equivalent to one created by a call to the *popen()* function defined in the System Interfaces volume of POSIX.1-2008 with the value of *expression* as the *command* argument and a value of *w* as the *mode* argument. As long as the stream remains open, subsequent calls in which *expression* evaluates to the same string value shall write output to the existing stream. The stream shall remain open until the **close** function (see [Input/Output and General Functions](#), on page 2468) is called with an expression that evaluates to the same string value. At that time, the stream shall be closed as if by a call to the *pclose()* function defined in the System Interfaces volume of POSIX.1-2008.

As described in detail by the grammar in [Grammar](#) (on page 2470), these output statements shall take a <comma>-separated list of *expressions* referred to in the grammar by the non-terminal symbols **expr_list**, **print_expr_list**, or **print_expr_list_opt**. This list is referred to here as the *expression list*, and each member is referred to as an *expression argument*.

The **print** statement shall write the value of each expression argument onto the indicated output stream separated by the current output field separator (see variable **OFS** above), and terminated by the output record separator (see variable **ORS** above). All expression arguments shall be taken as strings, being converted if necessary; this conversion shall be as described in [Expressions in awk](#) (on page 2455), with the exception that the **printf** format in **OFMT** shall be used instead of the value in **CONVFMT**. An empty expression list shall stand for the whole input record (\$0).

The **printf** statement shall produce output based on a notation similar to the File Format Notation used to describe file formats in this volume of POSIX.1-2008 (see XBD [Chapter 5](#), on page 121). Output shall be produced as specified with the first *expression* argument as the string *format* and subsequent *expression* arguments as the strings *arg1* to *argn*, inclusive, with the following exceptions:

1. The *format* shall be an actual character string rather than a graphical representation. Therefore, it cannot contain empty character positions. The <space> in the *format* string, in any context other than a *flag* of a conversion specification, shall be treated as an ordinary character that is copied to the output.
2. If the character set contains a '**Δ**' character and that character appears in the *format* string, it shall be treated as an ordinary character that is copied to the output.

3. The *escape sequences* beginning with a <backslash> character shall be treated as sequences of ordinary characters that are copied to the output. Note that these same sequences shall be interpreted lexically by *awk* when they appear in literal strings, but they shall not be treated specially by the **printf** statement.
4. A *field width* or *precision* can be specified as the ' * ' character instead of a digit string. In this case the next argument from the expression list shall be fetched and its numeric value taken as the field width or precision.
5. The implementation shall not precede or follow output from the *d* or *u* conversion specifier characters with <blank> characters not specified by the *format* string.
6. The implementation shall not precede output from the *o* conversion specifier character with leading zeros not specified by the *format* string.
7. For the *c* conversion specifier character: if the argument has a numeric value, the character whose encoding is that value shall be output. If the value is zero or is not the encoding of any character in the character set, the behavior is undefined. If the argument does not have a numeric value, the first character of the string value shall be output; if the string does not contain any characters, the behavior is undefined.
8. For each conversion specification that consumes an argument, the next expression argument shall be evaluated. With the exception of the *c* conversion specifier character, the value shall be converted (according to the rules specified in [Expressions in awk](#), on page 2455) to the appropriate type for the conversion specification.
9. If there are insufficient expression arguments to satisfy all the conversion specifications in the *format* string, the behavior is undefined.
10. If any character sequence in the *format* string begins with a ' % ' character, but does not form a valid conversion specification, the behavior is unspecified.

Both **print** and **printf** can output at least {LINE_MAX} bytes.

Functions

The *awk* language has a variety of built-in functions: arithmetic, string, input/output, and general.

Arithmetic Functions

The arithmetic functions, except for **int**, shall be based on the ISO C standard (see [Section 1.1.2](#), on page 2305). The behavior is undefined in cases where the ISO C standard specifies that an error be returned or that the behavior is undefined. Although the grammar (see [Grammar](#), on page 2470) permits built-in functions to appear with no arguments or parentheses, unless the argument or parentheses are indicated as optional in the following list (by displaying them within the " [] " brackets), such use is undefined.

atan2 (<i>y</i> , <i>x</i>)	Return arctangent of <i>y</i> / <i>x</i> in radians in the range $[-\pi, \pi]$.
cos (<i>x</i>)	Return cosine of <i>x</i> , where <i>x</i> is in radians.
sin (<i>x</i>)	Return sine of <i>x</i> , where <i>x</i> is in radians.
exp (<i>x</i>)	Return the exponential function of <i>x</i> .
log (<i>x</i>)	Return the natural logarithm of <i>x</i> .

79043 **sqrt**(*x*) Return the square root of *x*.

79044 **int**(*x*) Return the argument truncated to an integer. Truncation shall be toward 0 when
79045 *x*>0.

79046 **rand**() Return a random number *n*, such that $0 \leq n < 1$.

79047 **srand**([*expr*]) Set the seed value for *rand* to *expr* or use the time of day if *expr* is omitted. The
79048 previous seed value shall be returned.

79049 **String Functions**

79050 The string functions in the following list shall be supported. Although the grammar (see
79051 [Grammar](#), on page 2470) permits built-in functions to appear with no arguments or parentheses,
79052 unless the argument or parentheses are indicated as optional in the following list (by displaying
79053 them within the " [] " brackets), such use is undefined.

79054 **gsub**(*ere*, *repl* [, *in*])
79055 Behave like **sub** (see below), except that it shall replace all occurrences of the
79056 regular expression (like the *ed* utility global substitute) in *\$0* or in the *in* argument,
79057 when specified.

79058 **index**(*s*, *t*) Return the position, in characters, numbering from 1, in string *s* where string *t* first
79059 occurs, or zero if it does not occur at all.

79060 **length**[(*[s]*)] Return the length, in characters, of its argument taken as a string, or of the whole
79061 record, *\$0*, if there is no argument.

79062 **match**(*s*, *ere*) Return the position, in characters, numbering from 1, in string *s* where the
79063 extended regular expression *ere* occurs, or zero if it does not occur at all. RSTART
79064 shall be set to the starting position (which is the same as the returned value), zero
79065 if no match is found; RLENGTH shall be set to the length of the matched string, -1
79066 if no match is found.

79067 **split**(*s*, *a* [, *fs*])
79068 Split the string *s* into array elements *a*[1], *a*[2], ..., *a*[*n*], and return *n*. All elements
79069 of the array shall be deleted before the split is performed. The separation shall be
79070 done with the ERE *fs* or with the field separator **FS** if *fs* is not given. Each array
79071 element shall have a string value when created and, if appropriate, the array
79072 element shall be considered a numeric string (see [Expressions in awk](#), on page
79073 2455). The effect of a null string as the value of *fs* is unspecified.

79074 **sprintf**(*fnt*, *expr*, *expr*, ...)
79075 Format the expressions according to the **printf** format given by *fnt* and return the
79076 resulting string.

79077 **sub**(*ere*, *repl* [, *in*])
79078 Substitute the string *repl* in place of the first instance of the extended regular
79079 expression *ERE* in string *in* and return the number of substitutions. An
79080 <ampersand> (' & ') appearing in the string *repl* shall be replaced by the string
79081 from *in* that matches the ERE. An <ampersand> preceded with a <backslash> shall
79082 be interpreted as the literal <ampersand> character. An occurrence of two
79083 consecutive <backslash> characters shall be interpreted as just a single literal
79084 <backslash> character. Any other occurrence of a <backslash> (for example,
79085 preceding any other character) shall be treated as a literal <backslash> character.
79086 Note that if *repl* is a string literal (the lexical token **STRING**; see [Grammar](#), on page
79087 2470), the handling of the <ampersand> character occurs after any lexical

processing, including any lexical <backslash>-escape sequence processing. If *in* is specified and it is not an lvalue (see [Expressions in awk](#), on page 2455), the behavior is undefined. If *in* is omitted, *awk* shall use the current record (\$0) in its place.

substr(*s*, *m* [, *n*])

Return the at most *n*-character substring of *s* that begins at position *m*, numbering from 1. If *n* is omitted, or if *n* specifies more characters than are left in the string, the length of the substring shall be limited by the length of the string *s*.

tolower(*s*) Return a string based on the string *s*. Each character in *s* that is an uppercase letter specified to have a **tolower** mapping by the *LC_CTYPE* category of the current locale shall be replaced in the returned string by the lowercase letter specified by the mapping. Other characters in *s* shall be unchanged in the returned string.

toupper(*s*) Return a string based on the string *s*. Each character in *s* that is a lowercase letter specified to have a **toupper** mapping by the *LC_CTYPE* category of the current locale is replaced in the returned string by the uppercase letter specified by the mapping. Other characters in *s* are unchanged in the returned string.

All of the preceding functions that take *ERE* as a parameter expect a pattern or a string valued expression that is a regular expression as defined in [Regular Expressions](#) (on page 2461).

Input/Output and General Functions

The input/output and general functions are:

close(*expression*)

Close the file or pipe opened by a **print** or **printf** statement or a call to **getline** with the same string-valued *expression*. The limit on the number of open *expression* arguments is implementation-defined. If the close was successful, the function shall return zero; otherwise, it shall return non-zero.

expression | **getline** [*var*]

Read a record of input from a stream piped from the output of a command. The stream shall be created if no stream is currently open with the value of *expression* as its command name. The stream created shall be equivalent to one created by a call to the *popen*() function with the value of *expression* as the *command* argument and a value of *r* as the *mode* argument. As long as the stream remains open, subsequent calls in which *expression* evaluates to the same string value shall read subsequent records from the stream. The stream shall remain open until the **close** function is called with an expression that evaluates to the same string value. At that time, the stream shall be closed as if by a call to the *pclose*() function. If *var* is omitted, \$0 and **NF** shall be set; otherwise, *var* shall be set and, if appropriate, it shall be considered a numeric string (see [Expressions in awk](#), on page 2455).

The **getline** operator can form ambiguous constructs when there are unparenthesized operators (including concatenate) to the left of the ' | ' (to the beginning of the expression containing **getline**). In the context of the ' \$ ' operator, ' | ' shall behave as if it had a lower precedence than ' \$ '. The result of evaluating other operators is unspecified, and conforming applications shall parenthesize properly all such usages.

getline Set \$0 to the next input record from the current input file. This form of **getline** shall set the **NF**, **NR**, and **FNR** variables.

getline *var* Set variable *var* to the next input record from the current input file and, if appropriate, *var* shall be considered a numeric string (see [Expressions in awk](#), on page 2455). This form of **getline** shall set the **FNR** and **NR** variables.

getline [*var*] < *expression*

Read the next record of input from a named file. The *expression* shall be evaluated to produce a string that is used as a pathname. If the file of that name is not currently open, it shall be opened. As long as the stream remains open, subsequent calls in which *expression* evaluates to the same string value shall read subsequent records from the file. The file shall remain open until the **close** function is called with an expression that evaluates to the same string value. If *var* is omitted, **\$0** and **NF** shall be set; otherwise, *var* shall be set and, if appropriate, it shall be considered a numeric string (see [Expressions in awk](#), on page 2455).

The **getline** operator can form ambiguous constructs when there are unparenthesized binary operators (including concatenate) to the right of the '**<**' (up to the end of the expression containing the **getline**). The result of evaluating such a construct is unspecified, and conforming applications shall parenthesize properly all such usages.

system(*expression*)

Execute the command given by *expression* in a manner equivalent to the *system()* function defined in the System Interfaces volume of POSIX.1-2008 and return the exit status of the command.

All forms of **getline** shall return 1 for successful input, zero for end-of-file, and -1 for an error.

Where strings are used as the name of a file or pipeline, the application shall ensure that the strings are textually identical. The terminology “same string value” implies that “equivalent strings”, even those that differ only by <space> characters, represent different files.

User-Defined Functions

The *awk* language also provides user-defined functions. Such functions can be defined as:

```
function name([parameter, ...]) { statements }
```

A function can be referred to anywhere in an *awk* program; in particular, its use can precede its definition. The scope of a function is global.

Function parameters, if present, can be either scalars or arrays; the behavior is undefined if an array name is passed as a parameter that the function uses as a scalar, or if a scalar expression is passed as a parameter that the function uses as an array. Function parameters shall be passed by value if scalar and by reference if array name.

The number of parameters in the function definition need not match the number of parameters in the function call. Excess formal parameters can be used as local variables. If fewer arguments are supplied in a function call than are in the function definition, the extra parameters that are used in the function body as scalars shall evaluate to the uninitialized value until they are otherwise initialized, and the extra parameters that are used in the function body as arrays shall be treated as uninitialized arrays where each element evaluates to the uninitialized value until otherwise initialized.

When invoking a function, no white space can be placed between the function name and the opening parenthesis. Function calls can be nested and recursive calls can be made upon functions. Upon return from any nested or recursive function call, the values of all of the calling function's parameters shall be unchanged, except for array parameters passed by reference. The

return statement can be used to return a value. If a **return** statement appears outside of a function definition, the behavior is undefined.

In the function definition, <newline> characters shall be optional before the opening brace and after the closing brace. Function definitions can appear anywhere in the program where a *pattern-action* pair is allowed.

Grammar

The grammar in this section and the lexical conventions in the following section shall together describe the syntax for *awk* programs. The general conventions for this style of grammar are described in [Section 1.3](#) (on page 2309). A valid program can be represented as the non-terminal symbol *program* in the grammar. This formal syntax shall take precedence over the preceding text syntax description.

```
%token NAME NUMBER STRING ERE
%token FUNC_NAME /* Name followed by '(' without white space. */

/* Keywords */
%token Begin End
/* 'BEGIN' 'END' */

%token Break Continue Delete Do Else
/* 'break' 'continue' 'delete' 'do' 'else' */

%token Exit For Function If In
/* 'exit' 'for' 'function' 'if' 'in' */

%token Next Print Printf Return While
/* 'next' 'print' 'printf' 'return' 'while' */

/* Reserved function names */
%token BUILTIN_FUNC_NAME
/* One token for the following:
 * atan2 cos sin exp log sqrt int rand srand
 * gsub index length match split sprintf sub
 * substr tolower toupper close system
 */
%token GETLINE
/* Syntactically different from other built-ins. */

/* Two-character tokens. */
%token ADD_ASSIGN SUB_ASSIGN MUL_ASSIGN DIV_ASSIGN MOD_ASSIGN POW_ASSIGN
/* '+=' '-=' '*=' '/=' '%=' '^=' */

%token OR AND NO_MATCH EQ LE GE NE INCR DECR APPEND
/* '||' '&&' '!' '~' '==' '<=' '>=' '!=' '++' '--' '>>' */

/* One-character tokens. */
%token '{' '}' '(' ')' '[' ']' ',' ';' NEWLINE
%token '+' '-' '*' '%' '^' '!' '>' '<' '|' '?' ':' '~' '$' '='

%start program
%%

program : item_list
        | actionless_item_list
        ;
```

```

79222     item_list      : newline_opt
79223                     | actionless_item_list item terminator
79224                     | item_list           item terminator
79225                     | item_list           action terminator
79226                     ;

79227     actionless_item_list : item_list           pattern terminator
79228                     | actionless_item_list pattern terminator
79229                     ;

79230     item              : pattern action
79231                     | Function NAME      '(' param_list_opt ')'
79232                       newline_opt action
79233                     | Function FUNC_NAME '(' param_list_opt ')'
79234                       newline_opt action
79235                     ;

79236     param_list_opt    : /* empty */
79237                     | param_list
79238                     ;

79239     param_list        : NAME
79240                     | param_list ',' NAME
79241                     ;

79242     pattern           : Begin
79243                     | End
79244                     | expr
79245                     | expr ',' newline_opt expr
79246                     ;

79247     action            : '{' newline_opt
79248                     | '{' newline_opt terminated_statement_list
79249                     | '{' newline_opt unterminated_statement_list
79250                     ;

79251     terminator        : terminator ';'
79252                     | terminator NEWLINE
79253                     | ';'
79254                     | NEWLINE
79255                     ;

79256     terminated_statement_list : terminated_statement
79257                     | terminated_statement_list terminated_statement
79258                     ;

79259     unterminated_statement_list : unterminated_statement
79260                     | terminated_statement_list unterminated_statement
79261                     ;

79262     terminated_statement : action newline_opt
79263                     | If '(' expr ')' newline_opt terminated_statement
79264                     | If '(' expr ')' newline_opt terminated_statement
79265                       Else newline_opt terminated_statement
79266                     | While '(' expr ')' newline_opt terminated_statement
79267                     | For '(' simple_statement_opt ';'
79268                       expr_opt ';' simple_statement_opt ')' newline_opt

```

```

79269         terminated_statement
79270     | For '(' NAME In NAME ')' newline_opt
79271         terminated_statement
79272     | ';' newline_opt
79273     | terminatable_statement NEWLINE newline_opt
79274     | terminatable_statement ';' newline_opt
79275     ;

79276     unterminated_statement : terminatable_statement
79277     | If '(' expr ')' newline_opt unterminated_statement
79278     | If '(' expr ')' newline_opt terminated_statement
79279     | Else newline_opt unterminated_statement
79280     | While '(' expr ')' newline_opt unterminated_statement
79281     | For '(' simple_statement_opt ';'
79282     | expr_opt ';' simple_statement_opt ')' newline_opt
79283     | unterminated_statement
79284     | For '(' NAME In NAME ')' newline_opt
79285     | unterminated_statement
79286     ;

79287     terminatable_statement : simple_statement
79288     | Break
79289     | Continue
79290     | Next
79291     | Exit expr_opt
79292     | Return expr_opt
79293     | Do newline_opt terminated_statement While '(' expr ')'
79294     ;

79295     simple_statement_opt : /* empty */
79296     | simple_statement
79297     ;

79298     simple_statement : Delete NAME '[' expr_list ']'
79299     | expr
79300     | print_statement
79301     ;

79302     print_statement : simple_print_statement
79303     | simple_print_statement output_redirection
79304     ;

79305     simple_print_statement : Print print_expr_list_opt
79306     | Print '(' multiple_expr_list ')'
79307     | Printf print_expr_list
79308     | Printf '(' multiple_expr_list ')'
79309     ;

79310     output_redirection : '>' expr
79311     | APPEND expr
79312     | '|' expr
79313     ;

79314     expr_list_opt : /* empty */
79315     | expr_list
79316     ;

```

```

79317     expr_list      : expr
79318                     | multiple_expr_list
79319                     ;

79320     multiple_expr_list : expr ',' newline_opt expr
79321                       | multiple_expr_list ',' newline_opt expr
79322                       ;

79323     expr_opt         : /* empty */
79324                       | expr
79325                       ;

79326     expr              : unary_expr
79327                       | non_unary_expr
79328                       ;

79329     unary_expr        : '+' expr
79330                       | '-' expr
79331                       | unary_expr '^'      expr
79332                       | unary_expr '*'      expr
79333                       | unary_expr '/'      expr
79334                       | unary_expr '%'      expr
79335                       | unary_expr '+'      expr
79336                       | unary_expr '-'      expr
79337                       | unary_expr          non_unary_expr
79338                       | unary_expr '<'      expr
79339                       | unary_expr LE      expr
79340                       | unary_expr NE      expr
79341                       | unary_expr EQ      expr
79342                       | unary_expr '>'      expr
79343                       | unary_expr GE      expr
79344                       | unary_expr '~'      expr
79345                       | unary_expr NO_MATCH expr
79346                       | unary_expr In NAME
79347                       | unary_expr AND newline_opt expr
79348                       | unary_expr OR  newline_opt expr
79349                       | unary_expr '?' expr ':' expr
79350                       | unary_input_function
79351                       ;

79352     non_unary_expr    : '(' expr ')'
79353                       | '!' expr
79354                       | non_unary_expr '^'      expr
79355                       | non_unary_expr '*'      expr
79356                       | non_unary_expr '/'      expr
79357                       | non_unary_expr '%'      expr
79358                       | non_unary_expr '+'      expr
79359                       | non_unary_expr '-'      expr
79360                       | non_unary_expr          non_unary_expr
79361                       | non_unary_expr '<'      expr
79362                       | non_unary_expr LE      expr
79363                       | non_unary_expr NE      expr
79364                       | non_unary_expr EQ      expr
79365                       | non_unary_expr '>'      expr

```

```

79366         | non_unary_expr GE          expr
79367         | non_unary_expr '~'          expr
79368         | non_unary_expr NO_MATCH expr
79369         | non_unary_expr In NAME
79370         | '(' multiple_expr_list ')' In NAME
79371         | non_unary_expr AND newline_opt expr
79372         | non_unary_expr OR  newline_opt expr
79373         | non_unary_expr '?' expr ':' expr
79374         | NUMBER
79375         | STRING
79376         | lvalue
79377         | ERE
79378         | lvalue INCR
79379         | lvalue DECR
79380         | INCR lvalue
79381         | DECR lvalue
79382         | lvalue POW_ASSIGN expr
79383         | lvalue MOD_ASSIGN expr
79384         | lvalue MUL_ASSIGN expr
79385         | lvalue DIV_ASSIGN expr
79386         | lvalue ADD_ASSIGN expr
79387         | lvalue SUB_ASSIGN expr
79388         | lvalue '=' expr
79389         | FUNC_NAME '(' expr_list_opt ')'
79390         /* no white space allowed before '(' */
79391         | BUILTIN_FUNC_NAME '(' expr_list_opt ')'
79392         | BUILTIN_FUNC_NAME
79393         | non_unary_input_function
79394         ;

79395     print_expr_list_opt : /* empty */
79396         | print_expr_list
79397         ;

79398     print_expr_list    : print_expr
79399         | print_expr_list ',' newline_opt print_expr
79400         ;

79401     print_expr         : unary_print_expr
79402         | non_unary_print_expr
79403         ;

79404     unary_print_expr   : '+' print_expr
79405         | '-' print_expr
79406         | unary_print_expr '^'      print_expr
79407         | unary_print_expr '*'      print_expr
79408         | unary_print_expr '/'      print_expr
79409         | unary_print_expr '%'      print_expr
79410         | unary_print_expr '+'      print_expr
79411         | unary_print_expr '-'      print_expr
79412         | unary_print_expr          non_unary_print_expr
79413         | unary_print_expr '~'      print_expr
79414         | unary_print_expr NO_MATCH print_expr
79415         | unary_print_expr In NAME

```



```

79416         | unary_print_expr AND newline_opt print_expr
79417         | unary_print_expr OR  newline_opt print_expr
79418         | unary_print_expr '?' print_expr ':' print_expr
79419         ;

79420     non_unary_print_expr : '(' expr ')'
79421         | '!' print_expr
79422         | non_unary_print_expr '^'      print_expr
79423         | non_unary_print_expr '*'      print_expr
79424         | non_unary_print_expr '/'      print_expr
79425         | non_unary_print_expr '%'      print_expr
79426         | non_unary_print_expr '+'      print_expr
79427         | non_unary_print_expr '-'      print_expr
79428         | non_unary_print_expr          non_unary_print_expr
79429         | non_unary_print_expr '~'      print_expr
79430         | non_unary_print_expr NO_MATCH print_expr
79431         | non_unary_print_expr In NAME
79432         | '(' multiple_expr_list ')' In NAME
79433         | non_unary_print_expr AND newline_opt print_expr
79434         | non_unary_print_expr OR  newline_opt print_expr
79435         | non_unary_print_expr '?' print_expr ':' print_expr
79436         | NUMBER
79437         | STRING
79438         | lvalue
79439         | ERE
79440         | lvalue INCR
79441         | lvalue DECR
79442         | INCR lvalue
79443         | DECR lvalue
79444         | lvalue POW_ASSIGN print_expr
79445         | lvalue MOD_ASSIGN print_expr
79446         | lvalue MUL_ASSIGN print_expr
79447         | lvalue DIV_ASSIGN print_expr
79448         | lvalue ADD_ASSIGN print_expr
79449         | lvalue SUB_ASSIGN print_expr
79450         | lvalue '=' print_expr
79451         | FUNC_NAME '(' expr_list_opt ')'
79452             /* no white space allowed before '(' */
79453         | BUILTIN_FUNC_NAME '(' expr_list_opt ')'
79454         | BUILTIN_FUNC_NAME
79455         ;

79456     lvalue      : NAME
79457         | NAME '[' expr_list ']'
79458         | '$' expr
79459         ;

79460     non_unary_input_function : simple_get
79461         | simple_get '<' expr
79462         | non_unary_expr '|' simple_get
79463         ;

79464     unary_input_function : unary_expr '|' simple_get
79465         ;

```

```

79466 simple_get      : GETLINE
79467                  | GETLINE lvalue
79468                  ;
79469 newline_opt      : /* empty */
79470                  | newline_opt NEWLINE
79471                  ;

```

This grammar has several ambiguities that shall be resolved as follows:

- Operator precedence and associativity shall be as described in [Table 4-1](#) (on page 2455).
- In case of ambiguity, an **else** shall be associated with the most immediately preceding **if** that would satisfy the grammar.
- In some contexts, a `<slash>` (`'/'`) that is used to surround an ERE could also be the division operator. This shall be resolved in such a way that wherever the division operator could appear, a `<slash>` is assumed to be the division operator. (There is no unary division operator.)

Each expression in an *awk* program shall conform to the precedence and associativity rules, even when this is not needed to resolve an ambiguity. For example, because `'$'` has higher precedence than `'++'`, the string `"$x++--"` is not a valid *awk* expression, even though it is unambiguously parsed by the grammar as `"$(x++)--"`.

One convention that might not be obvious from the formal grammar is where `<newline>` characters are acceptable. There are several obvious placements such as terminating a statement, and a `<backslash>` can be used to escape `<newline>` characters between any lexical tokens. In addition, `<newline>` characters without `<backslash>` characters can follow a comma, an open brace, logical AND operator (`"&&"`), logical OR operator (`"||"`), the **do** keyword, the **else** keyword, and the closing parenthesis of an **if**, **for**, or **while** statement. For example:

```

79490 { print $1,
79491       $2 }

```

Lexical Conventions

The lexical conventions for *awk* programs, with respect to the preceding grammar, shall be as follows:

1. Except as noted, *awk* shall recognize the longest possible token or delimiter beginning at a given point.
2. A comment shall consist of any characters beginning with the `<number-sign>` character and terminated by, but excluding the next occurrence of, a `<newline>`. Comments shall have no effect, except to delimit lexical tokens.
3. The `<newline>` shall be recognized as the token **NEWLINE**.
4. A `<backslash>` character immediately followed by a `<newline>` shall have no effect.
5. The token **STRING** shall represent a string constant. A string constant shall begin with the character `'`. Within a string constant, a `<backslash>` character shall be considered to begin an escape sequence as specified in the table in [XBD Chapter 5](#) (on page 121) (`'\'`, `'\a'`, `'\b'`, `'\f'`, `'\n'`, `'\r'`, `'\t'`, `'\v'`). In addition, the escape sequences in [Table 4-2](#) (on page 2462) shall be recognized. A `<newline>` shall not occur within a string constant. A string constant shall be terminated by the first unescaped occurrence of the character `'` after the one that begins the string constant. The value of the string

shall be the sequence of all unescaped characters and values of escape sequences between, but not including, the two delimiting ' " ' characters.

6. The token **ERE** represents an extended regular expression constant. An ERE constant shall begin with the <slash> character. Within an ERE constant, a <backslash> character shall be considered to begin an escape sequence as specified in the table in XBD [Chapter 5](#) (on page 121). In addition, the escape sequences in [Table 4-2](#) (on page 2462) shall be recognized. The application shall ensure that a <newline> does not occur within an ERE constant. An ERE constant shall be terminated by the first unescaped occurrence of the <slash> character after the one that begins the ERE constant. The extended regular expression represented by the ERE constant shall be the sequence of all unescaped characters and values of escape sequences between, but not including, the two delimiting <slash> characters.

7. A <blank> shall have no effect, except to delimit lexical tokens or within **STRING** or **ERE** tokens.

8. The token **NUMBER** shall represent a numeric constant. Its form and numeric value shall either be equivalent to the **decimal-floating-constant** token as specified by the ISO C standard, or it shall be a sequence of decimal digits and shall be evaluated as an integer constant in decimal. In addition, implementations may accept numeric constants with the form and numeric value equivalent to the **hexadecimal-constant** and **hexadecimal-floating-constant** tokens as specified by the ISO C standard.

If the value is too large or too small to be representable (see [Section 1.1.2](#), on page 2305), the behavior is undefined.

9. A sequence of underscores, digits, and alphabetic characters from the portable character set (see XBD [Section 6.1](#), on page 125), beginning with an <underscore> or alphabetic character, shall be considered a word.

10. The following words are keywords that shall be recognized as individual tokens; the name of the token is the same as the keyword:

BEGIN	delete	END	function	in	printf
break	do	exit	getline	next	return
continue	else	for	if	print	while

11. The following words are names of built-in functions and shall be recognized as the token **BUILTIN_FUNC_NAME**:

atan2	gsub	log	split	sub	toupper
close	index	match	sprintf	substr	
cos	int	rand	sqrt	system	
exp	length	sin	srand	tolower	

The above-listed keywords and names of built-in functions are considered reserved words.

12. The token **NAME** shall consist of a word that is not a keyword or a name of a built-in function and is not followed immediately (without any delimiters) by the ' (' character.
13. The token **FUNC_NAME** shall consist of a word that is not a keyword or a name of a built-in function, followed immediately (without any delimiters) by the ' (' character. The ' (' character shall not be included as part of the token.

14. The following two-character sequences shall be recognized as the named tokens:

Token Name	Sequence	Token Name	Sequence
ADD_ASSIGN	+=	NO_MATCH	!~
SUB_ASSIGN	-=	EQ	==
MUL_ASSIGN	*=	LE	<=
DIV_ASSIGN	/=	GE	>=
MOD_ASSIGN	%=	NE	!=
POW_ASSIGN	^=	INCR	++
OR		DECR	--
AND	&&	APPEND	>>

15. The following single characters shall be recognized as tokens whose names are the character:

<newline> { } () [] , ; + - * % ^ ! > < | ? : ~ \$ =

There is a lexical ambiguity between the token **ERE** and the tokens **'/'** and **DIV_ASSIGN**. When an input sequence begins with a <slash> character in any syntactic context where the token **'/'** or **DIV_ASSIGN** could appear as the next token in a valid program, the longer of those two tokens that can be recognized shall be recognized. In any other syntactic context where the token **ERE** could appear as the next token in a valid program, the token **ERE** shall be recognized.

EXIT STATUS

The following exit values shall be returned:

- 0 All input files were processed successfully.
- >0 An error occurred.

The exit status can be altered within the program by using an **exit** expression.

CONSEQUENCES OF ERRORS

If any *file* operand is specified and the named file cannot be accessed, *awk* shall write a diagnostic message to standard error and terminate without any further action.

If the program specified by either the *program* operand or a *progfile* operand is not a valid *awk* program (as specified in the EXTENDED DESCRIPTION section), the behavior is undefined.

APPLICATION USAGE

The **index**, **length**, **match**, and **substr** functions should not be confused with similar functions in the ISO C standard; the *awk* versions deal with characters, while the ISO C standard deals with bytes.

Because the concatenation operation is represented by adjacent expressions rather than an explicit operator, it is often necessary to use parentheses to enforce the proper evaluation precedence.

EXAMPLES

The *awk* program specified in the command line is most easily specified within single-quotes (for example, *'program'*) for applications using *sh*, because *awk* programs commonly contain characters that are special to the shell, including double-quotes. In the cases where an *awk* program contains single-quote characters, it is usually easiest to specify most of the program as strings within single-quotes concatenated by the shell with quoted single-quote characters. For example:

```
awk '/'\''/' { print "quote:", $0 }'
```

prints all lines from the standard input containing a single-quote character, prefixed with *quote*..

The following are examples of simple *awk* programs:

1. Write to the standard output all input lines for which field 3 is greater than 5:

```
$3 > 5
```

2. Write every tenth line:

```
(NR % 10) == 0
```

3. Write any line with a substring matching the regular expression:

```
/(G|D)(2[0-9][[:alpha:]]*)/
```

4. Print any line with a substring containing a 'G' or 'D', followed by a sequence of digits and characters. This example uses character classes **digit** and **alpha** to match language-independent digit and alphabetic characters respectively:

```
/(G|D)([[:digit:]][[:alpha:]]*)/
```

5. Write any line in which the second field matches the regular expression and the fourth field does not:

```
$2 ~ /xyz/ && $4 !~ /xyz/
```

6. Write any line in which the second field contains a <backslash>:

```
$2 ~ /\
```

7. Write any line in which the second field contains a <backslash>. Note that <backslash>-escapes are interpreted twice; once in lexical processing of the string and once in processing the regular expression:

```
$2 ~ "\\
```

8. Write the second to the last and the last field in each line. Separate the fields by a <colon>:

```
{OFS=":";print $(NF-1), $NF}
```

9. Write the line number and number of fields in each line. The three strings representing the line number, the <colon>, and the number of fields are concatenated and that string is written to standard output:

```
{print NR ":" NF}
```

10. Write lines longer than 72 characters:

```
length($0) > 72
```

11. Write the first two fields in opposite order separated by **OFS**:

```
{ print $2, $1 }
```

12. Same, with input fields separated by a <comma> or <space> and <tab> characters, or both:

```
BEGIN { FS = ",[ \t]*|[ \t]+" }
{ print $2, $1 }
```

13. Add up the first column, print sum, and average:

```
{s += $1 }
END {print "sum is ", s, " average is", s/NR}
```

14. Write fields in reverse order, one per line (many lines out for each line in):
- ```
{ for (i = NF; i > 0; --i) print $i }
```
15. Write all lines between occurrences of the strings **start** and **stop**:
- ```
/start/, /stop/
```
16. Write all lines whose first field is different from the previous one:
- ```
$1 != prev { print; prev = $1 }
```
17. Simulate *echo*:
- ```
BEGIN {
    for (i = 1; i < ARGV; ++i)
        printf("%s%s", ARGV[i], i==ARGC-1?"\n":" ")
}
```
18. Write the path prefixes contained in the *PATH* environment variable, one per line:
- ```
BEGIN {
 n = split (ENVIRON["PATH"], path, ":")
 for (i = 1; i <= n; ++i)
 print path[i]
}
```
19. If there is a file named **input** containing page headers of the form:
- ```
Page #
```
- and a file named **program** that contains:
- ```
/Page/ { $2 = n++; }
 { print }
```
- then the command line:
- ```
awk -f program n=5 input
```
- prints the file **input**, filling in page numbers starting at 5.

RATIONALE

This description is based on the new *awk*, “nawk”, (see the referenced *The AWK Programming Language*), which introduced a number of new features to the historical *awk*:

1. New keywords: **delete**, **do**, **function**, **return**
2. New built-in functions: **atan2**, **close**, **cos**, **gsub**, **match**, **rand**, **sin**, **srand**, **sub**, **system**
3. New predefined variables: **FNR**, **ARGC**, **ARGV**, **RSTART**, **RLENGTH**, **SUBSEP**
4. New expression operators: **?**, **:**, **..**, **^**
5. The **FS** variable and the third argument to **split**, now treated as extended regular expressions.
6. The operator precedence, changed to more closely match the C language. Two examples of code that operate differently are:

```
while ( n /= 10 > 1) ...
if (!"wk" ~ /bwk/) ...
```

Several features have been added based on newer implementations of *awk*:

- Multiple instances of `-f progfile` are permitted.
- The new option `-v assignment`.
- The new predefined variable `ENVIRON`.
- New built-in functions `toupper` and `tolower`.
- More formatting capabilities are added to `printf` to match the ISO C standard.

The overall *awk* syntax has always been based on the C language, with a few features from the shell command language and other sources. Because of this, it is not completely compatible with any other language, which has caused confusion for some users. It is not the intent of the standard developers to address such issues. A few relatively minor changes toward making the language more compatible with the ISO C standard were made; most of these changes are based on similar changes in recent implementations, as described above. There remain several C-language conventions that are not in *awk*. One of the notable ones is the `<comma>` operator, which is commonly used to specify multiple expressions in the C language `for` statement. Also, there are various places where *awk* is more restrictive than the C language regarding the type of expression that can be used in a given context. These limitations are due to the different features that the *awk* language does provide.

Regular expressions in *awk* have been extended somewhat from historical implementations to make them a pure superset of extended regular expressions, as defined by POSIX.1-2008 (see XBD Section 9.4, on page 188). The main extensions are internationalization features and interval expressions. Historical implementations of *awk* have long supported `<backslash>`-escape sequences as an extension to extended regular expressions, and this extension has been retained despite inconsistency with other utilities. The number of escape sequences recognized in both extended regular expressions and strings has varied (generally increasing with time) among implementations. The set specified by POSIX.1-2008 includes most sequences known to be supported by popular implementations and by the ISO C standard. One sequence that is not supported is hexadecimal value escapes beginning with `'\x'`. This would allow values expressed in more than 9 bits to be used within *awk* as in the ISO C standard. However, because this syntax has a non-deterministic length, it does not permit the subsequent character to be a hexadecimal digit. This limitation can be dealt with in the C language by the use of lexical string concatenation. In the *awk* language, concatenation could also be a solution for strings, but not for extended regular expressions (either lexical ERE tokens or strings used dynamically as regular expressions). Because of this limitation, the feature has not been added to POSIX.1-2008.

When a string variable is used in a context where an extended regular expression normally appears (where the lexical token ERE is used in the grammar) the string does not contain the literal `<slash>` characters.

Some versions of *awk* allow the form:

```
func name(args, ... ) { statements }
```

This has been deprecated by the authors of the language, who asked that it not be specified.

Historical implementations of *awk* produce an error if a **next** statement is executed in a **BEGIN** action, and cause *awk* to terminate if a **next** statement is executed in an **END** action. This behavior has not been documented, and it was not believed that it was necessary to standardize it.

The specification of conversions between string and numeric values is much more detailed than in the documentation of historical implementations or in the referenced *The AWK Programming Language*. Although most of the behavior is designed to be intuitive, the details are necessary to

ensure compatible behavior from different implementations. This is especially important in relational expressions since the types of the operands determine whether a string or numeric comparison is performed. From the perspective of an application developer, it is usually sufficient to expect intuitive behavior and to force conversions (by adding zero or concatenating a null string) when the type of an expression does not obviously match what is needed. The intent has been to specify historical practice in almost all cases. The one exception is that, in historical implementations, variables and constants maintain both string and numeric values after their original value is converted by any use. This means that referencing a variable or constant can have unexpected side-effects. For example, with historical implementations the following program:

```
{
    a = "+2"
    b = 2
    if (NR % 2)
        c = a + b
    if (a == b)
        print "numeric comparison"
    else
        print "string comparison"
}
```

would perform a numeric comparison (and output numeric comparison) for each odd-numbered line, but perform a string comparison (and output string comparison) for each even-numbered line. POSIX.1-2008 ensures that comparisons will be numeric if necessary. With historical implementations, the following program:

```
BEGIN {
    OFMT = "%e"
    print 3.14
    OFMT = "%f"
    print 3.14
}
```

would output "3.140000e+00" twice, because in the second **print** statement the constant "3.14" would have a string value from the previous conversion. POSIX.1-2008 requires that the output of the second **print** statement be "3.140000". The behavior of historical implementations was seen as too unintuitive and unpredictable.

It was pointed out that with the rules contained in early drafts, the following script would print nothing:

```
BEGIN {
    y[1.5] = 1
    OFMT = "%e"
    print y[1.5]
}
```

Therefore, a new variable, **CONVFMT**, was introduced. The **OFMT** variable is now restricted to affecting output conversions of numbers to strings and **CONVFMT** is used for internal conversions, such as comparisons or array indexing. The default value is the same as that for **OFMT**, so unless a program changes **CONVFMT** (which no historical program would do), it will receive the historical behavior associated with internal string conversions.

The POSIX *awk* lexical and syntactic conventions are specified more formally than in other sources. Again the intent has been to specify historical practice. One convention that may not be

obvious from the formal grammar as in other verbal descriptions is where <newline> characters are acceptable. There are several obvious placements such as terminating a statement, and a <backslash> can be used to escape <newline> characters between any lexical tokens. In addition, <newline> characters without <backslash> characters can follow a comma, an open brace, a logical AND operator ("&&"), a logical OR operator ("||"), the **do** keyword, the **else** keyword, and the closing parenthesis of an **if**, **for**, or **while** statement. For example:

```
{ print $1,
    $2 }
```

The requirement that *awk* add a trailing <newline> to the program argument text is to simplify the grammar, making it match a text file in form. There is no way for an application or test suite to determine whether a literal <newline> is added or whether *awk* simply acts as if it did.

POSIX.1-2008 requires several changes from historical implementations in order to support internationalization. Probably the most subtle of these is the use of the decimal-point character, defined by the *LC_NUMERIC* category of the locale, in representations of floating-point numbers. This locale-specific character is used in recognizing numeric input, in converting between strings and numeric values, and in formatting output. However, regardless of locale, the <period> character (the decimal-point character of the POSIX locale) is the decimal-point character recognized in processing *awk* programs (including assignments in command line arguments). This is essentially the same convention as the one used in the ISO C standard. The difference is that the C language includes the *setlocale()* function, which permits an application to modify its locale. Because of this capability, a C application begins executing with its locale set to the C locale, and only executes in the environment-specified locale after an explicit call to *setlocale()*. However, adding such an elaborate new feature to the *awk* language was seen as inappropriate for POSIX.1-2008. It is possible to execute an *awk* program explicitly in any desired locale by setting the environment in the shell.

The undefined behavior resulting from NULs in extended regular expressions allows future extensions for the GNU *gawk* program to process binary data.

The behavior in the case of invalid *awk* programs (including lexical, syntactic, and semantic errors) is undefined because it was considered overly limiting on implementations to specify. In most cases such errors can be expected to produce a diagnostic and a non-zero exit status. However, some implementations may choose to extend the language in ways that make use of certain invalid constructs. Other invalid constructs might be deemed worthy of a warning, but otherwise cause some reasonable behavior. Still other constructs may be very difficult to detect in some implementations. Also, different implementations might detect a given error during an initial parsing of the program (before reading any input files) while others might detect it when executing the program after reading some input. Implementors should be aware that diagnosing errors as early as possible and producing useful diagnostics can ease debugging of applications, and thus make an implementation more usable.

The unspecified behavior from using multi-character **RS** values is to allow possible future extensions based on extended regular expressions used for record separators. Historical implementations take the first character of the string and ignore the others.

Unspecified behavior when *split(string,array,<null>)* is used is to allow a proposed future extension that would split up a string into an array of individual characters.

In the context of the **getline** function, equally good arguments for different precedences of the **|** and **<** operators can be made. Historical practice has been that:

```
getline < "a" "b"
```

is parsed as:

```
79814 ( getline < "a" ) "b"
```

79815 although many would argue that the intent was that the file **ab** should be read. However:

```
79816 getline < "x" + 1
```

79817 parses as:

```
79818 getline < ( "x" + 1 )
```

79819 Similar problems occur with the **|** version of **getline**, particularly in combination with **\$**. For
79820 example:

```
79821 $"echo hi" | getline
```

79822 (This situation is particularly problematic when used in a **print** statement, where the **|getline**
79823 part might be a redirection of the **print**.)

79824 Since in most cases such constructs are not (or at least should not) be used (because they have a
79825 natural ambiguity for which there is no conventional parsing), the meaning of these constructs
79826 has been made explicitly unspecified. (The effect is that a conforming application that runs into
79827 the problem must parenthesize to resolve the ambiguity.) There appeared to be few if any actual
79828 uses of such constructs.

79829 Grammars can be written that would cause an error under these circumstances. Where
79830 backwards-compatibility is not a large consideration, implementors may wish to use such
79831 grammars.

79832 Some historical implementations have allowed some built-in functions to be called without an
79833 argument list, the result being a default argument list chosen in some “reasonable” way. Use of
79834 **length** as a synonym for **length(\$0)** is the only one of these forms that is thought to be widely
79835 known or widely used; this particular form is documented in various places (for example, most
79836 historical *awk* reference pages, although not in the referenced *The AWK Programming Language*) as
79837 legitimate practice. With this exception, default argument lists have always been undocumented
79838 and vaguely defined, and it is not at all clear how (or if) they should be generalized to user-
79839 defined functions. They add no useful functionality and preclude possible future extensions that
79840 might need to name functions without calling them. Not standardizing them seems the simplest
79841 course. The standard developers considered that **length** merited special treatment, however,
79842 since it has been documented in the past and sees possibly substantial use in historical
79843 programs. Accordingly, this usage has been made legitimate, but Issue 5 removed the
79844 obsolescent marking for XSI-conforming implementations and many otherwise conforming
79845 applications depend on this feature.

79846 In **sub** and **gsub**, if *repl* is a string literal (the lexical token **STRING**), then two consecutive
79847 <backslash> characters should be used in the string to ensure a single <backslash> will precede
79848 the <ampersand> when the resultant string is passed to the function. (For example, to specify
79849 one literal <ampersand> in the replacement string, use **gsub(ERE, "\\&")**.)

79850 Historically, the only special character in the *repl* argument of **sub** and **gsub** string functions was
79851 the <ampersand> (**'&'**) character and preceding it with the <backslash> character was used to
79852 turn off its special meaning.

79853 The description in the ISO POSIX-2:1993 standard introduced behavior such that the
79854 <backslash> character was another special character and it was unspecified whether there were
79855 any other special characters. This description introduced several portability problems, some of
79856 which are described below, and so it has been replaced with the more historical description.
79857 Some of the problems include:

- Historically, to create the replacement string, a script could use `gsub(ERE, "\\&")`, but with the ISO POSIX-2:1993 standard wording, it was necessary to use `gsub(ERE, "\\\&")`. The `<backslash>` characters are doubled here because all string literals are subject to lexical analysis, which would reduce each pair of `<backslash>` characters to a single `<backslash>` before being passed to `gsub`.
- Since it was unspecified what the special characters were, for portable scripts to guarantee that characters are printed literally, each character had to be preceded with a `<backslash>`. (For example, a portable script had to use `gsub(ERE, "\\h\\i")` to produce a replacement string of `"hi"`.)

The description for comparisons in the ISO POSIX-2:1993 standard did not properly describe historical practice because of the way numeric strings are compared as numbers. The current rules cause the following code:

```
if (0 == "000")
    print "strange, but true"
else
    print "not true"
```

to do a numeric comparison, causing the `if` to succeed. It should be intuitively obvious that this is incorrect behavior, and indeed, no historical implementation of *awk* actually behaves this way.

To fix this problem, the definition of *numeric string* was enhanced to include only those values obtained from specific circumstances (mostly external sources) where it is not possible to determine unambiguously whether the value is intended to be a string or a numeric.

Variables that are assigned to a numeric string shall also be treated as a numeric string. (For example, the notion of a numeric string can be propagated across assignments.) In comparisons, all variables having the uninitialized value are to be treated as a numeric operand evaluating to the numeric value zero.

Uninitialized variables include all types of variables including scalars, array elements, and fields. The definition of an uninitialized value in [Variables and Special Variables](#) (on page 2459) is necessary to describe the value placed on uninitialized variables and on fields that are valid (for example, `< $NF`) but have no characters in them and to describe how these variables are to be used in comparisons. A valid field, such as `$1`, that has no characters in it can be obtained from an input line of `"\t\t"` when `FS='\t'`. Historically, the comparison (`$1<10`) was done numerically after evaluating `$1` to the value zero.

The phrase "... also shall have the numeric value of the numeric string" was removed from several sections of the ISO POSIX-2:1993 standard because it specifies an unnecessary implementation detail. It is not necessary for POSIX.1-2008 to specify that these objects be assigned two different values. It is only necessary to specify that these objects may evaluate to two different values depending on context.

Historical implementations of *awk* did not parse hexadecimal integer or floating constants like `"0xa"` and `"0xap0"`. Due to an oversight, the 2001 through 2004 editions of this standard required support for hexadecimal floating constants. This was due to the reference to `atof()`. This version of the standard allows but does not require implementations to use `atof()` and includes a description of how floating-point numbers are recognized as an alternative to match historic behavior. The intent of this change is to allow implementations to recognize floating-point constants according to either the ISO/IEC 9899:1990 standard or ISO/IEC 9899:1999 standard, and to allow (but not require) implementations to recognize hexadecimal integer constants.

Historical implementations of *awk* did not support floating-point infinities and NaNs in *numeric*

strings; e.g., `"-INF"` and `"NaN"`. However, implementations that use the `atof()` or `strtod()` functions to do the conversion picked up support for these values if they used a ISO/IEC 9899:1999 standard version of the function instead of a ISO/IEC 9899:1990 standard version. Due to an oversight, the 2001 through 2004 editions of this standard did not allow support for infinities and NaNs, but in this revision support is allowed (but not required). This is a silent change to the behavior of *awk* programs; for example, in the POSIX locale the expression:

```
("-INF" + 0 < 0)
```

formerly had the value 0 because `"-INF"` converted to 0, but now it may have the value 0 or 1.

FUTURE DIRECTIONS

None.

SEE ALSO

[Section 1.3](#) (on page 2309), *grep*, *lex*, *sed*

XBD [Chapter 5](#) (on page 121), [Section 6.1](#) (on page 125), [Chapter 8](#) (on page 173), [Chapter 9](#) (on page 181), [Section 12.2](#) (on page 215)

XSH *atof()*, *exec*, *isspace()*, *popen()*, *setlocale()*, *strtod()*

CHANGE HISTORY

First released in Issue 2.

Issue 5

The FUTURE DIRECTIONS section is added.

Issue 6

The *awk* utility is aligned with the IEEE P1003.2b draft standard.

The normative text is reworded to avoid use of the term “must” for application requirements.

IEEE PASC Interpretation 1003.2 #211 is applied, adding the sentence “An occurrence of two consecutive `<backslash>` characters shall be interpreted as just a single literal `<backslash>` character.” into the description of the **sub** string function.

Issue 7

PASC Interpretation 1003.2-1992 #107 (SD5-XCU-ERN-73) is applied, updating the description of the **OFS** variable.

Austin Group Interpretation 1003.1-2001 #189 is applied.

Austin Group Interpretation 1003.1-2001 #201 is applied, permitting implementations to support infinities and NaNs.

SD5-XCU-ERN-79 is applied, restoring the horizontal lines to [Table 4-1](#) (on page 2455), and SD5-XCU-ERN-80 is applied, changing the order of some table entries.

SD5-XCU-ERN-87 is applied, updating the descriptive text of the Grammar.

SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

The EXTENDED DESCRIPTION is changed to make the support of hexadecimal integer and floating constants optional.

POSIX.1-2008, Technical Corrigendum 1, XCU/TC1-2008/0057 [224], XCU/TC1-2008/0058 [454], XCU/TC1-2008/0059 [224], XCU/TC1-2008/0060 [224], XCU/TC1-2008/0061 [254], XCU/TC1-2008/0062 [254], XCU/TC1-2008/0063 [224], and XCU/TC1-2008/0064 [454] are applied.

79946 **NAME**79947 **basename** — return non-directory portion of a pathname79948 **SYNOPSIS**79949 **basename** *string* [*suffix*]79950 **DESCRIPTION**

79951 The *string* operand shall be treated as a pathname, as defined in XBD [Section 3.267](#) (on page 75).
 79952 The string *string* shall be converted to the filename corresponding to the last pathname
 79953 component in *string* and then the suffix string *suffix*, if present, shall be removed. This shall be
 79954 done by performing actions equivalent to the following steps in order:

- 79955 1. If *string* is a null string, it is unspecified whether the resulting string is ' .' or a null
 79956 string. In either case, skip steps 2 through 6.
- 79957 2. If *string* is "/", it is implementation-defined whether steps 3 to 6 are skipped or
 79958 processed.
- 79959 3. If *string* consists entirely of <slash> characters, *string* shall be set to a single <slash>
 79960 character. In this case, skip steps 4 to 6.
- 79961 4. If there are any trailing <slash> characters in *string*, they shall be removed.
- 79962 5. If there are any <slash> characters remaining in *string*, the prefix of *string* up to and
 79963 including the last <slash> character in *string* shall be removed.
- 79964 6. If the *suffix* operand is present, is not identical to the characters remaining in *string*, and is
 79965 identical to a suffix of the characters remaining in *string*, the suffix *suffix* shall be removed
 79966 from *string*. Otherwise, *string* is not modified by this step. It shall not be considered an
 79967 error if *suffix* is not found in *string*.

79968 The resulting string shall be written to standard output.

79969 **OPTIONS**

79970 None.

79971 **OPERANDS**

79972 The following operands shall be supported:

79973 *string* A string.79974 *suffix* A string.79975 **STDIN**

79976 Not used.

79977 **INPUT FILES**

79978 None.

79979 **ENVIRONMENT VARIABLES**79980 The following environment variables shall affect the execution of *basename*:

79981 **LANG** Provide a default value for the internationalization variables that are unset or null.
 79982 (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization
 79983 variables used to determine the values of locale categories.)

79984 **LC_ALL** If set to a non-empty string value, override the values of all the other
 79985 internationalization variables.

79986 **LC_CTYPE** Determine the locale for the interpretation of sequences of bytes of text data as
 79987 characters (for example, single-byte as opposed to multi-byte characters in
 79988 arguments).

```

79989      LC_MESSAGES
79990          Determine the locale that should be used to affect the format and contents of
79991          diagnostic messages written to standard error.
79992 XSI      NLSPATH      Determine the location of message catalogs for the processing of LC_MESSAGES.
79993 ASYNCHRONOUS EVENTS
79994      Default.
79995 STDOUT
79996      The basename utility shall write a line to the standard output in the following format:
79997      "%s\n", <resulting string>
79998 STDERR
79999      The standard error shall be used only for diagnostic messages.
80000 OUTPUT FILES
80001      None.
80002 EXTENDED DESCRIPTION
80003      None.
80004 EXIT STATUS
80005      The following exit values shall be returned:
80006      0    Successful completion.
80007      >0   An error occurred.
80008 CONSEQUENCES OF ERRORS
80009      Default.
80010 APPLICATION USAGE
80011      The definition of pathname specifies implementation-defined behavior for pathnames starting
80012      with two <slash> characters. Therefore, applications shall not arbitrarily add <slash> characters
80013      to the beginning of a pathname unless they can ensure that there are more or less than two or are
80014      prepared to deal with the implementation-defined consequences.
80015 EXAMPLES
80016      If the string string is a valid pathname:
80017      $(basename -- "string")
80018      produces a filename that could be used to open the file named by string in the directory returned
80019      by:
80020      $(dirname -- "string")
80021      If the string string is not a valid pathname, the same algorithm is used, but the result need not be
80022      a valid filename. The basename utility is not expected to make any judgements about the validity
80023      of string as a pathname; it just follows the specified algorithm to produce a result string.
80024      The following shell script compiles /usr/src/cmd/cat.c and moves the output to a file named cat
80025      in the current directory when invoked with the argument /usr/src/cmd/cat or with the argument
80026      /usr/src/cmd/cat.c:
80027      c99 -- "$(dirname -- "$1")/$(basename -- "$1" .c) .c" &&
80028      mv a.out "$(basename -- "$1" .c)"

```

80029 **RATIONALE**

80030 The behaviors of *basename* and *dirname* have been coordinated so that when *string* is a valid
 80031 pathname:

80032 `$(basename -- "string")` |

80033 would be a valid filename for the file in the directory:

80034 `$(dirname -- "string")` |

80035 This would not work for the early proposal versions of these utilities due to the way it specified
 80036 handling of trailing <slash> characters.

80037 Since the definition of *pathname* specifies implementation-defined behavior for pathnames
 80038 starting with two <slash> characters, this volume of POSIX.1-2008 specifies similar
 80039 implementation-defined behavior for the *basename* and *dirname* utilities.

80040 **FUTURE DIRECTIONS**

80041 None.

80042 **SEE ALSO**

80043 [Section 2.5](#) (on page 2323), *dirname*

80044 XBD [Section 3.267](#) (on page 75), [Chapter 8](#) (on page 173)

80045 **CHANGE HISTORY**

80046 First released in Issue 2.

80047 **Issue 6**

80048 IEEE PASC Interpretation 1003.2 #164 is applied.

80049 The normative text is reworded to avoid use of the term “must” for application requirements.

80050 **Issue 7**

80051 POSIX.1-2008, Technical Corrigendum 1, XCU/TC1-2008/0065 [192,538], XCU/TC1-2008/0066 +
 80052 [192,538], and XCU/TC1-2008/0067 [192,430,538] are applied.

80053 **NAME**

80054 batch — schedule commands to be executed in a batch queue

80055 **SYNOPSIS**80056 *batch*80057 **DESCRIPTION**80058 The *batch* utility shall read commands from standard input and schedule them for execution in a
80059 batch queue. It shall be the equivalent of the command:

80060 at -q b -m now

80061 where queue *b* is a special *at* queue, specifically for batch jobs. Batch jobs shall be submitted to
80062 the batch queue with no time constraints and shall be run by the system using algorithms, based
80063 on unspecified factors, that may vary with each invocation of *batch*.80064 XSI Users shall be permitted to use *batch* if their name appears in the file **at.allow** which is located in
80065 an implementation-defined directory. If that file does not exist, the file **at.deny**, which is located
80066 in an implementation-defined directory, shall be checked to determine whether the user shall be
80067 denied access to *batch*. If neither file exists, only a process with appropriate privileges shall be
80068 allowed to submit a job. If only **at.deny** exists and is empty, global usage shall be permitted. The
80069 **at.allow** and **at.deny** files shall consist of one user name per line.80070 **OPTIONS**

80071 None.

80072 **OPERANDS**

80073 None.

80074 **STDIN**80075 The standard input shall be a text file consisting of commands acceptable to the shell command
80076 language described in [Chapter 2](#) (on page 2319).80077 **INPUT FILES**80078 XSI The text files **at.allow** and **at.deny**, which are located in an implementation-defined directory,
80079 shall contain zero or more user names, one per line, of users who are, respectively, authorized or
80080 denied access to the *at* and *batch* utilities.80081 **ENVIRONMENT VARIABLES**80082 The following environment variables shall affect the execution of *batch*:80083 **LANG** Provide a default value for the internationalization variables that are unset or null.
80084 (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization
80085 variables used to determine the values of locale categories.)80086 **LC_ALL** If set to a non-empty string value, override the values of all the other
80087 internationalization variables.80088 **LC_CTYPE** Determine the locale for the interpretation of sequences of bytes of text data as
80089 characters (for example, single-byte as opposed to multi-byte characters in
80090 arguments and input files).80091 **LC_MESSAGES**80092 Determine the locale that should be used to affect the format and contents of
80093 diagnostic messages written to standard error and informative messages written to
80094 standard output.80095 **LC_TIME** Determine the format and contents for date and time strings written by *batch*.

80096	XSI	NLSPATH	Determine the location of message catalogs for the processing of <i>LC_MESSAGES</i> .
80097		SHELL	Determine the name of a command interpreter to be used to invoke the at-job. If the variable is unset or null, <i>sh</i> shall be used. If it is set to a value other than a name for <i>sh</i> , the implementation shall do one of the following: use that shell; use <i>sh</i> ; use the login shell from the user database; any of the preceding accompanied by a warning diagnostic about which was chosen.
80098			
80099			
80100			
80101			
80102		TZ	Determine the timezone. The job shall be submitted for execution at the time specified by <i>timespec</i> or <i>-t time</i> relative to the timezone specified by the <i>TZ</i> variable. If <i>timespec</i> specifies a timezone, it overrides <i>TZ</i> . If <i>timespec</i> does not specify a timezone and <i>TZ</i> is unset or null, an unspecified default timezone shall be used.
80103			
80104			
80105			
80106			
80107		ASYNCHRONOUS EVENTS	
80108			Default.
80109		STDOUT	
80110			When standard input is a terminal, prompts of unspecified format for each line of the user input described in the STDIN section may be written to standard output.
80111			
80112		STDERR	
80113			The following shall be written to standard error when a job has been successfully submitted:
80114			"job %s at %s\n", <i>at_job_id</i> , < <i>date</i> >
80115			where <i>date</i> shall be equivalent in format to the output of:
80116			date +"%a %b %e %T %Y"
80117			The date and time written shall be adjusted so that they appear in the timezone of the user (as determined by the <i>TZ</i> variable).
80118			
80119			Neither this, nor warning messages concerning the selection of the command interpreter, are considered a diagnostic that changes the exit status.
80120			
80121			Diagnostic messages, if any, shall be written to standard error.
80122		OUTPUT FILES	
80123			None.
80124		EXTENDED DESCRIPTION	
80125			None.
80126		EXIT STATUS	
80127			The following exit values shall be returned:
80128			0 Successful completion.
80129			>0 An error occurred.
80130		CONSEQUENCES OF ERRORS	
80131			The job shall not be scheduled.

APPLICATION USAGE

It may be useful to redirect standard output within the specified commands.

EXAMPLES

1. This sequence can be used at a terminal:

```
batch
sort < file >outfile
EOT
```

2. This sequence, which demonstrates redirecting standard error to a pipe, is useful in a command procedure (the sequence of output redirection specifications is significant):

```
batch <<!
diff file1 file2 2>&1 >outfile | mailx mygroup
!
```

RATIONALE

Early proposals described *batch* in a manner totally separated from *at*, even though the historical model treated it almost as a synonym for *at -qb*. A number of features were added to list and control batch work separately from those in *at*. Upon further reflection, it was decided that the benefit of this did not merit the change to the historical interface.

The *-m* option was included on the equivalent *at* command because it is historical practice to mail results to the submitter, even if all job-produced output is redirected. As explained in the RATIONALE for *at*, the **now** keyword submits the job for immediate execution (after scheduling delays), despite some historical systems where *at now* would have been considered an error.

FUTURE DIRECTIONS

None.

SEE ALSO

at

XBD [Chapter 8](#) (on page 173)

CHANGE HISTORY

First released in Issue 2.

Issue 6

This utility is marked as part of the User Portability Utilities option.

The NAME is changed to align with the IEEE P1003.2b draft standard.

The normative text is reworded to avoid use of the term “must” for application requirements.

Issue 7

The *batch* utility is moved from the User Portability Utilities option to the Base. User Portability Utilities is now an option for interactive utilities.

SD5-XCU-ERN-95 is applied, removing the references to fixed locations for the files referenced by the *batch* utility.

80169 **NAME**80170 `bc` — arbitrary-precision arithmetic language80171 **SYNOPSIS**80172 `bc [-l] [file...]`80173 **DESCRIPTION**

80174 The `bc` utility shall implement an arbitrary precision calculator. It shall take input from any files
 80175 given, then read from the standard input. If the standard input and standard output to `bc` are
 80176 attached to a terminal, the invocation of `bc` shall be considered to be *interactive*, causing
 80177 behavioral constraints described in the following sections.

80178 **OPTIONS**80179 The `bc` utility shall conform to XBD [Section 12.2](#) (on page 215).

80180 The following option shall be supported:

80181 `-l` (The letter ell.) Define the math functions and initialize *scale* to 20, instead of the
 80182 default zero; see the EXTENDED DESCRIPTION section.

80183 **OPERANDS**

80184 The following operand shall be supported:

80185 *file* A pathname of a text file containing `bc` program statements. After all *files* have
 80186 been read, `bc` shall read the standard input.

80187 **STDIN**

80188 See the INPUT FILES section.

80189 **INPUT FILES**

80190 Input files shall be text files containing a sequence of comments, statements, and function
 80191 definitions that shall be executed as they are read.

80192 **ENVIRONMENT VARIABLES**80193 The following environment variables shall affect the execution of `bc`:

80194 *LANG* Provide a default value for the internationalization variables that are unset or null.
 80195 (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization
 80196 variables used to determine the values of locale categories.)

80197 *LC_ALL* If set to a non-empty string value, override the values of all the other
 80198 internationalization variables.

80199 *LC_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as
 80200 characters (for example, single-byte as opposed to multi-byte characters in
 80201 arguments and input files).

80202 *LC_MESSAGES*

80203 Determine the locale that should be used to affect the format and contents of
 80204 diagnostic messages written to standard error.

80205 *XSI* *NLSPATH* Determine the location of message catalogs for the processing of *LC_MESSAGES*.

80206 **ASYNCHRONOUS EVENTS**

80207 Default.

80208 **STDOUT**

80209 The output of the `bc` utility shall be controlled by the program read, and consist of zero or more
 80210 lines containing the value of all executed expressions without assignments. The radix and
 80211 precision of the output shall be controlled by the values of the **obase** and **scale** variables; see the
 80212 EXTENDED DESCRIPTION section.

80213 **STDERR**

80214 The standard error shall be used only for diagnostic messages.

80215 **OUTPUT FILES**

80216 None.

80217 **EXTENDED DESCRIPTION**80218 **Grammar**

80219 The grammar in this section and the lexical conventions in the following section shall together
 80220 describe the syntax for *bc* programs. The general conventions for this style of grammar are
 80221 described in [Section 1.3](#) (on page 2309). A valid program can be represented as the non-terminal
 80222 symbol **program** in the grammar. This formal syntax shall take precedence over the text syntax
 80223 description.

```

80224 %token      EOF NEWLINE STRING LETTER NUMBER
80225 %token      MUL_OP
80226 /*          '*', '/', '%'                               */
80227 %token      ASSIGN_OP
80228 /*          '=', '+=', '-=', '*=', '/=', '%=', '^=' */
80229 %token      REL_OP
80230 /*          '==', '<=', '>=', '!=', '<', '>'               */
80231 %token      INCR_DECR
80232 /*          '++', '--'                                   */
80233 %token      Define      Break      Quit      Length
80234 /*          'define', 'break', 'quit', 'length'         */
80235 %token      Return      For        If        While      Sqrt
80236 /*          'return', 'for', 'if', 'while', 'sqrt'      */
80237 %token      Scale      Ibase      Obase      Auto
80238 /*          'scale', 'ibase', 'obase', 'auto'           */
80239 %start      program
80240 %%
80241 program      : EOF
80242               | input_item program
80243               ;
80244 input_item   : semicolon_list NEWLINE
80245               | function
80246               ;
80247 semicolon_list : /* empty */
80248               | statement
80249               | semicolon_list ';' statement
80250               | semicolon_list ';'
80251               ;
80252 statement_list : /* empty */
80253               | statement
80254               | statement_list NEWLINE

```

```

80255         | statement_list NEWLINE statement
80256         | statement_list ';'
80257         | statement_list ';' statement
80258         ;
80259     statement      : expression
80260         | STRING
80261         | Break
80262         | Quit
80263         | Return
80264         | Return '(' return_expression ')'
80265         | For '(' expression ';'
80266             relational_expression ';'
80267             expression ')' statement
80268         | If '(' relational_expression ')' statement
80269         | While '(' relational_expression ')' statement
80270         | '{' statement_list '}'
80271         ;
80272     function        : Define LETTER '(' opt_parameter_list ')'
80273                     '{' NEWLINE opt_auto_define_list
80274                     statement_list '}'
80275                     ;
80276     opt_parameter_list : /* empty */
80277         | parameter_list
80278         ;
80279     parameter_list   : LETTER
80280         | define_list ',' LETTER
80281         ;
80282     opt_auto_define_list : /* empty */
80283         | Auto define_list NEWLINE
80284         | Auto define_list ';'
80285         ;
80286     define_list       : LETTER
80287         | LETTER '[' ']'
80288         | define_list ',' LETTER
80289         | define_list ',' LETTER '[' ']'
80290         ;
80291     opt_argument_list : /* empty */
80292         | argument_list
80293         ;
80294     argument_list     : expression
80295         | LETTER '[' ']' ',' argument_list
80296         ;
80297     relational_expression : expression
80298         | expression REL_OP expression
80299         ;
80300     return_expression  : /* empty */
80301         | expression

```

```

80302                                     ;
80303     expression                       : named_expression
80304                                     | NUMBER
80305                                     | '(' expression ')'
80306                                     | LETTER '(' opt_argument_list ')'
80307                                     | '-' expression
80308                                     | expression '+' expression
80309                                     | expression '-' expression
80310                                     | expression MUL_OP expression
80311                                     | expression '^' expression
80312                                     | INCR_DECR named_expression
80313                                     | named_expression INCR_DECR
80314                                     | named_expression ASSIGN_OP expression
80315                                     | Length '(' expression ')'
80316                                     | Sqrt '(' expression ')'
80317                                     | Scale '(' expression ')'
80318                                     ;
80319     named_expression                  : LETTER
80320                                     | LETTER '[' expression ']'
80321                                     | Scale
80322                                     | Ibase
80323                                     | Obase
80324                                     ;

```

80325 Lexical Conventions in bc

80326 The lexical conventions for *bc* programs, with respect to the preceding grammar, shall be as
 80327 follows:

- 80328 1. Except as noted, *bc* shall recognize the longest possible token or delimiter beginning at a
 80329 given point.
- 80330 2. A comment shall consist of any characters beginning with the two adjacent characters
 80331 `"/*"` and terminated by the next occurrence of the two adjacent characters `"*/"`.
 80332 Comments shall have no effect except to delimit lexical tokens.
- 80333 3. The <newline> shall be recognized as the token **NEWLINE**.
- 80334 4. The token **STRING** shall represent a string constant; it shall consist of any characters
 80335 beginning with the double-quote character (`'"`) and terminated by another occurrence
 80336 of the double-quote character. The value of the string is the sequence of all characters
 80337 between, but not including, the two double-quote characters. All characters shall be taken
 80338 literally from the input, and there is no way to specify a string containing a double-quote
 80339 character. The length of the value of each string shall be limited to {BC_STRING_MAX}
 80340 bytes.
- 80341 5. A <blank> shall have no effect except as an ordinary character if it appears within a
 80342 **STRING** token, or to delimit a lexical token other than **STRING**.
- 80343 6. The combination of a <backslash> character immediately followed by a <newline> shall
 80344 have no effect other than to delimit lexical tokens with the following exceptions:

- 80345 • It shall be interpreted as the character sequence "\<newline>" in **STRING** tokens.
- 80346 • It shall be ignored as part of a multi-line **NUMBER** token.
- 80347 7. The token **NUMBER** shall represent a numeric constant. It shall be recognized by the
- 80348 following grammar:
- ```

80349 NUMBER : integer
80350 | '.' integer
80351 | integer '.'
80352 | integer '.' integer
80353 ;

80354 integer : digit
80355 | integer digit
80356 ;

80357 digit : 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7
80358 | 8 | 9 | A | B | C | D | E | F
80359 ;

```
- 80360       8. The value of a **NUMBER** token shall be interpreted as a numeral in the base specified by
- 80361       the value of the internal register **ibase** (described below). Each of the **digit** characters
- 80362       shall have the value from 0 to 15 in the order listed here, and the <period> character shall
- 80363       represent the radix point. The behavior is undefined if digits greater than or equal to the
- 80364       value of **ibase** appear in the token. However, note the exception for single-digit values
- 80365       being assigned to **ibase** and **obase** themselves, in [Operations in bc](#) (on page 2498).
- 80366       9. The following keywords shall be recognized as tokens:
- ```

80367       auto       ibase       length   return   while
80368       break      if        obase     scale
80369       define     for       quit      sqrt

```
- 80370 10. Any of the following characters occurring anywhere except within a keyword shall be
- 80371 recognized as the token **LETTER**:
- ```

80372 a b c d e f g h i j k l m n o p q r s t u v w x y z

```
- 80373       11. The following single-character and two-character sequences shall be recognized as the
- 80374       token **ASSIGN\_OP**:
- ```

80375       =   +=   -=   *=   /=   %=   ^=

```
- 80376 12. If an '=' character, as the beginning of a token, is followed by a '-' character with no
- 80377 intervening delimiter, the behavior is undefined.
- 80378 13. The following single-characters shall be recognized as the token **MUL_OP**:
- ```

80379 * / %

```
- 80380       14. The following single-character and two-character sequences shall be recognized as the
- 80381       token **REL\_OP**:
- ```

80382       ==   <=   >=   !=   <   >

```
- 80383 15. The following two-character sequences shall be recognized as the token **INCR_DECR**:
- ```

80384 ++ --

```

16. The following single characters shall be recognized as tokens whose names are the character:

<newline> ( ) , + - ; [ ] ^ { }

17. The token **EOF** is returned when the end of input is reached.

### Operations in bc

There are three kinds of identifiers: ordinary identifiers, array identifiers, and function identifiers. All three types consist of single lowercase letters. Array identifiers shall be followed by square brackets ("[]"). An array subscript is required except in an argument or auto list. Arrays are singly dimensioned and can contain up to {BC\_DIM\_MAX} elements. Indexing shall begin at zero so an array is indexed from 0 to {BC\_DIM\_MAX}-1. Subscripts shall be truncated to integers. The application shall ensure that function identifiers are followed by parentheses, possibly enclosing arguments. The three types of identifiers do not conflict.

The following table summarizes the rules for precedence and associativity of all operators. Operators on the same line shall have the same precedence; rows are in order of decreasing precedence.

**Table 4-3** Operators in *bc*

| Operator                  | Associativity |
|---------------------------|---------------|
| ++, --                    | N/A           |
| unary -                   | N/A           |
| ^                         | Right to left |
| *, /, %                   | Left to right |
| +, binary -               | Left to right |
| =, +=, -=, *=, /=, %=, ^= | Right to left |
| ==, <=, >=, !=, <, >      | None          |

Each expression or named expression has a *scale*, which is the number of decimal digits that shall be maintained as the fractional portion of the expression.

*Named expressions* are places where values are stored. Named expressions shall be valid on the left side of an assignment. The value of a named expression shall be the value stored in the place named. Simple identifiers and array elements are named expressions; they have an initial value of zero and an initial scale of zero.

The internal registers **scale**, **ibase**, and **obase** are all named expressions. The scale of an expression consisting of the name of one of these registers shall be zero; values assigned to any of these registers are truncated to integers. The **scale** register shall contain a global value used in computing the scale of expressions (as described below). The value of the register **scale** is limited to  $0 \leq \text{scale} \leq \{\text{BC\_SCALE\_MAX}\}$  and shall have a default value of zero. The **ibase** and **obase** registers are the input and output number radix, respectively. The value of **ibase** shall be limited to:

$$2 \leq \text{ibase} \leq 16$$

The value of **obase** shall be limited to:

$$2 \leq \text{obase} \leq \{\text{BC\_BASE\_MAX}\}$$

When either **ibase** or **obase** is assigned a single **digit** value from the list in [Lexical Conventions in bc](#) (on page 2496), the value shall be assumed in hexadecimal. (For example, **ibase=A** sets to base ten, regardless of the current **ibase** value.) Otherwise, the behavior is undefined when



digits greater than or equal to the value of **ibase** appear in the input. Both **ibase** and **obase** shall have initial values of 10.

Internal computations shall be conducted as if in decimal, regardless of the input and output bases, to the specified number of decimal digits. When an exact result is not achieved (for example, **scale**=0; 3.2/1), the result shall be truncated.

For all values of **obase** specified by this volume of POSIX.1-2008, *bc* shall output numeric values by performing each of the following steps in order:

1. If the value is less than zero, a <hyphen> ( ' - ' ) character shall be output.
2. One of the following is output, depending on the numerical value:
  - If the absolute value of the numerical value is greater than or equal to one, the integer portion of the value shall be output as a series of digits appropriate to **obase** (as described below), most significant digit first. The most significant non-zero digit shall be output next, followed by each successively less significant digit.
  - If the absolute value of the numerical value is less than one but greater than zero and the scale of the numerical value is greater than zero, it is unspecified whether the character 0 is output.
  - If the numerical value is zero, the character 0 shall be output.
3. If the scale of the value is greater than zero and the numeric value is not zero, a <period> character shall be output, followed by a series of digits appropriate to **obase** (as described below) representing the most significant portion of the fractional part of the value. If *s* represents the scale of the value being output, the number of digits output shall be *s* if **obase** is 10, less than or equal to *s* if **obase** is greater than 10, or greater than or equal to *s* if **obase** is less than 10. For **obase** values other than 10, this should be the number of digits needed to represent a precision of  $10^s$ .

For **obase** values from 2 to 16, valid digits are the first **obase** of the single characters:

0 1 2 3 4 5 6 7 8 9 A B C D E F

which represent the values zero to 15, inclusive, respectively.

For bases greater than 16, each digit shall be written as a separate multi-digit decimal number. Each digit except the most significant fractional digit shall be preceded by a single <space>. For bases from 17 to 100, *bc* shall write two-digit decimal numbers; for bases from 101 to 1 000, three-digit decimal strings, and so on. For example, the decimal number 1 024 in base 25 would be written as:

Δ01Δ15Δ24

and in base 125, as:

Δ008Δ024

Very large numbers shall be split across lines with 70 characters per line in the POSIX locale; other locales may split at different character boundaries. Lines that are continued shall end with a <backslash>.

A function call shall consist of a function name followed by parentheses containing a <comma>-separated list of expressions, which are the function arguments. A whole array passed as an argument shall be specified by the array name followed by empty square brackets. All function arguments shall be passed by value. As a result, changes made to the formal parameters shall have no effect on the actual arguments. If the function terminates by executing

80471 a **return** statement, the value of the function shall be the value of the expression in the  
 80472 parentheses of the **return** statement or shall be zero if no expression is provided or if there is no  
 80473 **return** statement.

80474 The result of **sqrt**(*expression*) shall be the square root of the expression. The result shall be  
 80475 truncated in the least significant decimal place. The scale of the result shall be the scale of the  
 80476 expression or the value of **scale**, whichever is larger.

80477 The result of **length**(*expression*) shall be the total number of significant decimal digits in the  
 80478 expression. The scale of the result shall be zero.

80479 The result of **scale**(*expression*) shall be the scale of the expression. The scale of the result shall be  
 80480 zero.

80481 A numeric constant shall be an expression. The scale shall be the number of digits that follow the  
 80482 radix point in the input representing the constant, or zero if no radix point appears.

80483 The sequence ( *expression* ) shall be an expression with the same value and scale as *expression*.  
 80484 The parentheses can be used to alter the normal precedence.

80485 The semantics of the unary and binary operators are as follows:

80486 *-expression*  
 80487 The result shall be the negative of the *expression*. The scale of the result shall be the scale of  
 80488 *expression*.

80489 The unary increment and decrement operators shall not modify the scale of the named  
 80490 expression upon which they operate. The scale of the result shall be the scale of that named  
 80491 expression.

80492 *++named-expression*  
 80493 The named expression shall be incremented by one. The result shall be the value of the  
 80494 named expression after incrementing.

80495 *--named-expression*  
 80496 The named expression shall be decremented by one. The result shall be the value of the  
 80497 named expression after decrementing.

80498 *named-expression++*  
 80499 The named expression shall be incremented by one. The result shall be the value of the  
 80500 named expression before incrementing.

80501 *named-expression--*  
 80502 The named expression shall be decremented by one. The result shall be the value of the  
 80503 named expression before decrementing.

80504 The exponentiation operator, <circumflex> ( ' ^ ' ), shall bind right to left.

80505 *expression^expression*  
 80506 The result shall be the first *expression* raised to the power of the second *expression*. If the  
 80507 second expression is not an integer, the behavior is undefined. If *a* is the scale of the left  
 80508 expression and *b* is the absolute value of the right expression, the scale of the result shall be:  
 80509 if *b* >= 0 min(*a* \* *b*, max(scale, *a*)) if *b* < 0 scale

80510 The multiplicative operators ( ' \* ' , ' / ' , ' % ' ) shall bind left to right.

80511 *expression\*expression*  
 80512 The result shall be the product of the two expressions. If *a* and *b* are the scales of the two  
 80513 expressions, then the scale of the result shall be:

80514  $\min(a+b, \max(\text{scale}, a, b))$

80515 *expression*/*expression*

80516 The result shall be the quotient of the two expressions. The scale of the result shall be the  
80517 value of **scale**.

80518 *expression*%*expression*

80519 For expressions *a* and *b*, *a*%*b* shall be evaluated equivalent to the steps:

80520 1. Compute *a*/*b* to current scale.

80521 2. Use the result to compute:

80522  $a - (a / b) * b$

80523 to scale:

80524  $\max(\text{scale} + \text{scale}(b), \text{scale}(a))$

80525 The scale of the result shall be:

80526  $\max(\text{scale} + \text{scale}(b), \text{scale}(a))$

80527 When **scale** is zero, the '**%**' operator is the mathematical remainder operator.

80528 The additive operators ('+', '-') shall bind left to right.

80529 *expression*+*expression*

80530 The result shall be the sum of the two expressions. The scale of the result shall be the  
80531 maximum of the scales of the expressions.

80532 *expression*-*expression*

80533 The result shall be the difference of the two expressions. The scale of the result shall be the  
80534 maximum of the scales of the expressions.

80535 The assignment operators ('=', '+=', '==', '\*=', '/=', '%=', '^=') shall bind right to left.

80536 *named-expression*=*expression*

80537 This expression shall result in assigning the value of the expression on the right to the  
80538 named expression on the left. The scale of both the named expression and the result shall be  
80539 the scale of *expression*.

80540 The compound assignment forms:

80541 *named-expression* <operator>= *expression*

80542 shall be equivalent to:

80543 *named-expression*=*named-expression* <operator> *expression*

80544 except that the *named-expression* shall be evaluated only once.

80545 Unlike all other operators, the relational operators ('<', '>', '<=', '>=', '==', '!=') shall be  
80546 only valid as the object of an **if**, **while**, or inside a **for** statement.

80547 *expression1*<*expression2*

80548 The relation shall be true if the value of *expression1* is strictly less than the value of  
80549 *expression2*.

80550 *expression1*>*expression2*

80551 The relation shall be true if the value of *expression1* is strictly greater than the value of  
80552 *expression2*.

80553 *expression1* <= *expression2*  
 80554     The relation shall be true if the value of *expression1* is less than or equal to the value of  
 80555     *expression2*.

80556 *expression1* >= *expression2*  
 80557     The relation shall be true if the value of *expression1* is greater than or equal to the value of  
 80558     *expression2*.

80559 *expression1* = *expression2*  
 80560     The relation shall be true if the values of *expression1* and *expression2* are equal.

80561 *expression1* != *expression2*  
 80562     The relation shall be true if the values of *expression1* and *expression2* are unequal.

80563 There are only two storage classes in *bc*: global and automatic (local). Only identifiers that are  
 80564 local to a function need be declared with the **auto** command. The arguments to a function shall  
 80565 be local to the function. All other identifiers are assumed to be global and available to all  
 80566 functions. All identifiers, global and local, have initial values of zero. Identifiers declared as auto  
 80567 shall be allocated on entry to the function and released on returning from the function. They  
 80568 therefore do not retain values between function calls. Auto arrays shall be specified by the array  
 80569 name followed by empty square brackets. On entry to a function, the old values of the names  
 80570 that appear as parameters and as automatic variables shall be pushed onto a stack. Until the  
 80571 function returns, reference to these names shall refer only to the new values.

80572 References to any of these names from other functions that are called from this function also  
 80573 refer to the new value until one of those functions uses the same name for a local variable.

80574 When a statement is an expression, unless the main operator is an assignment, execution of the  
 80575 statement shall write the value of the expression followed by a <newline>.

80576 When a statement is a string, execution of the statement shall write the value of the string.

80577 Statements separated by <semicolon> or <newline> characters shall be executed sequentially. In  
 80578 an interactive invocation of *bc*, each time a <newline> is read that satisfies the grammatical  
 80579 production:

80580 `input_item : semicolon_list NEWLINE`

80581 the sequential list of statements making up the **semicolon\_list** shall be executed immediately  
 80582 and any output produced by that execution shall be written without any delay due to buffering.

80583 In an **if** statement (**if**(*relation*) *statement*), the *statement* shall be executed if the relation is true.

80584 The **while** statement (**while**(*relation*) *statement*) implements a loop in which the *relation* is tested;  
 80585 each time the *relation* is true, the *statement* shall be executed and the *relation* retested. When the  
 80586 *relation* is false, execution shall resume after *statement*.

80587 A **for** statement(**for**(*expression*; *relation*; *expression*) *statement*) shall be the same as:

80588 `first-expression`  
 80589 `while (relation) {`  
 80590     `statement`  
 80591     `last-expression`  
 80592 `}`

80593 The application shall ensure that all three expressions are present.

80594 The **break** statement shall cause termination of a **for** or **while** statement.

80595 The **auto** statement (**auto** *identifier* [*identifier*] ...) shall cause the values of the identifiers to be

pushed down. The identifiers can be ordinary identifiers or array identifiers. Array identifiers shall be specified by following the array name by empty square brackets. The application shall ensure that the **auto** statement is the first statement in a function definition.

A **define** statement:

```
define LETTER (opt_parameter_list) {
 opt_auto_define_list
 statement_list
}
```

defines a function named **LETTER**. If a function named **LETTER** was previously defined, the **define** statement shall replace the previous definition. The expression:

```
LETTER (opt_argument_list)
```

shall invoke the function named **LETTER**. The behavior is undefined if the number of arguments in the invocation does not match the number of parameters in the definition. Functions shall be defined before they are invoked. A function shall be considered to be defined within its own body, so recursive calls are valid. The values of numeric constants within a function shall be interpreted in the base specified by the value of the **ibase** register when the function is invoked.

The **return** statements (**return** and **return(expression)**) shall cause termination of a function, popping of its auto variables, and specification of the result of the function. The first form shall be equivalent to **return(0)**. The value and scale of the result returned by the function shall be the value and scale of the expression returned.

The **quit** statement (**quit**) shall stop execution of a *bc* program at the point where the statement occurs in the input, even if it occurs in a function definition, or in an **if**, **for**, or **while** statement.

The following functions shall be defined when the **-l** option is specified:

```
s(expression)
 Sine of argument in radians.

c(expression)
 Cosine of argument in radians.

a(expression)
 Arctangent of argument.

l(expression)
 Natural logarithm of argument.

e(expression)
 Exponential function of argument.

j(expression, expression)
 Bessel function of integer order.
```

The scale of the result returned by these functions shall be the value of the **scale** register at the time the function is invoked. The value of the **scale** register after these functions have completed their execution shall be the same value it had upon invocation. The behavior is undefined if any of these functions is invoked with an argument outside the domain of the mathematical function.

**EXIT STATUS**

The following exit values shall be returned:

0 All input files were processed successfully.

*unspecified* An error occurred.

**CONSEQUENCES OF ERRORS**

If any *file* operand is specified and the named file cannot be accessed, *bc* shall write a diagnostic message to standard error and terminate without any further action.

In an interactive invocation of *bc*, the utility should print an error message and recover following any error in the input. In a non-interactive invocation of *bc*, invalid input causes undefined behavior.

**APPLICATION USAGE**

Automatic variables in *bc* do not work in exactly the same way as in either C or PL/1.

For historical reasons, the exit status from *bc* cannot be relied upon to indicate that an error has occurred. Returning zero after an error is possible. Therefore, *bc* should be used primarily by interactive users (who can react to error messages) or by application programs that can somehow validate the answers returned as not including error messages.

The *bc* utility always uses the <period> ( ' . ' ) character to represent a radix point, regardless of any decimal-point character specified as part of the current locale. In languages like C or *awk*, the <period> character is used in program source, so it can be portable and unambiguous, while the locale-specific character is used in input and output. Because there is no distinction between source and input in *bc*, this arrangement would not be possible. Using the locale-specific character in *bc*'s input would introduce ambiguities into the language; consider the following example in a locale with a <comma> as the decimal-point character:

```
define f(a,b) {
 ...
}
...
f(1,2,3)
```

Because of such ambiguities, the <period> character is used in input. Having input follow different conventions from output would be confusing in either pipeline usage or interactive usage, so the <period> is also used in output.

**EXAMPLES**

In the shell, the following assigns an approximation of the first ten digits of '  $\pi$  ' to the variable *x*:

```
x=$(printf "%s\n" 'scale = 10; 104348/33215' | bc)
```

The following *bc* program prints the same approximation of '  $\pi$  ', with a label, to standard output:

```
scale = 10
"pi equals "
104348 / 33215
```

The following defines a function to compute an approximate value of the exponential function (note that such a function is predefined if the *-l* option is specified):

```
scale = 20
define e(x){
 auto a, b, c, i, s
```

```

80681 a = 1
80682 b = 1
80683 s = 1
80684 for (i = 1; 1 == 1; i++){
80685 a = a*x
80686 b = b*i
80687 c = a/b
80688 if (c == 0) {
80689 return(s)
80690 }
80691 s = s+c
80692 }
80693 }

```

80694 The following prints approximate values of the exponential function of the first ten integers:

```

80695 for (i = 1; i <= 10; ++i) {
80696 e(i)
80697 }

```

## 80698 RATIONALE

80699 The *bc* utility is implemented historically as a front-end processor for *dc*; *dc* was not selected to  
80700 be part of this volume of POSIX.1-2008 because *bc* was thought to have a more intuitive  
80701 programmatic interface. Current implementations that implement *bc* using *dc* are expected to be  
80702 compliant.

80703 The exit status for error conditions has been left unspecified for several reasons:

- 80704 • The *bc* utility is used in both interactive and non-interactive situations. Different exit codes  
80705 may be appropriate for the two uses.
- 80706 • It is unclear when a non-zero exit should be given; divide-by-zero, undefined functions,  
80707 and syntax errors are all possibilities.
- 80708 • It is not clear what utility the exit status has.
- 80709 • In the 4.3 BSD, System V, and Ninth Edition implementations, *bc* works in conjunction with  
80710 *dc*. The *dc* utility is the parent, *bc* is the child. This was done to cleanly terminate *bc* if *dc*  
80711 aborted.

80712 The decision to have *bc* exit upon encountering an inaccessible input file is based on the belief  
80713 that *bc file1 file2* is used most often when at least *file1* contains data/function  
80714 declarations/initializations. Having *bc* continue with prerequisite files missing is probably not  
80715 useful. There is no implication in the CONSEQUENCES OF ERRORS section that *bc* must check  
80716 all its files for accessibility before opening any of them.

80717 There was considerable debate on the appropriateness of the language accepted by *bc*. Several  
80718 reviewers preferred to see either a pure subset of the C language or some changes to make the  
80719 language more compatible with C. While the *bc* language has some obvious similarities to C, it  
80720 has never claimed to be compatible with any version of C. An interpreter for a subset of C might  
80721 be a very worthwhile utility, and it could potentially make *bc* obsolete. However, no such utility  
80722 is known in historical practice, and it was not within the scope of this volume of POSIX.1-2008 to  
80723 define such a language and utility. If and when they are defined, it may be appropriate to  
80724 include them in a future version of this standard. This left the following alternatives:

1. Exclude any calculator language from this volume of POSIX.1-2008.

The consensus of the standard developers was that a simple programmatic calculator language is very useful for both applications and interactive users. The only arguments for excluding any calculator were that it would become obsolete if and when a C-compatible one emerged, or that the absence would encourage the development of such a C-compatible one. These arguments did not sufficiently address the needs of current application developers.

2. Standardize the historical *dc*, possibly with minor modifications.

The consensus of the standard developers was that *dc* is a fundamentally less usable language and that that would be far too severe a penalty for avoiding the issue of being similar to but incompatible with C.

3. Standardize the historical *bc*, possibly with minor modifications.

This was the approach taken. Most of the proponents of changing the language would not have been satisfied until most or all of the incompatibilities with C were resolved. Since most of the changes considered most desirable would break historical applications and require significant modification to historical implementations, almost no modifications were made. The one significant modification that was made was the replacement of the historical *bc* assignment operators "*=+*", and so on, with the more modern "*+=*", and so on. The older versions are considered to be fundamentally flawed because of the lexical ambiguity in uses like *a=-1*.

In order to permit implementations to deal with backwards-compatibility as they see fit, the behavior of this one ambiguous construct was made undefined. (At least three implementations have been known to support this change already, so the degree of change involved should not be great.)

The '*%*' operator is the mathematical remainder operator when **scale** is zero. The behavior of this operator for other values of **scale** is from historical implementations of *bc*, and has been maintained for the sake of historical applications despite its non-intuitive nature.

Historical implementations permit setting **ibase** and **obase** to a broader range of values. This includes values less than 2, which were not seen as sufficiently useful to standardize. These implementations do not interpret input properly for values of **ibase** that are greater than 16. This is because numeric constants are recognized syntactically, rather than lexically, as described in this volume of POSIX.1-2008. They are built from lexical tokens of single hexadecimal digits and <period> characters. Since <blank> characters between tokens are not visible at the syntactic level, it is not possible to recognize the multi-digit "digits" used in the higher bases properly. The ability to recognize input in these bases was not considered useful enough to require modifying these implementations. Note that the recognition of numeric constants at the syntactic level is not a problem with conformance to this volume of POSIX.1-2008, as it does not impact the behavior of conforming applications (and correct *bc* programs). Historical implementations also accept input with all of the digits '*0*'–'*9*' and '*A*'–'*F*' regardless of the value of **ibase**; since digits with value greater than or equal to **ibase** are not really appropriate, the behavior when they appear is undefined, except for the common case of:

```
ibase=8;
/* Process in octal base. */
...
ibase=A
/* Restore decimal base. */
```

In some historical implementations, if the expression to be written is an uninitialized array



80772 element, a leading <space> and/or up to four leading 0 characters may be output before the  
80773 character zero. This behavior is considered a bug; it is unlikely that any currently conforming  
80774 application relies on:

80775 `echo 'b[3]' | bc`

80776 returning 00000 rather than 0.

80777 Exact calculation of the number of fractional digits to output for a given value in a base other  
80778 than 10 can be computationally expensive. Historical implementations use a faster  
80779 approximation, and this is permitted. Note that the requirements apply only to values of **obase**  
80780 that this volume of POSIX.1-2008 requires implementations to support (in particular, not to 1, 0,  
80781 or negative bases, if an implementation supports them as an extension).

80782 Historical implementations of *bc* did not allow array parameters to be passed as the last  
80783 parameter to a function. New implementations are encouraged to remove this restriction even  
80784 though it is not required by the grammar.

#### 80785 **FUTURE DIRECTIONS**

80786 None.

#### 80787 **SEE ALSO**

80788 [Section 1.3](#) (on page 2309), *awk*

80789 XBD [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 215)

#### 80790 **CHANGE HISTORY**

80791 First released in Issue 4.

#### 80792 **Issue 5**

80793 The FUTURE DIRECTIONS section is added.

#### 80794 **Issue 6**

80795 Updated to align with the IEEE P1003.2b draft standard, which included resolution of several  
80796 interpretations of the ISO POSIX-2: 1993 standard.

80797 The normative text is reworded to avoid use of the term “must” for application requirements.

#### 80798 **Issue 7**

80799 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

80800 **NAME**

80801           bg — run jobs in the background

80802 **SYNOPSIS**80803 UP       bg [*job\_id*...]80804 **DESCRIPTION**

80805       If job control is enabled (see the description of *set -m*), the *bg* utility shall resume suspended jobs  
 80806       from the current environment (see [Section 2.12](#), on page 2353) by running them as background  
 80807       jobs. If the job specified by *job\_id* is already a running background job, the *bg* utility shall have  
 80808       no effect and shall exit successfully.

80809       Using *bg* to place a job into the background shall cause its process ID to become “known in the  
 80810       current shell execution environment”, as if it had been started as an asynchronous list; see  
 80811       [Section 2.9.3.1](#) (on page 2342).

80812 **OPTIONS**

80813       None.

80814 **OPERANDS**

80815       The following operand shall be supported:

80816       *job\_id*       Specify the job to be resumed as a background job. If no *job\_id* operand is given,  
 80817       the most recently suspended job shall be used. The format of *job\_id* is described in  
 80818       XBD [Section 3.204](#) (on page 65).

80819 **STDIN**

80820       Not used.

80821 **INPUT FILES**

80822       None.

80823 **ENVIRONMENT VARIABLES**80824       The following environment variables shall affect the execution of *bg*:

80825       *LANG*       Provide a default value for the internationalization variables that are unset or null.  
 80826       (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 80827       variables used to determine the values of locale categories.)

80828       *LC\_ALL*       If set to a non-empty string value, override the values of all the other  
 80829       internationalization variables.

80830       *LC\_CTYPE*   Determine the locale for the interpretation of sequences of bytes of text data as  
 80831       characters (for example, single-byte as opposed to multi-byte characters in  
 80832       arguments).

80833       *LC\_MESSAGES*

80834       Determine the locale that should be used to affect the format and contents of  
 80835       diagnostic messages written to standard error.

80836 XSI       *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

80837 **ASYNCHRONOUS EVENTS**

80838       Default.

80839 **STDOUT**80840       The output of *bg* shall consist of a line in the format:80841       "[*%d*] *%s*\n", <*job-number*>, <*command*>

80842 where the fields are as follows:

80843 <job-number> A number that can be used to identify the job to the *wait*, *fg*, and *kill* utilities. Using  
80844 these utilities, the job can be identified by prefixing the job number with ' % '.

80845 <command> The associated command that was given to the shell.

#### 80846 **STDERR**

80847 The standard error shall be used only for diagnostic messages.

#### 80848 **OUTPUT FILES**

80849 None.

#### 80850 **EXTENDED DESCRIPTION**

80851 None.

#### 80852 **EXIT STATUS**

80853 The following exit values shall be returned:

80854 0 Successful completion.

80855 >0 An error occurred.

#### 80856 **CONSEQUENCES OF ERRORS**

80857 If job control is disabled, the *bg* utility shall exit with an error and no job shall be placed in the  
80858 background.

#### 80859 **APPLICATION USAGE**

80860 A job is generally suspended by typing the SUSP character (<control>-Z on most systems); see  
80861 XBD [Chapter 11](#) (on page 199). At that point, *bg* can put the job into the background. This is  
80862 most effective when the job is expecting no terminal input and its output has been redirected to  
80863 non-terminal files. A background job can be forced to stop when it has terminal output by  
80864 issuing the command:

80865 `stty tostop`

80866 A background job can be stopped with the command:

80867 `kill -s stop job ID`

80868 The *bg* utility does not work as expected when it is operating in its own utility execution  
80869 environment because that environment has no suspended jobs. In the following examples:

80870 `... | xargs bg`  
80871 `(bg)`

80872 each *bg* operates in a different environment and does not share its parent shell's understanding  
80873 of jobs. For this reason, *bg* is generally implemented as a shell regular built-in.

#### 80874 **EXAMPLES**

80875 None.

#### 80876 **RATIONALE**

80877 The extensions to the shell specified in this volume of POSIX.1-2008 have mostly been based on  
80878 features provided by the KornShell. The job control features provided by *bg*, *fg*, and *jobs* are also  
80879 based on the KornShell. The standard developers examined the characteristics of the C shell  
80880 versions of these utilities and found that differences exist. Despite widespread use of the C shell,  
80881 the KornShell versions were selected for this volume of POSIX.1-2008 to maintain a degree of  
80882 uniformity with the rest of the KornShell features selected (such as the very popular command  
80883 line editing features).

80884 The *bg* utility is expected to wrap its output if the output exceeds the number of display  
80885 columns.

80886 **FUTURE DIRECTIONS**

80887 None.

80888 **SEE ALSO**

80889 [Section 2.9.3.1](#) (on page 2342), *fg*, *kill*, *jobs*, *wait*

80890 XBD [Section 3.204](#) (on page 65), [Chapter 8](#) (on page 173), [Chapter 11](#) (on page 199)

80891 **CHANGE HISTORY**

80892 First released in Issue 4.

80893 **Issue 6**

80894 This utility is marked as part of the User Portability Utilities option.

80895 The JC margin marker on the SYNOPSIS is removed since support for Job Control is mandatory  
80896 in this version. This is a FIPS requirement.

80897 **Issue 7**

80898 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

**NAME**

c99 — compile standard C programs

**SYNOPSIS**

```
CD c99 [options...] pathname [[pathname] [-I directory]
 [-L directory] [-l library]]...
```

**DESCRIPTION**

The *c99* utility is an interface to the standard C compilation system; it shall accept source code conforming to the ISO C standard. The system conceptually consists of a compiler and link editor. The input files referenced by *pathname* operands and *-I* option-arguments shall be compiled and linked to produce an executable file. (It is unspecified whether the linking occurs entirely within the operation of *c99*; some implementations may produce objects that are not fully resolved until the file is executed.)

If the *-c* option is specified, for all *pathname* operands of the form *file.c*, the files:

```
$(basename pathname .c).o
```

shall be created as the result of successful compilation. If the *-c* option is not specified, it is unspecified whether such *.o* files are created or deleted for the *file.c* operands.

If there are no options that prevent link editing (such as *-c* or *-E*), and all input files compile and link without error, the resulting executable file shall be written according to the *-o outfile* option (if present) or to the file *a.out*.

The executable file shall be created as specified in [Section 1.1.1.4](#) (on page 2302), except that the file permission bits shall be set to:

```
S_IRWXO | S_IRWXG | S_IRWXU
```

and the bits specified by the *umask* of the process shall be cleared.

**OPTIONS**

The *c99* utility shall conform to XBD [Section 12.2](#) (on page 215), except that:

- Options can be interspersed with operands.
- The order of specifying the *-L* and *-I* options, and the order of specifying *-l* options with respect to *pathname* operands is significant.
- Conforming applications shall specify each option separately; that is, grouping option letters (for example, *-cO*) need not be recognized by all implementations.

The following options shall be supported:

**-c** Suppress the link-edit phase of the compilation, and do not remove any object files that are produced.

**-D name[=value]**

Define *name* as if by a C-language **#define** directive. If no *=value* is given, a value of 1 shall be used. The *-D* option has lower precedence than the *-U* option. That is, if *name* is used in both a *-U* and a *-D* option, *name* shall be undefined regardless of the order of the options. Additional implementation-defined *names* may be provided by the compiler. Implementations shall support at least 2 048 bytes of *-D* definitions and 256 *names*.

- 80939        **-E**            Copy C-language source files to standard output, expanding all preprocessor  
80940 directives; no compilation shall be performed. If any operand is not a text file, the  
80941 effects are unspecified.
- 80942        **-g**            Produce symbolic information in the object or executable files; the nature of this  
80943 information is unspecified, and may be modified by implementation-defined  
80944 interactions with other options.
- 80945        **-I *directory***    Change the algorithm for searching for headers whose names are not absolute  
80946 pathnames to look in the directory named by the *directory* pathname before looking  
80947 in the usual places. Thus, headers whose names are enclosed in double-quotes ("")  
80948 shall be searched for first in the directory of the file with the **#include** line, then in  
80949 directories named in **-I** options, and last in the usual places. For headers whose  
80950 names are enclosed in angle brackets ("*< >*"), the header shall be searched for only  
80951 in directories named in **-I** options and then in the usual places. Directories named  
80952 in **-I** options shall be searched in the order specified. If the **-I** option is used to  
80953 specify a directory that is one of the usual places searched by default, the results  
80954 are unspecified. Implementations shall support at least ten instances of this option  
80955 in a single *c99* command invocation.
- 80956        **-L *directory***    Change the algorithm of searching for the libraries named in the **-l** objects to look  
80957 in the directory named by the *directory* pathname before looking in the usual  
80958 places. Directories named in **-L** options shall be searched in the order specified. If  
80959 the **-L** option is used to specify a directory that is one of the usual places searched  
80960 by default, the results are unspecified. Implementations shall support at least ten  
80961 instances of this option in a single *c99* command invocation. If a directory specified  
80962 by a **-L** option contains files with names starting with any of the strings "*libc.*",  
80963 "*libl.*", "*libpthread.*", "*libm.*", "*librt.*", "*libtrace.*", "*libxnet.*",  
80964 or "*liby.*", the results are unspecified.
- 80965        **-l *library***    Search the library named **liblibrary.a**. A library shall be searched when its name is  
80966 encountered, so the placement of a **-l** option is significant. Several standard  
80967 libraries can be specified in this manner, as described in the EXTENDED  
80968 DESCRIPTION section. Implementations may recognize implementation-defined  
80969 suffixes other than **.a** as denoting libraries.
- 80970        **-O *optlevel***    Specify the level of code optimization. If the *optlevel* option-argument is the digit  
80971 '*0*', all special code optimizations shall be disabled. If it is the digit '*1*', the  
80972 nature of the optimization is unspecified. If the **-O** option is omitted, the nature of  
80973 the system's default optimization is unspecified. It is unspecified whether code  
80974 generated in the presence of the **-O 0** option is the same as that generated when  
80975 **-O** is omitted. Other *optlevel* values may be supported.
- 80976        **-o *outfile***    Use the pathname *outfile*, instead of the default **a.out**, for the executable file  
80977 produced. If the **-o** option is present with **-c** or **-E**, the result is unspecified.
- 80978        **-s**            Produce object or executable files, or both, from which symbolic and other  
80979 information not required for proper execution using the *exec* family defined in the  
80980 System Interfaces volume of POSIX.1-2008 has been removed (stripped). If both **-g**  
80981 and **-s** options are present, the action taken is unspecified.
- 80982        **-U *name***        Remove any initial definition of *name*.
- 80983        Multiple instances of the **-D**, **-I**, **-L**, **-l**, and **-U** options can be specified.

**OPERANDS**

The application shall ensure that at least one *pathname* operand is specified. The following forms for *pathname* operands shall be supported:

*file.c* A C-language source file to be compiled and optionally linked. The application shall ensure that the operand is of this form if the `-c` option is used.

*file.a* A library of object files typically produced by the *ar* utility, and passed directly to the link editor. Implementations may recognize implementation-defined suffixes other than *.a* as denoting object file libraries.

*file.o* An object file produced by *c99 -c* and passed directly to the link editor. Implementations may recognize implementation-defined suffixes other than *.o* as denoting object files.

The processing of other files is implementation-defined.

**STDIN**

Not used.

**INPUT FILES**

Each input file shall be one of the following: a text file containing a C-language source program, an object file in the format produced by *c99 -c*, or a library of object files, in the format produced by archiving zero or more object files, using *ar*. Implementations may supply additional utilities that produce files in these formats. Additional input file formats are implementation-defined.

**ENVIRONMENT VARIABLES**

The following environment variables shall affect the execution of *c99*:

*LANG* Provide a default value for the internationalization variables that are unset or null. (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization variables used to determine the values of locale categories.)

*LC\_ALL* If set to a non-empty string value, override the values of all the other internationalization variables.

*LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as characters (for example, single-byte as opposed to multi-byte characters in arguments and input files).

*LC\_MESSAGES*

Determine the locale that should be used to affect the format and contents of diagnostic messages written to standard error.

*NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

*TMPDIR* Provide a pathname that should override the default directory for temporary files, if any. On XSI-conforming systems, provide a pathname that shall override the default directory for temporary files, if any.

**ASYNCHRONOUS EVENTS**

Default.

**STDOUT**

If more than one *pathname* operand ending in *.c* (or possibly other unspecified suffixes) is given, for each such file:

"%s:\n", *<pathname>*

may be written. These messages, if written, shall precede the processing of each input file; they

81027 shall not be written to the standard output if they are written to the standard error, as described  
81028 in the STDERR section.

81029 If the **-E** option is specified, the standard output shall be a text file that represents the results of  
81030 the preprocessing stage of the language; it may contain extra information appropriate for  
81031 subsequent compilation passes.

## 81032 **STDERR**

81033 The standard error shall be used only for diagnostic messages. If more than one *pathname*  
81034 operand ending in **.c** (or possibly other unspecified suffixes) is given, for each such file:

81035 `"%s:\n", <pathname>`

81036 may be written to allow identification of the diagnostic and warning messages with the  
81037 appropriate input file. These messages, if written, shall precede the processing of each input file;  
81038 they shall not be written to the standard error if they are written to the standard output, as  
81039 described in the STDOUT section.

81040 This utility may produce warning messages about certain conditions that do not warrant  
81041 returning an error (non-zero) exit value.

## 81042 **OUTPUT FILES**

81043 Object files or executable files or both are produced in unspecified formats. If the pathname of  
81044 an object file or executable file to be created by *c99* resolves to an existing directory entry for a  
81045 file that is not a regular file, it is unspecified whether *c99* shall attempt to create the file or shall  
81046 issue a diagnostic and exit with a non-zero exit status.

## 81047 **EXTENDED DESCRIPTION**

### 81048 **Standard Libraries**

81049 The *c99* utility shall recognize the following **-l** options for standard libraries:

81050 **-l c** This option shall make available all interfaces referenced in the System Interfaces  
81051 volume of POSIX.1-2008, with the possible exception of those interfaces listed as  
81052 residing in **<aio.h>**, **<arpa/inet.h>**, **<complex.h>**, **<fenv.h>**, **<math.h>**,  
81053 **<mqueue.h>**, **<netdb.h>**, **<net/if.h>**, **<netinet/in.h>**, **<pthread.h>**, **<sched.h>**,  
81054 **<semaphore.h>**, **<spawn.h>**, **<sys/socket.h>**, *pthread\_kill()*, and *pthread\_sigmask()*  
81055 in **<signal.h>**, **<trace.h>**, interfaces marked as optional in **<sys/mman.h>**,  
81056 interfaces marked as ADV (Advisory Information) in **<fcntl.h>**, and interfaces  
81057 beginning with the prefix *clock\_* or *time\_* in **<time.h>**. This option shall not be  
81058 required to be present to cause a search of this library.

81059 **-l l** This option shall make available all interfaces required by the C-language output  
81060 of *lex* that are not made available through the **-l c** option.

81061 **-l pthread** This option shall make available all interfaces referenced in **<pthread.h>** and  
81062 *pthread\_kill()* and *pthread\_sigmask()* referenced in **<signal.h>**. An implementation  
81063 may search this library in the absence of this option.

81064 **-l m** This option shall make available all interfaces referenced in **<math.h>**,  
81065 **<complex.h>**, and **<fenv.h>**. An implementation may search this library in the  
81066 absence of this option.

81067 **-l rt** This option shall make available all interfaces referenced in **<aio.h>**, **<mqueue.h>**,  
81068 **<sched.h>**, **<semaphore.h>**, and **<spawn.h>**, interfaces marked as optional in  
81069 **<sys/mman.h>**, interfaces marked as ADV (Advisory Information) in **<fcntl.h>**,  
81070 and interfaces beginning with the prefix *clock\_* and *time\_* in **<time.h>**. An



81071 implementation may search this library in the absence of this option.

81072 OB **-l trace** This option shall make available all interfaces referenced in **<trace.h>**. An  
 81073 implementation may search this library in the absence of this option.

81074 **-l xnet** This option shall make available all interfaces referenced in **<arpa/inet.h>**,  
 81075 **<netdb.h>**, **<net/if.h>**, **<netinet/in.h>**, and **<sys/socket.h>**. An implementation  
 81076 may search this library in the absence of this option.

81077 **-l y** This option shall make available all interfaces required by the C-language output  
 81078 of *yacc* that are not made available through the **-l c** option.

81079 In the absence of options that inhibit invocation of the link editor, such as **-c** or **-E**, the *c99* utility  
 81080 shall cause the equivalent of a **-l c** option to be passed to the link editor after the last *pathname*  
 81081 operand or **-l** option, causing it to be searched after all other object files and libraries are loaded.

81082 OB It is unspecified whether the libraries **libc.a**, **libl.a**, **libm.a**, **libpthread.a**, **librt.a**, **libtrace.a**,  
 81083 **libxnet.a**, or **liby.a** exist as regular files. The implementation may accept as **-l** option-arguments  
 81084 names of objects that do not exist as regular files.

## 81085 External Symbols

81086 The C compiler and link editor shall support the significance of external symbols up to a length  
 81087 of at least 31 bytes; the action taken upon encountering symbols exceeding the implementation-  
 81088 defined maximum symbol length is unspecified.

81089 The compiler and link editor shall support a minimum of 511 external symbols per source or  
 81090 object file, and a minimum of 4095 external symbols in total. A diagnostic message shall be  
 81091 written to the standard output if the implementation-defined limit is exceeded; other actions are  
 81092 unspecified.

## 81093 Header Search

81094 If a file with the same name as one of the standard headers defined in XBD Chapter 13 (on page +  
 81095 219), not provided as part of the implementation, is placed in any of the usual places that are +  
 81096 searched by default for headers, the results are unspecified.

## 81097 Programming Environments

81098 All implementations shall support one of the following programming environments as a default.  
 81099 Implementations may support more than one of the following programming environments.  
 81100 Applications can use *sysconf()* or *getconf* to determine which programming environments are  
 81101 supported.

Table 4-4 Programming Environments: Type Sizes

| Programming Environment<br><i>getconf</i> Name | Bits in<br>int | Bits in<br>long | Bits in<br>pointer | Bits in<br>off_t |
|------------------------------------------------|----------------|-----------------|--------------------|------------------|
| _POSIX_V7_ILP32_OFF32                          | 32             | 32              | 32                 | 32               |
| _POSIX_V7_ILP32_OFFBIG                         | 32             | 32              | 32                 | ≥64              |
| _POSIX_V7_LP64_OFF64                           | 32             | 64              | 64                 | 64               |
| _POSIX_V7_LP64_OFFBIG                          | ≥32            | ≥64             | ≥64                | ≥64              |

All implementations shall support one or more environments where the widths of the following types are no greater than the width of type **long**:

|                  |                    |                 |
|------------------|--------------------|-----------------|
| <b>blksize_t</b> | <b>ptrdiff_t</b>   | <b>tcflag_t</b> |
| <b>cc_t</b>      | <b>size_t</b>      | <b>wchar_t</b>  |
| <b>mode_t</b>    | <b>speed_t</b>     | <b>wint_t</b>   |
| <b>nfds_t</b>    | <b>ssize_t</b>     |                 |
| <b>pid_t</b>     | <b>suseconds_t</b> |                 |

The executable files created when these environments are selected shall be in a proper format for execution by the *exec* family of functions. Each environment may be one of the ones in Table 4-4, or it may be another environment. The names for the environments that meet this requirement shall be output by a *getconf* command using the `_POSIX_V7_WIDTH_RESTRICTED_ENVS` argument, as a <newline>-separated list of names suitable for use with the *getconf* `-v` option. If more than one environment meets the requirement, the names of all such environments shall be output on separate lines. Any of these names can then be used in a subsequent *getconf* command to obtain the flags specific to that environment with the following suffixes added as appropriate:

`_CFLAGS` To get the C compiler flags.

`_LDFLAGS` To get the linker/loader flags.

`_LIBS` To get the libraries.

This requirement may be removed in a future version.

When this utility processes a file containing a function called *main()*, it shall be defined with a return type equivalent to **int**. Using return from the initial call to *main()* shall be equivalent (other than with respect to language scope issues) to calling *exit()* with the returned value. Reaching the end of the initial call to *main()* shall be equivalent to calling *exit(0)*. The implementation shall not declare a prototype for this function.

Implementations provide configuration strings for C compiler flags, linker/loader flags, and libraries for each supported environment. When an application needs to use a specific programming environment rather than the implementation default programming environment while compiling, the application shall first verify that the implementation supports the desired environment. If the desired programming environment is supported, the application shall then invoke *c99* with the appropriate C compiler flags as the first options for the compile, the appropriate linker/loader flags after any other options except `-l` but before any operands or `-l` options, and the appropriate libraries at the end of the operands and `-l` options.

Conforming applications shall not attempt to link together object files compiled for different programming models. Applications shall also be aware that binary data placed in shared memory or in files might not be recognized by applications built for other programming models.

**Table 4-5** Programming Environments: *c99* Arguments

| Programming Environment<br><i>getconf</i> Name | Use                                                  | <i>c99</i> Arguments<br><i>getconf</i> Name                                                 |
|------------------------------------------------|------------------------------------------------------|---------------------------------------------------------------------------------------------|
| _POSIX_V7_ILP32_OFF32                          | C Compiler Flags<br>Linker/Loader Flags<br>Libraries | POSIX_V7_ILP32_OFF32_CFLAGS<br>POSIX_V7_ILP32_OFF32_LDFLAGS<br>POSIX_V7_ILP32_OFF32_LIBS    |
| _POSIX_V7_ILP32_OFFBIG                         | C Compiler Flags<br>Linker/Loader Flags<br>Libraries | POSIX_V7_ILP32_OFFBIG_CFLAGS<br>POSIX_V7_ILP32_OFFBIG_LDFLAGS<br>POSIX_V7_ILP32_OFFBIG_LIBS |
| _POSIX_V7_LP64_OFF64                           | C Compiler Flags<br>Linker/Loader Flags<br>Libraries | POSIX_V7_LP64_OFF64_CFLAGS<br>POSIX_V7_LP64_OFF64_LDFLAGS<br>POSIX_V7_LP64_OFF64_LIBS       |
| _POSIX_V7_LPBIG_OFFBIG                         | C Compiler Flags<br>Linker/Loader Flags<br>Libraries | POSIX_V7_LPBIG_OFFBIG_CFLAGS<br>POSIX_V7_LPBIG_OFFBIG_LDFLAGS<br>POSIX_V7_LPBIG_OFFBIG_LIBS |

In addition to the type size programming environments above, all implementations also support a multi-threaded programming environment that is orthogonal to all of the programming environments listed above. The *getconf* utility can be used to get flags for the threaded programming environment, as indicated in [Table 4-6](#).

**Table 4-6** Threaded Programming Environment: *c99* Arguments

| Programming Environment<br><i>getconf</i> Name | Use                                     | <i>c99</i> Arguments<br><i>getconf</i> Name         |
|------------------------------------------------|-----------------------------------------|-----------------------------------------------------|
| _POSIX_THREADS                                 | C Compiler Flags<br>Linker/Loader Flags | POSIX_V7_THREADS_CFLAGS<br>POSIX_V7_THREADS_LDFLAGS |

These programming environment flags may be used in conjunction with any of the type size programming environments supported by the implementation.

**EXIT STATUS**

The following exit values shall be returned:

0 Successful compilation or link edit.

>0 An error occurred.

**CONSEQUENCES OF ERRORS**

When *c99* encounters a compilation error that causes an object file not to be created, it shall write a diagnostic to standard error and continue to compile other source code operands, but it shall not perform the link phase and return a non-zero exit status. If the link edit is unsuccessful, a diagnostic message shall be written to standard error and *c99* exits with a non-zero status. A conforming application shall rely on the exit status of *c99*, rather than on the existence or mode of the executable file.

## APPLICATION USAGE

Since the *c99* utility usually creates files in the current directory during the compilation process, it is typically necessary to run the *c99* utility in a directory in which a file can be created.

On systems providing POSIX Conformance (see XBD [Chapter 2](#), on page 15), *c99* is required only with the C-Language Development option; XSI-conformant systems always provide *c99*.

Some historical implementations have created *.o* files when *-c* is not specified and more than one source file is given. Since this area is left unspecified, the application cannot rely on *.o* files being created, but it also must be prepared for any related *.o* files that already exist being deleted at the completion of the link edit.

There is the possible implication that if a user supplies versions of the standard functions (before they would be encountered by an implicit *-l c* or explicit *-l m*), that those versions would be used in place of the standard versions. There are various reasons this might not be true (functions defined as macros, manipulations for clean name space, and so on), so the existence of files named in the same manner as the standard libraries within the *-L* directories is explicitly stated to produce unspecified behavior.

All of the functions specified in the System Interfaces volume of POSIX.1-2008 may be made visible by implementations when the Standard C Library is searched. Conforming applications must explicitly request searching the other standard libraries when functions made visible by those libraries are used.

In the ISO C standard the mapping from physical source characters to the C source character set is implementation-defined. Implementations may strip white-space characters before the terminating *<newline>* of a (physical) line as part of this mapping and, as a consequence of this, one or more white-space characters (and no other characters) between a *<backslash>* character and the *<newline>* character that terminates the line produces implementation-defined results. Portable applications should not use such constructs.

Some *c99* compilers not conforming to POSIX.1-2008 do not support trigraphs by default.

## EXAMPLES

1. The following usage example compiles **foo.c** and creates the executable file **foo**:

```
c99 -o foo foo.c
```

The following usage example compiles **foo.c** and creates the object file **foo.o**:

```
c99 -c foo.c
```

The following usage example compiles **foo.c** and creates the executable file **a.out**:

```
c99 foo.c
```

The following usage example compiles **foo.c**, links it with **bar.o**, and creates the executable file **a.out**. It may also create and leave **foo.o**:

```
c99 foo.c bar.o
```

2. The following example shows how an application using threads interfaces can test for support of and use a programming environment supporting 32-bit **int**, **long**, and **pointer** types and an **off\_t** type using at least 64 bits:

```
offbig_env=$(getconf _POSIX_V7_ILP32_OFFBIG)
if [$offbig_env != "-1"] && [$offbig_env != "undefined"]
then
 c99 $(getconf POSIX_V7_ILP32_OFFBIG_CFLAGS) \
```

```

81224 $(getconf POSIX_V7_THREADS_CFLAGS) -D_XOPEN_SOURCE=700 \
81225 $(getconf POSIX_V7_ILP32_OFFBIG_LDFLAGS) \
81226 $(getconf POSIX_V7_THREADS_LDFLAGS) foo.c -o foo \
81227 $(getconf POSIX_V7_ILP32_OFFBIG_LIBS) \
81228 -l pthread
81229 else
81230 echo ILP32_OFFBIG programming environment not supported
81231 exit 1
81232 fi

```

3. The following examples clarify the use and interactions of **-L** and **-l** options.

Consider the case in which module **a.c** calls function *f()* in library **libQ.a**, and module **b.c** calls function *g()* in library **libp.a**. Assume that both libraries reside in **/a/b/c**. The command line to compile and link in the desired way is:

```
c99 -L /a/b/c main.o a.c -l Q b.c -l p
```

In this case the **-L** option need only precede the first **-l** option, since both **libQ.a** and **libp.a** reside in the same directory.

Multiple **-L** options can be used when library name collisions occur. Building on the previous example, suppose that the user wants to use a new **libp.a**, in **/a/a/a**, but still wants *f()* from **/a/b/c/libQ.a**:

```
c99 -L /a/a/a -L /a/b/c main.o a.c -l Q b.c -l p
```

In this example, the linker searches the **-L** options in the order specified, and finds **/a/a/a/libp.a** before **/a/b/c/libp.a** when resolving references for **b.c**. The order of the **-l** options is still important, however.

4. The following example shows how an application can use a programming environment where the widths of the following types:

**blksize\_t**, **cc\_t**, **mode\_t**, **nfds\_t**, **pid\_t**, **ptrdiff\_t**, **size\_t**, **speed\_t**, **ssize\_t**, **suseconds\_t**, **tcflag\_t**, **wchar\_t**, **wint\_t**

are no greater than the width of type **long**:

```

81251 # First choose one of the listed environments ...
81252
81253 # ... if there are no additional constraints, the first one will do:
81254 CENV=$(getconf POSIX_V7_WIDTH_RESTRICTED_ENVS | head -n 1)
81255
81256 # ... or, if an environment that supports large files is preferred,
81257 # look for names that contain "OFF64" or "OFFBIG". (This chooses
81258 # the last one in the list if none match.)
81259 for CENV in $(getconf POSIX_V7_WIDTH_RESTRICTED_ENVS)
81260 do
81261 case $CENV in
81262 *OFF64*|*OFFBIG*) break ;;
81263 esac
81264 done
81265
81266 # The chosen environment name can now be used like this:
81267
81268 c99 $(getconf ${CENV}_CFLAGS) -D _POSIX_C_SOURCE=200809L \
81269 $(getconf ${CENV}_LDFLAGS) foo.c -o foo \
81270 $(getconf ${CENV}_LIBS)

```

**RATIONALE**

The *c99* utility is based on the *c89* utility originally introduced in the ISO POSIX-2:1993 standard.

Some of the changes from *c89* include the ability to intersperse options and operands (which many *c89* implementations allowed despite it not being specified), the description of **-l** as an option instead of an operand, and the modification to the contents of the Standard Libraries section to account for new headers and options; for example, **<spawn.h>** added to the description of **-l rt**, and **-l trace** added for the Tracing option.

POSIX.1-2008 specifies that the *c99* utility must be able to use regular files for **\*.o** files and for **a.out** files. Implementations are free to overwrite existing files of other types when attempting to create object files and executable files, but are not required to do so. If something other than a regular file is specified and using it fails for any reason, *c99* is required to issue a diagnostic message and exit with a non-zero exit status. But for some file types, the problem may not be noticed for a long time. For example, if a FIFO named **a.out** exists in the current directory, *c99* may attempt to open **a.out** and will hang in the *open()* call until another process opens the FIFO for reading. Then *c99* may write most of the **a.out** to the FIFO and fail when it tries to seek back close to the start of the file to insert a timestamp (FIFOs are not seekable files). The *c99* utility is also allowed to issue a diagnostic immediately if it encounters an **a.out** or **\*.o** file that is not a regular file. For portable use, applications should ensure that any **a.out**, **-o** option-argument, or **\*.o** files corresponding to any **\*.c** files do not conflict with names already in use that are not regular files or symbolic links that point to regular files.

On many systems, multi-threaded applications run in a programming environment that is distinct from that used by single-threaded applications. This multi-threaded programming environment (in addition to needing to specify **-l pthread** at link time) may require additional flags to be set when headers are processed at compile time (**-D\_REENTRANT** being common). This programming environment is orthogonal to the type size programming environments discussed above and listed in [Table 4-4](#) (on page 2516). This version of the standard adds *getconf* utility calls to provide the C compiler flags and linker/loader flags needed to support multi-threaded applications. Note that on a system where single-threaded applications are a special case of a multi-threaded application, both of these *getconf* calls may return NULL strings; on other implementations both of these strings may be non-NULL strings.

The C standardization committee invented trigraphs (e.g., "??!" to represent ' | ') to address character portability problems in development environments based on national variants of the 7-bit ISO/IEC 646:1991 standard character set. However, these environments were already obsolete by the time the first ISO C standard was published, and in practice trigraphs have not been used for their intended purpose, and usually are intended to have their original meaning in K&R C. For example, in practice a C-language source string like "What??!" is usually intended to end in two <question-mark> characters and an <exclamation-mark>, not in ' | '.

**FUTURE DIRECTIONS**

None.

**SEE ALSO**

[Section 1.1.1.4](#) (on page 2302), *ar*, *getconf*, *make*, *nm*, *strip*, *umask*

[XBD Chapter 8](#) (on page 173), [Section 12.2](#) (on page 215), [Chapter 13](#) (on page 219)

XSH *exec*, *sysconf()*

**CHANGE HISTORY**

First released in Issue 6. Included for alignment with the ISO/IEC 9899:1999 standard.

IEEE Std 1003.1-2001/Cor 1-2002, item XCU/TC1/D6/12 is applied, correcting the EXTENDED DESCRIPTION of **-l c** and **-l m**. Previously, the text did not take into account the presence of the *c99* math headers.

IEEE Std 1003.1-2001/Cor 1-2002, item XCU/TC1/D6/13 is applied, changing the reference to the **libxnet** library to **libxnet.a**.

IEEE Std 1003.1-2001/Cor 2-2004, item XCU/TC2/D6/5 is applied, updating the OPTIONS section, so that the names of files contained in the directory specified by the **-L** option are not assumed to end in the **.a** suffix. The set of library prefixes is also updated.

IEEE Std 1003.1-2001/Cor 2-2004, item XCU/TC2/D6/6 is applied, removing the lead underscore from the `POSIX_V6_WIDTH_RESTRICTED_ENVS` variable in the EXTENDED DESCRIPTION and the EXAMPLES sections.

**Issue 7**

Austin Group Interpretation 1003.1-2001 #020 (SD5-XCU-ERN-10) is applied, adding to the OUTPUT FILES section and also adding associated RATIONALE.

Austin Group Interpretation 1003.1-2001 #095 is applied, clarifying the **-l library** operand.

Austin Group Interpretation 1003.1-2001 #166 is applied.

Austin Group Interpretation 1003.1-2001 #190 is applied, clarifying the handling of trailing white-space characters.

Austin Group Interpretation 1003.1-2001 #191 is applied, adding APPLICATION USAGE and RATIONALE regarding C-language trigraphs.

SD5-XCU-ERN-6 is applied, clarifying that Guideline 9 of the Utility Syntax Guidelines does not apply (options can be interspersed with operands).

SD5-XCU-ERN-11 is applied, adding the **<net/if.h>** header to the descriptions of **-l c** and **-l xnet**.

SD5-XCU-ERN-65 is applied, updating the EXAMPLES section.

SD5-XCU-ERN-67 and SD5-XCU-ERN-97 are applied, updating the SYNOPSIS.

SD5-XCU-ERN-133 is applied, updating the EXTENDED DESCRIPTION.

The *getconf* variables for the supported programming environments are updated to be V7.

The **-l trace** operand is marked obsolescent.

The *c99* reference page is rewritten to describe **-l** as an option rather than an operand.

POSIX.1-2008, Technical Corrigendum 1, XCU/TC1-2008/0068 [129], XCU/TC1-2008/0069 + [187], XCU/TC1-2008/0070 [187], XCU/TC1-2008/0071 [131], XCU/TC1-2008/0072 [187], and + XCU/TC1-2008/0073 [364,430] are applied.

81347 **NAME**

81348 cal — print a calendar

81349 **SYNOPSIS**81350 XSI cal **[** [*month*] *year* **]**81351 **DESCRIPTION**

81352 The *cal* utility shall write a calendar to standard output using the Julian calendar for dates from  
 81353 January 1, 1 through September 2, 1752 and the Gregorian calendar for dates from September 14,  
 81354 1752 through December 31, 9999 as though the Gregorian calendar had been adopted on  
 81355 September 14, 1752.

81356 If no operands are given, *cal* shall produce a one-month calendar for the current month in the +  
 81357 current year. If only the *year* operand is given, *cal* shall produce a calendar for all twelve months +  
 81358 in the given calendar year. If both *month* and *year* operands are given, *cal* shall produce a one- +  
 81359 month calendar for the given month in the given year.

81360 **OPTIONS**

81361 None.

81362 **OPERANDS**

81363 The following operands shall be supported:

81364 *month* Specify the month to be displayed, represented as a decimal integer from 1 |  
 81365 (January) to 12 (December).

81366 *year* Specify the year for which the calendar is displayed, represented as a decimal |  
 81367 integer from 1 to 9999.

81368 **STDIN**

81369 Not used.

81370 **INPUT FILES**

81371 None.

81372 **ENVIRONMENT VARIABLES**81373 The following environment variables shall affect the execution of *cal*:

81374 *LANG* Provide a default value for the internationalization variables that are unset or null.  
 81375 (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 81376 variables used to determine the values of locale categories.)

81377 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
 81378 internationalization variables.

81379 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
 81380 characters (for example, single-byte as opposed to multi-byte characters in  
 81381 arguments).

81382 *LC\_MESSAGES*

81383 Determine the locale that should be used to affect the format and contents of  
 81384 diagnostic messages written to standard error, and informative messages written  
 81385 to standard output.

81386 *LC\_TIME* Determine the format and contents of the calendar.

81387 *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.



81388            *TZ*            Determine the timezone used to calculate the value of the current month.

81389    **ASYNCHRONOUS EVENTS**

81390            Default.

81391    **STDOUT**

81392            The standard output shall be used to display the calendar, in an unspecified format.

81393    **STDERR**

81394            The standard error shall be used only for diagnostic messages.

81395    **OUTPUT FILES**

81396            None.

81397    **EXTENDED DESCRIPTION**

81398            None.

81399    **EXIT STATUS**

81400            The following exit values shall be returned:

81401            0    Successful completion.

81402            >0   An error occurred.

81403    **CONSEQUENCES OF ERRORS**

81404            Default.

81405    **APPLICATION USAGE**

81406            Note that:

81407            `cal 83`

81408            refers to A.D. 83, not 1983.

81409    **EXAMPLES**

81410            None.

81411    **RATIONALE**

81412            Earlier versions of this standard incorrectly required that the command:

81413            `cal 2000`

81414            write a one-month calendar for the current calendar month (no matter what the current year is)

81415            in the year 2000 to standard output. This did not match historic practice in any known version of

81416            the *cal* utility. The description has been updated to match historic practice. When only the *year*

81417            operand is given, *cal* writes a twelve-month calendar for the specified year.

81418    **FUTURE DIRECTIONS**

81419            A future version of this standard may support locale-specific recognition of the date of adoption

81420            of the Gregorian calendar.

81421    **SEE ALSO**

81422            XBD [Chapter 8](#) (on page 173)

81423    **CHANGE HISTORY**

81424            First released in Issue 2.

81425    **Issue 6**

81426            The DESCRIPTION is updated to allow for traditional behavior for years before the adoption of

81427            the Gregorian calendar.

81428 **Issue 7**

81429 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

81430 POSIX.1-2008, Technical Corrigendum 1, XCU/TC1-2008/0074 [56] and XCU/TC1-2008/0075 +  
81431 [56] are applied.

81432 **NAME**

81433       cat — concatenate and print files

81434 **SYNOPSIS**81435       cat [-u] [*file*...]81436 **DESCRIPTION**81437       The *cat* utility shall read files in sequence and shall write their contents to the standard output in  
81438       the same sequence.81439 **OPTIONS**81440       The *cat* utility shall conform to XBD [Section 12.2](#) (on page 215).

81441       The following option shall be supported:

81442       -u           Write bytes from the input file to the standard output without delay as each is  
81443       read.81444 **OPERANDS**

81445       The following operand shall be supported:

81446       *file*        A pathname of an input file. If no *file* operands are specified, the standard input  
81447       shall be used. If a *file* is '-', the *cat* utility shall read from the standard input at  
81448       that point in the sequence. The *cat* utility shall not close and reopen standard input  
81449       when it is referenced in this way, but shall accept multiple occurrences of '-' as a  
81450       *file* operand.81451 **STDIN**81452       The standard input shall be used only if no *file* operands are specified, or if a *file* operand is '-'.  
81453       See the INPUT FILES section.81454 **INPUT FILES**

81455       The input files can be any file type.

81456 **ENVIRONMENT VARIABLES**81457       The following environment variables shall affect the execution of *cat*:81458       LANG        Provide a default value for the internationalization variables that are unset or null.  
81459       (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
81460       variables used to determine the values of locale categories.)81461       LC\_ALL      If set to a non-empty string value, override the values of all the other  
81462       internationalization variables.81463       LC\_CTYPE    Determine the locale for the interpretation of sequences of bytes of text data as  
81464       characters (for example, single-byte as opposed to multi-byte characters in  
81465       arguments).

81466       LC\_MESSAGES

81467       Determine the locale that should be used to affect the format and contents of  
81468       diagnostic messages written to standard error.

81469 XSI       NLSPATH   Determine the location of message catalogs for the processing of LC\_MESSAGES.

81470 **ASYNCHRONOUS EVENTS**

81471       Default.

**81472 STDOUT**

81473 The standard output shall contain the sequence of bytes read from the input files. Nothing else  
81474 shall be written to the standard output.

**81475 STDERR**

81476 The standard error shall be used only for diagnostic messages.

**81477 OUTPUT FILES**

81478 None.

**81479 EXTENDED DESCRIPTION**

81480 None.

**81481 EXIT STATUS**

81482 The following exit values shall be returned:

81483 0 All input files were output successfully.

81484 >0 An error occurred.

**81485 CONSEQUENCES OF ERRORS**

81486 Default.

**81487 APPLICATION USAGE**

81488 The **-u** option has value in prototyping non-blocking reads from FIFOs. The intent is to support  
81489 the following sequence:

```
81490 mkfifo foo
81491 cat -u foo > /dev/tty13 &
81492 cat -u > foo
```

81493 It is unspecified whether standard output is or is not buffered in the default case. This is  
81494 sometimes of interest when standard output is associated with a terminal, since buffering may  
81495 delay the output. The presence of the **-u** option guarantees that unbuffered I/O is available. It is  
81496 implementation-defined whether the *cat* utility buffers output if the **-u** option is not specified.  
81497 Traditionally, the **-u** option is implemented using the equivalent of the *setvbuf()* function  
81498 defined in the System Interfaces volume of POSIX.1-2008.

**81499 EXAMPLES**

81500 The following command:

```
81501 cat myfile
```

81502 writes the contents of the file **myfile** to standard output.

81503 The following command:

```
81504 cat doc1 doc2 > doc.all
```

81505 concatenates the files **doc1** and **doc2** and writes the result to **doc.all**.

81506 Because of the shell language mechanism used to perform output redirection, a command such  
81507 as this:

```
81508 cat doc doc.end > doc
```

81509 causes the original data in **doc** to be lost.

81510 The command:

```
81511 cat start - middle - end > file
```

81512 when standard input is a terminal, gets two arbitrary pieces of input from the terminal with a

single invocation of *cat*. Note, however, that if standard input is a regular file, this would be equivalent to the command:

```
cat start - middle /dev/null end > file
```

because the entire contents of the file would be consumed by *cat* the first time '-' was used as a *file* operand and an end-of-file condition would be detected immediately when '-' was referenced the second time.

## RATIONALE

Historical versions of the *cat* utility include the *-e*, *-t*, and *-v*, options which permit the ends of lines, <tab> characters, and invisible characters, respectively, to be rendered visible in the output. The standard developers omitted these options because they provide too fine a degree of control over what is made visible, and similar output can be obtained using a command such as:

```
sed -n l pathname
```

The latter also has the advantage that its output is unambiguous, whereas the output of historical *cat -etv* is not.

The *-s* option was omitted because it corresponds to different functions in BSD and System V-based systems. The BSD *-s* option to squeeze blank lines can be accomplished by the shell script shown in the following example:

```
sed -n '
Write non-empty lines.
./ {
 p
 d
}
Write a single empty line, then look for more empty lines.
/^$/ p
Get next line, discard the held <newline> (empty line),
and look for more empty lines.
:Empty
/^$/ {
 N
 s/./ /
 b Empty
}
Write the non-empty line before going back to search
for the first in a set of empty lines.
p
'
```

The System V *-s* option to silence error messages can be accomplished by redirecting the standard error. Note that the BSD documentation for *cat* uses the term "blank line" to mean the same as the POSIX "empty line": a line consisting only of a <newline>.

The BSD *-n* option was omitted because similar functionality can be obtained from the *-n* option of the *pr* utility.

## FUTURE DIRECTIONS

None.

81557 **SEE ALSO**81558 [more](#)81559 XBD [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 215)81560 XSH [setvbuf\(\)](#)81561 **CHANGE HISTORY**

81562 First released in Issue 2.

81563 **Issue 7**

81564 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

81565 SD5-XCU-ERN-174 is applied, changing the RATIONALE.

**NAME**

cd — change the working directory

**SYNOPSIS**

cd [-L|-P] [*directory*]

cd -

**DESCRIPTION**

The *cd* utility shall change the working directory of the current shell execution environment (see [Section 2.12](#), on page 2353) by executing the following steps in sequence. (In the following steps, the symbol **curpath** represents an intermediate value used to simplify the description of the algorithm used by *cd*. There is no requirement that **curpath** be made visible to the application.)

1. If no *directory* operand is given and the *HOME* environment variable is empty or undefined, the default behavior is implementation-defined and no further steps shall be taken.
2. If no *directory* operand is given and the *HOME* environment variable is set to a non-empty value, the *cd* utility shall behave as if the directory named in the *HOME* environment variable was specified as the *directory* operand.
3. If the *directory* operand begins with a <slash> character, set **curpath** to the operand and proceed to step 7.
4. If the first component of the *directory* operand is dot or dot-dot, proceed to step 6.
5. Starting with the first pathname in the <colon>-separated pathnames of *CDPATH* (see the ENVIRONMENT VARIABLES section) if the pathname is non-null, test if the concatenation of that pathname, a <slash> character if that pathname did not end with a <slash> character, and the *directory* operand names a directory. If the pathname is null, test if the concatenation of dot, a <slash> character, and the operand names a directory. In either case, if the resulting string names an existing directory, set **curpath** to that string and proceed to step 7. Otherwise, repeat this step with the next pathname in *CDPATH* until all pathnames have been tested.
6. Set **curpath** to the *directory* operand.
7. If the *-P* option is in effect, proceed to step 10. If **curpath** does not begin with a <slash> character, set **curpath** to the string formed by the concatenation of the value of *PWD*, a <slash> character if the value of *PWD* did not end with a <slash> character, and **curpath**.
8. The **curpath** value shall then be converted to canonical form as follows, considering each component from beginning to end, in sequence:
  - a. Dot components and any <slash> characters that separate them from the next component shall be deleted.
  - b. For each dot-dot component, if there is a preceding component and it is neither root nor dot-dot, then:
    - i. If the preceding component does not refer (in the context of pathname resolution with symbolic links followed) to a directory, then the *cd* utility shall display an appropriate error message and no further steps shall be taken.
    - ii. The preceding component, all <slash> characters separating the preceding component from dot-dot, dot-dot, and all <slash> characters separating dot-dot from the following component (if any) shall be deleted.

- 81610 c. An implementation may further simplify **curpath** by removing any trailing  
 81611 <slash> characters that are not also leading <slash> characters, replacing multiple  
 81612 non-leading consecutive <slash> characters with a single <slash>, and replacing  
 81613 three or more leading <slash> characters with a single <slash>. If, as a result of  
 81614 this canonicalization, the **curpath** variable is null, no further steps shall be taken.
- 81615 9. If **curpath** is longer than {PATH\_MAX} bytes (including the terminating null) and the  
 81616 *directory* operand was not longer than {PATH\_MAX} bytes (including the terminating  
 81617 null), then **curpath** shall be converted from an absolute pathname to an equivalent  
 81618 relative pathname if possible. This conversion shall always be considered possible if the  
 81619 value of *PWD*, with a trailing <slash> added if it does not already have one, is an initial  
 81620 substring of **curpath**. Whether or not it is considered possible under other circumstances  
 81621 is unspecified. Implementations may also apply this conversion if **curpath** is not longer  
 81622 than {PATH\_MAX} bytes or the *directory* operand was longer than {PATH\_MAX} bytes.
- 81623 10. The *cd* utility shall then perform actions equivalent to the *chdir()* function called with  
 81624 **curpath** as the *path* argument. If these actions fail for any reason, the *cd* utility shall  
 81625 display an appropriate error message and the remainder of this step shall not be  
 81626 executed. If the **-P** option is not in effect, the *PWD* environment variable shall be set to  
 81627 the value that **curpath** had on entry to step 9 (i.e., before conversion to a relative  
 81628 pathname). If the **-P** option is in effect, the *PWD* environment variable shall be set to the  
 81629 string that would be output by *pwd -P*. If there is insufficient permission on the new  
 81630 directory, or on any parent of that directory, to determine the current working directory,  
 81631 the value of the *PWD* environment variable is unspecified.

81632 If, during the execution of the above steps, the *PWD* environment variable is set, the *OLDPWD* |  
 81633 environment variable shall also be set to the value of the old working directory (that is the  
 81634 current working directory immediately prior to the call to *cd*).

## 81635 OPTIONS

81636 The *cd* utility shall conform to XBD [Section 12.2](#) (on page 215).

81637 The following options shall be supported by the implementation:

- 81638 **-L** Handle the operand dot-dot logically; symbolic link components shall not be  
 81639 resolved before dot-dot components are processed (see steps 8. and 9. in the  
 81640 DESCRIPTION).
- 81641 **-P** Handle the operand dot-dot physically; symbolic link components shall be  
 81642 resolved before dot-dot components are processed (see step 7. in the  
 81643 DESCRIPTION).

81644 If both **-L** and **-P** options are specified, the last of these options shall be used and all others  
 81645 ignored. If neither **-L** nor **-P** is specified, the operand shall be handled dot-dot logically; see the  
 81646 DESCRIPTION.

## 81647 OPERANDS

81648 The following operands shall be supported:

- 81649 *directory* An absolute or relative pathname of the directory that shall become the new  
 81650 working directory. The interpretation of a relative pathname by *cd* depends on the  
 81651 **-L** option and the *CDPATH* and *PWD* environment variables. If *directory* is an  
 81652 empty string, the results are unspecified.
- 81653 **-** When a <hyphen> is used as the operand, this shall be equivalent to the command:  
 81654 `cd "$OLDPWD" && pwd`



81655 which changes to the previous working directory and then writes its name.

#### 81656 STDIN

81657 Not used.

#### 81658 INPUT FILES

81659 None.

#### 81660 ENVIRONMENT VARIABLES

81661 The following environment variables shall affect the execution of *cd*:

81662 *CDPATH* A <colon>-separated list of pathnames that refer to directories. The *cd* utility shall  
81663 use this list in its attempt to change the directory, as described in the  
81664 DESCRIPTION. An empty string in place of a directory pathname represents the  
81665 current directory. If *CDPATH* is not set, it shall be treated as if it were an empty  
81666 string.

81667 *HOME* The name of the directory, used when no *directory* operand is specified.

81668 *LANG* Provide a default value for the internationalization variables that are unset or null.  
81669 (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
81670 variables used to determine the values of locale categories.)

81671 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
81672 internationalization variables.

81673 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
81674 characters (for example, single-byte as opposed to multi-byte characters in  
81675 arguments).

81676 *LC\_MESSAGES*

81677 Determine the locale that should be used to affect the format and contents of  
81678 diagnostic messages written to standard error.

81679 XSI *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

81680 *OLDPWD* A pathname of the previous working directory, used by *cd -*.

81681 *PWD* This variable shall be set as specified in the DESCRIPTION. If an application sets  
81682 or unsets the value of *PWD*, the behavior of *cd* is unspecified.

#### 81683 ASYNCHRONOUS EVENTS

81684 Default.

#### 81685 STDOUT

81686 If a non-empty directory name from *CDPATH* is used, or if *cd -* is used, an absolute pathname of  
81687 the new working directory shall be written to the standard output as follows:

81688 "%s\n", <new directory>

81689 Otherwise, there shall be no output.

#### 81690 STDERR

81691 The standard error shall be used only for diagnostic messages.

#### 81692 OUTPUT FILES

81693 None.

81694 **EXTENDED DESCRIPTION**

81695 None.

81696 **EXIT STATUS**

81697 The following exit values shall be returned:

81698 0 The directory was successfully changed.

81699 &gt;0 An error occurred.

81700 **CONSEQUENCES OF ERRORS**

81701 The working directory shall remain unchanged.

81702 **APPLICATION USAGE**81703 Since *cd* affects the current shell execution environment, it is always provided as a shell regular  
81704 built-in. If it is called in a subshell or separate utility execution environment, such as one of the  
81705 following:81706 (cd /tmp)  
81707 nohup cd  
81708 find . -exec cd {} \;

81709 it does not affect the working directory of the caller's environment.

81710 The user must have execute (search) permission in *directory* in order to change to it.81711 **EXAMPLES**81712 The following template can be used to perform processing in the directory specified by *location* |  
81713 and end up in the current working directory in use before the first *cd* command was issued: |81714 cd *location* |  
81715 if [ \$? -ne 0 ] |  
81716 then |  
81717 print error message |  
81718 exit 1 |  
81719 fi |  
81720 ... do whatever is desired as long as the OLDPWD environment variable |  
81721 is not modified |  
81722 cd - |81723 **RATIONALE**81724 The use of the *CDPATH* was introduced in the System V shell. Its use is analogous to the use of  
81725 the *PATH* variable in the shell. The BSD C shell used a shell parameter *cdpath* for this purpose.81726 A common extension when *HOME* is undefined is to get the login directory from the user  
81727 database for the invoking user. This does not occur on System V implementations.81728 Some historical shells, such as the KornShell, took special actions when the directory name  
81729 contained a dot-dot component, selecting the logical parent of the directory, rather than the  
81730 actual parent directory; that is, it moved up one level toward the '/' in the pathname,  
81731 remembering what the user typed, rather than performing the equivalent of:

81732 chdir("../");

81733 In such a shell, the following commands would not necessarily produce equivalent output for all  
81734 directories:

81735 cd .. &amp;&amp; ls ls ..

81736 This behavior is now the default. It is not consistent with the definition of dot-dot in most

81737 historical practice; that is, while this behavior has been optionally available in the KornShell,  
 81738 other shells have historically not supported this functionality. The logical pathname is stored in  
 81739 the *PWD* environment variable when the *cd* utility completes and this value is used to construct  
 81740 the next directory name if *cd* is invoked with the *-L* option.

#### 81741 FUTURE DIRECTIONS

81742 None.

#### 81743 SEE ALSO

81744 [Section 2.12](#) (on page 2353), *pwd*

81745 XBD [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 215)

81746 XSH *chdir()*

#### 81747 CHANGE HISTORY

81748 First released in Issue 2.

#### 81749 Issue 6

81750 The following new requirements on POSIX implementations derive from alignment with the  
 81751 Single UNIX Specification:

- 81752 • The *cd* *-* operand, *PWD*, and *OLDPWD* are added.

81753 The *-L* and *-P* options are added to align with the IEEE P1003.2b draft standard. This also  
 81754 includes the introduction of a new description to include the effect of these options.

81755 IEEE Std 1003.1-2001/Cor 1-2002, item XCU/TC1/D6/14 is applied, changing the SYNOPSIS to  
 81756 make it clear that the *-L* and *-P* options are mutually-exclusive.

#### 81757 Issue 7

81758 Austin Group Interpretation 1003.1-2001 #037 is applied.

81759 Austin Group Interpretation 1003.1-2001 #199 is applied, clarifying how the *cd* utility handles  
 81760 concatenation of two pathnames when the first pathname ends in a <slash> character.

81761 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

81762 Step 7 of the processing performed by *cd* is revised to refer to **curpath** instead of “the operand”.

81763 Changes to the *pwd* utility and *PWD* environment variable have been made to match the  
 81764 changes to the *getcwd()* function made for Austin Group Interpretation 1003.1-2001 #140.

81765 POSIX.1-2008, Technical Corrigendum 1, XCU/TC1-2008/0076 [230], XCU/TC1-2008/0077 +  
 81766 [240], XCU/TC1-2008/0078 [240], and XCU/TC1-2008/0079 [123] are applied.

81767 **NAME**81768 cflow — generate a C-language flowgraph (**DEVELOPMENT**)81769 **SYNOPSIS**

```
81770 XSI cflow [-r] [-d num] [-D name[=def]]... [-i incl] [-I dir]...
81771 [-U dir]... file...
```

81772 **DESCRIPTION**

81773 The *cflow* utility shall analyze a collection of object files or assembler, C-language, *lex*, or *yacc*  
 81774 source files, and attempt to build a graph, written to standard output, charting the external  
 81775 references.

81776 **OPTIONS**

81777 The *cflow* utility shall conform to XBD [Section 12.2](#) (on page 215), except that the order of the **-D**,  
 81778 **-I**, and **-U** options (which are identical to their interpretation by *c99*) is significant.

81779 The following options shall be supported:

- 81780 **-d num** Indicate the depth at which the flowgraph is cut off. The application shall ensure  
 81781 that the argument *num* is a decimal integer. By default this is a very large number  
 81782 (typically greater than 32 000). Attempts to set the cut-off depth to a non-positive  
 81783 integer shall be ignored.
- 81784 **-i incl** Increase the number of included symbols. The *incl* option-argument is one of the  
 81785 following characters:
- 81786 *x* Include external and static data symbols. The default shall be to include only  
 81787 functions in the flowgraph.
- 81788 *\_* (Underscore) Include names that begin with an <underscore>. The default  
 81789 shall be to exclude these functions (and data if **-i x** is used).
- 81790 **-r** Reverse the caller: callee relationship, producing an inverted listing showing the  
 81791 callers of each function. The listing shall also be sorted in lexicographical order by  
 81792 callee.

81793 **OPERANDS**

81794 The following operand is supported:

- 81795 *file* The pathname of a file for which a graph is to be generated. Filenames suffixed by  
 81796 **.l** shall be taken to be *lex* input, **.y** as *yacc* input, **.c** as *c99* input, and **.i** as the  
 81797 output of *c99* **-E**. Such files shall be processed as appropriate, determined by their  
 81798 suffix.
- 81799 Files suffixed by **.s** (conventionally assembler source) may have more limited  
 81800 information extracted from them.

81801 **STDIN**

81802 Not used.

81803 **INPUT FILES**

81804 The input files shall be object files or assembler, C-language, *lex*, or *yacc* source files.

81805 **ENVIRONMENT VARIABLES**

81806 The following environment variables shall affect the execution of *cflow*:

- 81807 **LANG** Provide a default value for the internationalization variables that are unset or null.  
 81808 (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 81809 variables used to determine the values of locale categories.)

81810 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
 81811 internationalization variables.

81812 *LC\_COLLATE*  
 81813 Determine the locale for the ordering of the output when the *-r* option is used.

81814 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
 81815 characters (for example, single-byte as opposed to multi-byte characters in  
 81816 arguments and input files).

81817 *LC\_MESSAGES*  
 81818 Determine the locale that should be used to affect the format and contents of  
 81819 diagnostic messages written to standard error.

81820 *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

81821 **ASYNCHRONOUS EVENTS**  
 81822 Default.

81823 **STDOUT**  
 81824 The flowgraph written to standard output shall be formatted as follows:  
 81825 "%d %s:%s\n", <reference number>, <global>, <definition>

81826 Each line of output begins with a reference (that is, line) number, followed by indentation of at  
 81827 least one column position per level. This is followed by the name of the global, a <colon>, and  
 81828 its definition. Normally globals are only functions not defined as an external or beginning with  
 81829 an <underscore>; see the **OPTIONS** section for the *-i* inclusion option. For information extracted  
 81830 from C-language source, the definition consists of an abstract type declaration (for example, **char**  
 81831 \*) and, delimited by angle brackets, the name of the source file and the line number where the  
 81832 definition was found. Definitions extracted from object files indicate the filename and location  
 81833 counter under which the symbol appeared (for example, *text*).

81834 Once a definition of a name has been written, subsequent references to that name contain only  
 81835 the reference number of the line where the definition can be found. For undefined references,  
 81836 only "<>" shall be written.

81837 **STDERR**  
 81838 The standard error shall be used only for diagnostic messages.

81839 **OUTPUT FILES**  
 81840 None.

81841 **EXTENDED DESCRIPTION**  
 81842 None.

81843 **EXIT STATUS**  
 81844 The following exit values shall be returned:  
 81845 0 Successful completion.  
 81846 >0 An error occurred.

81847 **CONSEQUENCES OF ERRORS**  
 81848 Default.

**APPLICATION USAGE**

Files produced by *lex* and *yacc* cause the reordering of line number declarations, and this can confuse *cflow*. To obtain proper results, the input of *yacc* or *lex* must be directed to *cflow*.

**EXAMPLES**

Given the following in **file.c**:

```
int i;
int f();
int g();
int h();
int
main()
{
 f();
 g();
 f();
}
int
f()
{
 i = h();
}
```

The command:

```
cflow -i x file.c
```

produces the output:

```
1 main: int(), <file.c 6>
2 f: int(), <file.c 13>
3 h: <>
4 i: int, <file.c 1>
5 g: <>
```

**RATIONALE**

None.

**FUTURE DIRECTIONS**

None.

**SEE ALSO**

*c99*, *lex*, *yacc*

XBD [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 215)

**CHANGE HISTORY**

First released in Issue 2.

**Issue 6**

The normative text is reworded to avoid use of the term “must” for application requirements.

**Issue 7**

SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

81891 **NAME**

81892       chgrp — change the file group ownership

81893 **SYNOPSIS**

81894       chgrp [-h] group file...

81895       chgrp -R [-H|-L|-P] group file...

81896 **DESCRIPTION**81897       The *chgrp* utility shall set the group ID of the file named by each *file* operand to the group ID  
81898       specified by the *group* operand.81899       For each *file* operand, or, if the **-R** option is used, each file encountered while walking the  
81900       directory trees specified by the *file* operands, the *chgrp* utility shall perform actions equivalent to  
81901       the *chown()* function defined in the System Interfaces volume of POSIX.1-2008, called with the  
81902       following arguments:

- 81903
- The *file* operand shall be used as the *path* argument.
  - 81904       • The user ID of the file shall be used as the *owner* argument.
  - 81905       • The specified group ID shall be used as the *group* argument.

81906       Unless *chgrp* is invoked by a process with appropriate privileges, the set-user-ID and set-group-  
81907       ID bits of a regular file shall be cleared upon successful completion; the set-user-ID and set-  
81908       group-ID bits of other file types may be cleared.81909 **OPTIONS**81910       The *chgrp* utility shall conform to XBD [Section 12.2](#) (on page 215).

81911       The following options shall be supported by the implementation:

- 81912
- h**
- For each
- file*
- operand that names a file of type symbolic link,
- chgrp*
- shall attempt to |
- 
- 81913       set the group ID of the symbolic link instead of the file referenced by the symbolic |
- 
- 81914       link.
- 
- 81915
- H**
- If the
- R**
- option is specified and a symbolic link referencing a file of type directory
- 
- 81916       is specified on the command line,
- chgrp*
- shall change the group of the directory
- 
- 81917       referenced by the symbolic link and all files in the file hierarchy below it.
- 
- 81918
- L**
- If the
- R**
- option is specified and a symbolic link referencing a file of type directory
- 
- 81919       is specified on the command line or encountered during the traversal of a file
- 
- 81920       hierarchy,
- chgrp*
- shall change the group of the directory referenced by the symbolic
- 
- 81921       link and all files in the file hierarchy below it.
- 
- 81922
- P**
- If the
- R**
- option is specified and a symbolic link is specified on the command line
- 
- 81923       or encountered during the traversal of a file hierarchy,
- chgrp*
- shall change the group |
- 
- 81924       ID of the symbolic link. The
- chgrp*
- utility shall not follow the symbolic link to any
- 
- 81925       other part of the file hierarchy.
- 
- 81926
- R**
- Recursively change file group IDs. For each
- file*
- operand that names a directory,
- 
- 81927
- chgrp*
- shall change the group of the directory and all files in the file hierarchy
- 
- 81928       below it. Unless a
- H**
- ,
- L**
- , or
- P**
- option is specified, it is unspecified which of these
- 
- 81929       options will be used as the default.

81930       Specifying more than one of the mutually-exclusive options **-H**, **-L**, and **-P** shall not be  
81931       considered an error. The last option specified shall determine the behavior of the utility.

81932 **OPERANDS**

81933 The following operands shall be supported:

81934 *group* A group name from the group database or a numeric group ID. Either specifies a  
 81935 group ID to be given to each file named by one of the *file* operands. If a numeric  
 81936 *group* operand exists in the group database as a group name, the group ID number  
 81937 associated with that group name is used as the group ID.

81938 *file* A pathname of a file whose group ID is to be modified.

81939 **STDIN**

81940 Not used.

81941 **INPUT FILES**

81942 None.

81943 **ENVIRONMENT VARIABLES**81944 The following environment variables shall affect the execution of *chgrp*:

81945 *LANG* Provide a default value for the internationalization variables that are unset or null.  
 81946 (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 81947 variables used to determine the values of locale categories.)

81948 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
 81949 internationalization variables.

81950 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
 81951 characters (for example, single-byte as opposed to multi-byte characters in  
 81952 arguments).

81953 *LC\_MESSAGES*

81954 Determine the locale that should be used to affect the format and contents of  
 81955 diagnostic messages written to standard error.

81956 XSI *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

81957 **ASYNCHRONOUS EVENTS**

81958 Default.

81959 **STDOUT**

81960 Not used.

81961 **STDERR**

81962 The standard error shall be used only for diagnostic messages.

81963 **OUTPUT FILES**

81964 None.

81965 **EXTENDED DESCRIPTION**

81966 None.

81967 **EXIT STATUS**

81968 The following exit values shall be returned:

81969 0 The utility executed successfully and all requested changes were made.

81970 &gt;0 An error occurred.



81971 **CONSEQUENCES OF ERRORS**

81972 Default.

81973 **APPLICATION USAGE**81974 Only the owner of a file or the user with appropriate privileges may change the owner or group  
81975 of a file.81976 Some implementations restrict the use of *chgrp* to a user with appropriate privileges when the  
81977 *group* specified is not the effective group ID or one of the supplementary group IDs of the calling  
81978 process.81979 **EXAMPLES**

81980 None.

81981 **RATIONALE**81982 The System V and BSD versions use different exit status codes. Some implementations used the  
81983 exit status as a count of the number of errors that occurred; this practice is unworkable since it  
81984 can overflow the range of valid exit status values. The standard developers chose to mask these  
81985 by specifying only 0 and >0 as exit values.81986 The functionality of *chgrp* is described substantially through references to *chown()*. In this way,  
81987 there is no duplication of effort required for describing the interactions of permissions, multiple  
81988 groups, and so on.81989 **FUTURE DIRECTIONS**

81990 None.

81991 **SEE ALSO**81992 *chmod*, *chown*81993 XBD [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 215)81994 XSH *chown()*81995 **CHANGE HISTORY**

81996 First released in Issue 2.

81997 **Issue 6**81998 New options **-H**, **-L**, and **-P** are added to align with the IEEE P1003.2b draft standard. These  
81999 options affect the processing of symbolic links.82000 IEEE PASC Interpretation 1003.2 #172 is applied, changing the CONSEQUENCES OF ERRORS  
82001 section to "Default."82002 IEEE Std 1003.1-2001/Cor 1-2002, item XCU/TC1/D6/15 is applied, changing the SYNOPSIS to  
82003 make it clear that **-h** and **-R** are optional.82004 **Issue 7**82005 SD5-XCU-ERN-8 is applied, removing the **-R** from the first line of the SYNOPSIS.

82006 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

82007 POSIX.1-2008, Technical Corrigendum 1, XCU/TC1-2008/0080 [237,341] is applied.

+

82008 **NAME**

82009        chmod — change the file modes

82010 **SYNOPSIS**

82011        chmod [-R] *mode file...*

82012 **DESCRIPTION**

82013        The *chmod* utility shall change any or all of the file mode bits of the file named by each *file*  
82014        operand in the way specified by the *mode* operand.

82015        It is implementation-defined whether and how the *chmod* utility affects any alternate or  
82016        additional file access control mechanism (see XBD [Section 4.4](#), on page 108) being used for the  
82017        specified file.

82018        Only a process whose effective user ID matches the user ID of the file, or a process with  
82019        appropriate privileges, shall be permitted to change the file mode bits of a file.

82020        Upon successfully changing the file mode bits of a file, the *chmod* utility shall mark for update  
82021        the last file status change timestamp of the file.

82022 **OPTIONS**

82023        The *chmod* utility shall conform to XBD [Section 12.2](#) (on page 215).

82024        The following option shall be supported:

82025        **-R**        Recursively change file mode bits. For each *file* operand that names a directory,  
82026        *chmod* shall change the file mode bits of the directory and all files in the file  
82027        hierarchy below it.

82028 **OPERANDS**

82029        The following operands shall be supported:

82030        *mode*        Represents the change to be made to the file mode bits of each file named by one of  
82031        the *file* operands; see the EXTENDED DESCRIPTION section.

82032        *file*        A pathname of a file whose file mode bits shall be modified.

82033 **STDIN**

82034        Not used.

82035 **INPUT FILES**

82036        None.

82037 **ENVIRONMENT VARIABLES**

82038        The following environment variables shall affect the execution of *chmod*:

82039        *LANG*        Provide a default value for the internationalization variables that are unset or null.  
82040        (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
82041        variables used to determine the values of locale categories.)

82042        *LC\_ALL*        If set to a non-empty string value, override the values of all the other  
82043        internationalization variables.

82044        *LC\_CTYPE*     Determine the locale for the interpretation of sequences of bytes of text data as  
82045        characters (for example, single-byte as opposed to multi-byte characters in  
82046        arguments).

82047        *LC\_MESSAGES*

82048        Determine the locale that should be used to affect the format and contents of  
82049        diagnostic messages written to standard error.

|       |     |                             |                                                                                                                                                        |
|-------|-----|-----------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------|
| 82050 | XSI | <b>NLSPATH</b>              | Determine the location of message catalogs for the processing of <i>LC_MESSAGES</i> .                                                                  |
| 82051 |     | <b>ASYNCHRONOUS EVENTS</b>  |                                                                                                                                                        |
| 82052 |     |                             | Default.                                                                                                                                               |
| 82053 |     | <b>STDOUT</b>               |                                                                                                                                                        |
| 82054 |     |                             | Not used.                                                                                                                                              |
| 82055 |     | <b>STDERR</b>               |                                                                                                                                                        |
| 82056 |     |                             | The standard error shall be used only for diagnostic messages.                                                                                         |
| 82057 |     | <b>OUTPUT FILES</b>         |                                                                                                                                                        |
| 82058 |     |                             | None.                                                                                                                                                  |
| 82059 |     | <b>EXTENDED DESCRIPTION</b> |                                                                                                                                                        |
| 82060 |     |                             | The <i>mode</i> operand shall be either a <i>symbolic_mode</i> expression or a non-negative octal integer. The                                         |
| 82061 |     |                             | <i>symbolic_mode</i> form is described by the grammar later in this section.                                                                           |
| 82062 |     |                             | Each <b>clause</b> shall specify an operation to be performed on the current file mode bits of each <i>file</i> .                                      |
| 82063 |     |                             | The operations shall be performed on each <i>file</i> in the order in which the <b>clauses</b> are specified.                                          |
| 82064 |     |                             | The <b>who</b> symbols <b>u</b> , <b>g</b> , and <b>o</b> shall specify the <i>user</i> , <i>group</i> , and <i>other</i> parts of the file mode bits, |
| 82065 |     |                             | respectively. A <b>who</b> consisting of the symbol <b>a</b> shall be equivalent to <b>ugo</b> .                                                       |
| 82066 |     |                             | The <b>perm</b> symbols <b>r</b> , <b>w</b> , and <b>x</b> represent the <i>read</i> , <i>write</i> , and <i>execute/search</i> portions of file mode  |
| 82067 |     |                             | bits, respectively. The <b>perm</b> symbol <b>s</b> shall represent the <i>set-user-ID-on-execution</i> (when <b>who</b>                               |
| 82068 |     |                             | contains or implies <b>u</b> ) and <i>set-group-ID-on-execution</i> (when <b>who</b> contains or implies <b>g</b> ) bits.                              |
| 82069 |     |                             | The <b>perm</b> symbol <b>X</b> shall represent the execute/search portion of the file mode bits if the file is a                                      |
| 82070 |     |                             | directory or if the current (unmodified) file mode bits have at least one of the execute bits                                                          |
| 82071 |     |                             | (S_IXUSR, S_IXGRP, or S_IXOTH) set. It shall be ignored if the file is not a directory and none of                                                     |
| 82072 |     |                             | the execute bits are set in the current file mode bits.                                                                                                |
| 82073 |     |                             | The <b>permcop</b> symbols <b>u</b> , <b>g</b> , and <b>o</b> shall represent the current permissions associated with the                              |
| 82074 |     |                             | user, group, and other parts of the file mode bits, respectively. For the remainder of this section,                                                   |
| 82075 |     |                             | <b>perm</b> refers to the non-terminals <b>perm</b> and <b>permcop</b> in the grammar.                                                                 |
| 82076 |     |                             | If multiple <b>actionlists</b> are grouped with a single <b>wholist</b> in the grammar, each <b>actionlist</b> shall be                                |
| 82077 |     |                             | applied in the order specified with that <b>wholist</b> . The <i>op</i> symbols shall represent the operation                                          |
| 82078 |     |                             | performed, as follows:                                                                                                                                 |
| 82079 |     | +                           | If <b>perm</b> is not specified, the '+' operation shall not change the file mode bits.                                                                |
| 82080 |     |                             | If <b>who</b> is not specified, the file mode bits represented by <b>perm</b> for the owner, group, and                                                |
| 82081 |     |                             | other permissions, except for those with corresponding bits in the file mode creation mask                                                             |
| 82082 |     |                             | of the invoking process, shall be set.                                                                                                                 |
| 82083 |     |                             | Otherwise, the file mode bits represented by the specified <b>who</b> and <b>perm</b> values shall be set.                                             |
| 82084 |     | -                           | If <b>perm</b> is not specified, the '-' operation shall not change the file mode bits.                                                                |
| 82085 |     |                             | If <b>who</b> is not specified, the file mode bits represented by <b>perm</b> for the owner, group, and                                                |
| 82086 |     |                             | other permissions, except for those with corresponding bits in the file mode creation mask                                                             |
| 82087 |     |                             | of the invoking process, shall be cleared.                                                                                                             |
| 82088 |     |                             | Otherwise, the file mode bits represented by the specified <b>who</b> and <b>perm</b> values shall be                                                  |
| 82089 |     |                             | cleared.                                                                                                                                               |
| 82090 |     | =                           | Clear the file mode bits specified by the <b>who</b> value, or, if no <b>who</b> value is specified, all of the                                        |
| 82091 |     |                             | file mode bits specified in this volume of POSIX.1-2008.                                                                                               |

If **perm** is not specified, the '=' operation shall make no further modifications to the file mode bits.

If **who** is not specified, the file mode bits represented by **perm** for the owner, group, and other permissions, except for those with corresponding bits in the file mode creation mask of the invoking process, shall be set.

Otherwise, the file mode bits represented by the specified **who** and **perm** values shall be set.

When using the symbolic mode form on a regular file, it is implementation-defined whether or not:

- Requests to set the set-user-ID-on-execution or set-group-ID-on-execution bit when all execute bits are currently clear and none are being set are ignored.
- Requests to clear all execute bits also clear the set-user-ID-on-execution and set-group-ID-on-execution bits.
- Requests to clear the set-user-ID-on-execution or set-group-ID-on-execution bits when all execute bits are currently clear are ignored. However, if the command *ls -l file* writes an *s* in the position indicating that the set-user-ID-on-execution or set-group-ID-on-execution is set, the commands *chmod u-s file* or *chmod g-s file*, respectively, shall not be ignored.

When using the symbolic mode form on other file types, it is implementation-defined whether or not requests to set or clear the set-user-ID-on-execution or set-group-ID-on-execution bits are honored.

If the **who** symbol **o** is used in conjunction with the **perm** symbol **s** with no other **who** symbols being specified, the set-user-ID-on-execution and set-group-ID-on-execution bits shall not be modified. It shall not be an error to specify the **who** symbol **o** in conjunction with the **perm** symbol **s**.

The **perm** symbol **t** shall specify the S\_ISVTX bit. When used with a file of type directory, it can be used with the **who** symbol **a**, or with no **who** symbol. It shall not be an error to specify a **who** symbol of **u**, **g**, or **o** in conjunction with the **perm** symbol **t**, but the meaning of these combinations is unspecified. The effect when using the **perm** symbol **t** with any file type other than directory is unspecified.

For an octal integer *mode* operand, the file mode bits shall be set absolutely.

For each bit set in the octal number, the corresponding file permission bit shown in the following table shall be set; all other file permission bits shall be cleared. For regular files, for each bit set in the octal number corresponding to the set-user-ID-on-execution or the set-group-ID-on-execution, bits shown in the following table shall be set; if these bits are not set in the octal number, they are cleared. For other file types, it is implementation-defined whether or not requests to set or clear the set-user-ID-on-execution or set-group-ID-on-execution bits are honored.

| Octal | Mode Bit | Octal | Mode Bit | Octal | Mode Bit | Octal | Mode Bit |
|-------|----------|-------|----------|-------|----------|-------|----------|
| 4000  | S_ISUID  | 0400  | S_IRUSR  | 0040  | S_IRGRP  | 0004  | S_IROTH  |
| 2000  | S_ISGID  | 0200  | S_IWUSR  | 0020  | S_IWGRP  | 0002  | S_IWOTH  |
| 1000  | S_ISVTX  | 0100  | S_IXUSR  | 0010  | S_IXGRP  | 0001  | S_IXOTH  |

When bits are set in the octal number other than those listed in the table above, the behavior is unspecified.

82134 **Grammar for chmod**

82135 The grammar and lexical conventions in this section describe the syntax for the *symbolic\_mode*  
 82136 operand. The general conventions for this style of grammar are described in [Section 1.3](#) (on page  
 82137 2309). A valid *symbolic\_mode* can be represented as the non-terminal symbol *symbolic\_mode* in  
 82138 the grammar. This formal syntax shall take precedence over the preceding text syntax  
 82139 description.

82140 The lexical processing is based entirely on single characters. Implementations need not allow  
 82141 <blank> characters within the single argument being processed.

82142 %start symbolic\_mode

82143 %%

82144 symbolic\_mode : clause  
 82145 | symbolic\_mode ',' clause  
 82146 ;

82147 clause : actionlist  
 82148 | wholist actionlist  
 82149 ;

82150 wholist : who  
 82151 | wholist who  
 82152 ;

82153 who : 'u' | 'g' | 'o' | 'a'  
 82154 ;

82155 actionlist : action  
 82156 | actionlist action  
 82157 ;

82158 action : op  
 82159 | op permlist  
 82160 | op permcopy  
 82161 ;

82162 permcopy : 'u' | 'g' | 'o'  
 82163 ;

82164 op : '+' | '-' | '='  
 82165 ;

82166 permlist : perm  
 82167 | perm permlist  
 82168 ;

82169 XSI perm : 'r' | 'w' | 'x' | 'X' | 's' | 't'  
 82170 ;

82171 **EXIT STATUS**

82172 The following exit values shall be returned:

82173 0 The utility executed successfully and all requested changes were made.

82174 >0 An error occurred.

## CONSEQUENCES OF ERRORS

Default.

## APPLICATION USAGE

Some implementations of the *chmod* utility change the mode of a directory before the files in the directory when performing a recursive (**-R** option) change; others change the directory mode after the files in the directory. If an application tries to remove read or search permission for a file hierarchy, the removal attempt fails if the directory is changed first; on the other hand, trying to re-enable permissions to a restricted hierarchy fails if directories are changed last. Users should not try to make a hierarchy inaccessible to themselves.

Some implementations of *chmod* never used the *umask* of the process when changing modes; systems conformant with this volume of POSIX.1-2008 do so when **who** is not specified. Note the difference between:

```
chmod a-w file
```

which removes all write permissions, and:

```
chmod -- -w file
```

which removes write permissions that would be allowed if **file** was created with the same *umask*.

Conforming applications should never assume that they know how the set-user-ID and set-group-ID bits on directories are interpreted.

## EXAMPLES

| Mode         | Results                                                                                            |
|--------------|----------------------------------------------------------------------------------------------------|
| <i>a+=</i>   | Equivalent to <i>a+,a=</i> ; clears all file mode bits.                                            |
| <i>go+-w</i> | Equivalent to <i>go+,go-w</i> ; clears group and other write bits.                                 |
| <i>g=o-w</i> | Equivalent to <i>g=o,g-w</i> ; sets group bit to match other bits and then clears group write bit. |
| <i>g-r+w</i> | Equivalent to <i>g-r,g+w</i> ; clears group read bit and sets group write bit.                     |
| <i>uo=g</i>  | Sets owner bits to match group bits and sets other bits to match group bits.                       |

## RATIONALE

The functionality of *chmod* is described substantially through references to concepts defined in the System Interfaces volume of POSIX.1-2008. In this way, there is less duplication of effort required for describing the interactions of permissions. However, the behavior of this utility is not described in terms of the *chmod()* function from the System Interfaces volume of POSIX.1-2008 because that specification requires certain side-effects upon alternate file access control mechanisms that might not be appropriate, depending on the implementation.

Implementations that support mandatory file and record locking as specified by the 1984 /usr/group standard historically used the combination of set-group-ID bit set and group execute bit clear to indicate mandatory locking. This condition is usually set or cleared with the symbolic mode **perm** symbol **l** instead of the **perm** symbols **s** and **x** so that the mandatory locking mode is not changed without explicit indication that that was what the user intended. Therefore, the details on how the implementation treats these conditions must be defined in the documentation. This volume of POSIX.1-2008 does not require mandatory locking (nor does the System Interfaces volume of POSIX.1-2008), but does allow it as an extension. However, this volume of POSIX.1-2008 does require that the *ls* and *chmod* utilities work consistently in this

area. If *ls -l file* indicates that the set-group-ID bit is set, *chmod g-s file* must clear it (assuming appropriate privileges exist to change modes).

The System V and BSD versions use different exit status codes. Some implementations used the exit status as a count of the number of errors that occurred; this practice is unworkable since it can overflow the range of valid exit status values. This problem is avoided here by specifying only 0 and >0 as exit values.

The System Interfaces volume of POSIX.1-2008 indicates that implementation-defined restrictions may cause the S\_ISUID and S\_ISGID bits to be ignored. This volume of POSIX.1-2008 allows the *chmod* utility to choose to modify these bits before calling *chmod()* (or some function providing equivalent capabilities) for non-regular files. Among other things, this allows implementations that use the set-user-ID and set-group-ID bits on directories to enable extended features to handle these extensions in an intelligent manner.

The **X perm** symbol was adopted from BSD-based systems because it provides commonly desired functionality when doing recursive (**-R** option) modifications. Similar functionality is not provided by the *find* utility. Historical BSD versions of *chmod*, however, only supported **X** with *op+*; it has been extended in this volume of POSIX.1-2008 because it is also useful with *op=*. (It has also been added for *op-* even though it duplicates **x**, in this case, because it is intuitive and easier to explain.)

The grammar was extended with the *permcopy* non-terminal to allow historical-practice forms of symbolic modes like **o=u -g** (that is, set the “other” permissions to the permissions of “owner” minus the permissions of “group”).

## FUTURE DIRECTIONS

None.

## SEE ALSO

*ls*, *umask*

XBD [Section 4.4](#) (on page 108), [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 215)

XSH *chmod()*

## CHANGE HISTORY

First released in Issue 2.

## Issue 6

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- Octal modes have been kept and made mandatory despite being marked obsolescent in the ISO POSIX-2: 1993 standard.

IEEE PASC Interpretation 1003.2 #172 is applied, changing the CONSEQUENCES OF ERRORS section to “Default.”.

The Open Group Base Resolution bwg2001-010 is applied, adding the description of the S\_ISVTX bit and the **t perm** symbol as part of the XSI option.

IEEE Std 1003.1-2001/Cor 1-2002, item XCU/TC1/D6/16 is applied, changing the XSI shaded text in the EXTENDED DESCRIPTION from:

“The **perm** symbol **t** shall specify the S\_ISVTX bit and shall apply to directories only. The effect when using it with any other file type is unspecified. It can be used with the **who** symbols **o**, **a**, or with no **who** symbol. It shall not be an error to specify a **who** symbol of **u** or **g** in conjunction with the **perm** symbol **t**; it shall be ignored for **u** and **g**.”

82265 to:  
82266 “The **perm** symbol **t** shall specify the S\_ISVTX bit. When used with a file of type directory,  
82267 it can be used with the **who** symbol **a**, or with no **who** symbol. It shall not be an error to  
82268 specify a **who** symbol of **u**, **g**, or **o** in conjunction with the **perm** symbol **t**, but the meaning  
82269 of these combinations is unspecified. The effect when using the **perm** symbol **t** with any  
82270 file type other than directory is unspecified.”

82271 This change is to permit historical behavior.

82272 **Issue 7**

82273 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

82274 Austin Group Interpretation 1003.1-2001 #130 is applied, adding text to the DESCRIPTION  
82275 about about marking for update the last file status change timestamp of the file.



82276 **NAME**82277 **chown** — change the file ownership82278 **SYNOPSIS**82279 `chown [-h] owner[:group] file...`82280 `chown -R [-H|-L|-P] owner[:group] file...`82281 **DESCRIPTION**82282 The *chown* utility shall set the user ID of the file named by each *file* operand to the user ID  
82283 specified by the *owner* operand.82284 For each *file* operand, or, if the **-R** option is used, each file encountered while walking the  
82285 directory trees specified by the *file* operands, the *chown* utility shall perform actions equivalent to  
82286 the *chown()* function defined in the System Interfaces volume of POSIX.1-2008, called with the  
82287 following arguments:

- 82288 1. The *file* operand shall be used as the *path* argument.
- 82289 2. The user ID indicated by the *owner* portion of the first operand shall be used as the *owner*  
82290 argument.
- 82291 3. If the *group* portion of the first operand is given, the group ID indicated by it shall be used  
82292 as the *group* argument; otherwise, the group ownership shall not be changed.

82293 Unless *chown* is invoked by a process with appropriate privileges, the set-user-ID and set-group-  
82294 ID bits of a regular file shall be cleared upon successful completion; the set-user-ID and set-  
82295 group-ID bits of other file types may be cleared.82296 **OPTIONS**82297 The *chown* utility shall conform to XBD [Section 12.2](#) (on page 215).

82298 The following options shall be supported by the implementation:

- 82299 **-h** For each file operand that names a file of type symbolic link, *chown* shall attempt to  
82300 set the user ID of the symbolic link. If a group ID was specified, for each file  
82301 operand that names a file of type symbolic link, *chown* shall attempt to set the  
82302 group ID of the symbolic link.
- 82303 **-H** If the **-R** option is specified and a symbolic link referencing a file of type directory  
82304 is specified on the command line, *chown* shall change the user ID (and group ID, if  
82305 specified) of the directory referenced by the symbolic link and all files in the file  
82306 hierarchy below it.
- 82307 **-L** If the **-R** option is specified and a symbolic link referencing a file of type directory  
82308 is specified on the command line or encountered during the traversal of a file  
82309 hierarchy, *chown* shall change the user ID (and group ID, if specified) of the  
82310 directory referenced by the symbolic link and all files in the file hierarchy below it.
- 82311 **-P** If the **-R** option is specified and a symbolic link is specified on the command line  
82312 or encountered during the traversal of a file hierarchy, *chown* shall change the  
82313 owner ID (and group ID, if specified) of the symbolic link. The *chown* utility shall  
82314 not follow the symbolic link to any other part of the file hierarchy.
- 82315 **-R** Recursively change file user and group IDs. For each *file* operand that names a  
82316 directory, *chown* shall change the user ID (and group ID, if specified) of the  
82317 directory and all files in the file hierarchy below it. Unless a **-H**, **-L**, or **-P** option is  
82318 specified, it is unspecified which of these options will be used as the default.

82319 Specifying more than one of the mutually-exclusive options **-H**, **-L**, and **-P** shall not be

82320 considered an error. The last option specified shall determine the behavior of the utility.

#### 82321 OPERANDS

82322 The following operands shall be supported:

82323 *owner[:group]* A user ID and optional group ID to be assigned to *file*. The *owner* portion of this  
 82324 operand shall be a user name from the user database or a numeric user ID. Either  
 82325 specifies a user ID which shall be given to each file named by one of the *file*  
 82326 operands. If a numeric *owner* operand exists in the user database as a user name,  
 82327 the user ID number associated with that user name shall be used as the user ID.  
 82328 Similarly, if the *group* portion of this operand is present, it shall be a group name  
 82329 from the group database or a numeric group ID. Either specifies a group ID which  
 82330 shall be given to each file. If a numeric group operand exists in the group database  
 82331 as a group name, the group ID number associated with that group name shall be  
 82332 used as the group ID.

82333 *file* A pathname of a file whose user ID is to be modified.

#### 82334 STDIN

82335 Not used.

#### 82336 INPUT FILES

82337 None.

#### 82338 ENVIRONMENT VARIABLES

82339 The following environment variables shall affect the execution of *chown*:

82340 *LANG* Provide a default value for the internationalization variables that are unset or null.  
 82341 (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 82342 variables used to determine the values of locale categories.)

82343 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
 82344 internationalization variables.

82345 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
 82346 characters (for example, single-byte as opposed to multi-byte characters in  
 82347 arguments).

82348 *LC\_MESSAGES*

82349 Determine the locale that should be used to affect the format and contents of  
 82350 diagnostic messages written to standard error.

82351 *XSI* *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

#### 82352 ASYNCHRONOUS EVENTS

82353 Default.

#### 82354 STDOUT

82355 Not used.

#### 82356 STDERR

82357 The standard error shall be used only for diagnostic messages.

#### 82358 OUTPUT FILES

82359 None.

**82360 EXTENDED DESCRIPTION**

82361 None.

**82362 EXIT STATUS**

82363 The following exit values shall be returned:

82364 0 The utility executed successfully and all requested changes were made.

82365 >0 An error occurred.

**82366 CONSEQUENCES OF ERRORS**

82367 Default.

**82368 APPLICATION USAGE**

82369 Only the owner of a file or the user with appropriate privileges may change the owner or group  
82370 of a file.

82371 Some implementations restrict the use of *chown* to a user with appropriate privileges.

**82372 EXAMPLES**

82373 None.

**82374 RATIONALE**

82375 The System V and BSD versions use different exit status codes. Some implementations used the  
82376 exit status as a count of the number of errors that occurred; this practice is unworkable since it  
82377 can overflow the range of valid exit status values. These are masked by specifying only 0 and >0  
82378 as exit values.

82379 The functionality of *chown* is described substantially through references to functions in the  
82380 System Interfaces volume of POSIX.1-2008. In this way, there is no duplication of effort required  
82381 for describing the interactions of permissions, multiple groups, and so on.

82382 The 4.3 BSD method of specifying both owner and group was included in this volume of  
82383 POSIX.1-2008 because:

- 82384 • There are cases where the desired end condition could not be achieved using the *chgrp* and  
82385 *chown* (that only changed the user ID) utilities. (If the current owner is not a member of the  
82386 desired group and the desired owner is not a member of the current group, the *chown()*  
82387 function could fail unless both owner and group are changed at the same time.)
- 82388 • Even if they could be changed independently, in cases where both are being changed, there  
82389 is a 100% performance penalty caused by being forced to invoke both utilities.

82390 The BSD syntax *user[.group]* was changed to *user[:group]* in this volume of POSIX.1-2008 because  
82391 the <period> is a valid character in login names (as specified by the Base Definitions volume of  
82392 POSIX.1-2008, login names consist of characters in the portable filename character set). The  
82393 <colon> character was chosen as the replacement for the <period> character because it would  
82394 never be allowed as a character in a user name or group name on historical implementations.

82395 The **-R** option is considered by some observers as an undesirable departure from the historical  
82396 UNIX system tools approach; since a tool, *find*, already exists to recurse over directories, there  
82397 seemed to be no good reason to require other tools to have to duplicate that functionality.  
82398 However, the **-R** option was deemed an important user convenience, is far more efficient than  
82399 forking a separate process for each element of the directory hierarchy, and is in widespread  
82400 historical use.

82401 **FUTURE DIRECTIONS**

82402 None.

82403 **SEE ALSO**82404 *chgrp, chmod*82405 XBD [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 215)82406 XSH *chown()*82407 **CHANGE HISTORY**

82408 First released in Issue 2.

82409 **Issue 6**82410 New options **-h**, **-H**, **-L**, and **-P** are added to align with the IEEE P1003.2b draft standard. These  
82411 options affect the processing of symbolic links.

82412 The normative text is reworded to avoid use of the term “must” for application requirements.

82413 IEEE PASC Interpretation 1003.2 #172 is applied, changing the CONSEQUENCES OF ERRORS  
82414 section to “Default.”.82415 The “otherwise, ...” text in item 3. of the DESCRIPTION is changed to “otherwise, the group  
82416 ownership shall not be changed”.82417 IEEE Std 1003.1-2001/Cor 1-2002, item XCU/TC1/D6/17 is applied, changing the SYNOPSIS to  
82418 make it clear that **-h** and **-R** are optional.82419 **Issue 7**82420 SD5-XCU-ERN-9 is applied, removing the **-R** from the first line of the SYNOPSIS.

82421 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

82422 The description of the **-h** and **-P** options is revised.

82423 **NAME**

82424 cksum — write file checksums and sizes

82425 **SYNOPSIS**82426 cksum [*file*...]82427 **DESCRIPTION**

82428 The *cksum* utility shall calculate and write to standard output a cyclic redundancy check (CRC)  
 82429 for each input file, and also write to standard output the number of octets in each file. The CRC  
 82430 used is based on the polynomial used for CRC error checking in the ISO/IEC 8802-3:1996  
 82431 standard (Ethernet).

82432 The encoding for the CRC checksum is defined by the generating polynomial:

$$82433 G(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$$

82434 Mathematically, the CRC value corresponding to a given file shall be defined by the following  
 82435 procedure:

- 82436 1. The *n* bits to be evaluated are considered to be the coefficients of a mod 2 polynomial  
 82437  $M(x)$  of degree  $n-1$ . These *n* bits are the bits from the file, with the most significant bit  
 82438 being the most significant bit of the first octet of the file and the last bit being the least  
 82439 significant bit of the last octet, padded with zero bits (if necessary) to achieve an integral  
 82440 number of octets, followed by one or more octets representing the length of the file as a  
 82441 binary value, least significant octet first. The smallest number of octets capable of  
 82442 representing this integer shall be used.
- 82443 2.  $M(x)$  is multiplied by  $x^{32}$  (that is, shifted left 32 bits) and divided by  $G(x)$  using mod 2  
 82444 division, producing a remainder  $R(x)$  of degree  $\leq 31$ .
- 82445 3. The coefficients of  $R(x)$  are considered to be a 32-bit sequence.
- 82446 4. The bit sequence is complemented and the result is the CRC.

82447 **OPTIONS**

82448 None.

82449 **OPERANDS**

82450 The following operand shall be supported:

82451 *file* A pathname of a file to be checked. If no *file* operands are specified, the standard  
 82452 input shall be used.

82453 **STDIN**

82454 The standard input shall be used if no *file* operands are specified, and shall be used if a *file*  
 82455 operand is '-' and the implementation treats the '-' as meaning standard input. Otherwise,  
 82456 the standard input shall not be used. See the INPUT FILES section.

82457 **INPUT FILES**

82458 The input files can be any file type.

82459 **ENVIRONMENT VARIABLES**82460 The following environment variables shall affect the execution of *cksum*:

- 82461 *LANG* Provide a default value for the internationalization variables that are unset or null.  
 82462 (See XBD Section 8.2 (on page 174) for the precedence of internationalization  
 82463 variables used to determine the values of locale categories.)
- 82464 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
 82465 internationalization variables.

82466 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
 82467 characters (for example, single-byte as opposed to multi-byte characters in  
 82468 arguments).

82469 *LC\_MESSAGES*  
 82470 Determine the locale that should be used to affect the format and contents of  
 82471 diagnostic messages written to standard error.

82472 XSI *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

82473 **ASYNCHRONOUS EVENTS**  
 82474 Default.

82475 **STDOUT**  
 82476 For each file processed successfully, the *cksum* utility shall write in the following format:  
 82477 "%u %d %s\n", <checksum>, <# of octets>, <pathname>  
 82478 If no *file* operand was specified, the pathname and its leading <space> shall be omitted.

82479 **STDERR**  
 82480 The standard error shall be used only for diagnostic messages.

82481 **OUTPUT FILES**  
 82482 None.

82483 **EXTENDED DESCRIPTION**  
 82484 None.

82485 **EXIT STATUS**  
 82486 The following exit values shall be returned:  
 82487 0 All files were processed successfully.  
 82488 >0 An error occurred.

82489 **CONSEQUENCES OF ERRORS**  
 82490 Default.

82491 **APPLICATION USAGE**  
 82492 The *cksum* utility is typically used to quickly compare a suspect file against a trusted version of  
 82493 the same, such as to ensure that files transmitted over noisy media arrive intact. However, this  
 82494 comparison cannot be considered cryptographically secure. The chances of a damaged file  
 82495 producing the same CRC as the original are small; deliberate deception is difficult, but probably  
 82496 not impossible.

82497 Although input files to *cksum* can be any type, the results need not be what would be expected  
 82498 on character special device files or on file types not described by the System Interfaces volume of  
 82499 POSIX.1-2008. Since this volume of POSIX.1-2008 does not specify the block size used when  
 82500 doing input, checksums of character special files need not process all of the data in those files.

82501 The algorithm is expressed in terms of a bitstream divided into octets. If a file is transmitted  
 82502 between two systems and undergoes any data transformation (such as changing little-endian  
 82503 byte ordering to big-endian), identical CRC values cannot be expected. Implementations  
 82504 performing such transformations may extend *cksum* to handle such situations.

82505 **EXAMPLES**  
 82506 None.

**RATIONALE**

The following C-language program can be used as a model to describe the algorithm. It assumes that a **char** is one octet. It also assumes that the entire file is available for one pass through the function. This was done for simplicity in demonstrating the algorithm, rather than as an implementation model.

```
static unsigned long crctab[] = {
0x00000000,
0x04c11db7, 0x09823b6e, 0x0d4326d9, 0x130476dc, 0x17c56b6b,
0x1a864db2, 0x1e475005, 0x2608edb8, 0x22c9f00f, 0x2f8ad6d6,
0x2b4bcb61, 0x350c9b64, 0x31cd86d3, 0x3c8ea00a, 0x384fbdbd,
0x4c11db70, 0x48d0c6c7, 0x4593e01e, 0x4152fda9, 0x5f15adac,
0x5bd4b01b, 0x569796c2, 0x52568b75, 0x6a1936c8, 0x6ed82b7f,
0x639b0da6, 0x675a1011, 0x791d4014, 0x7ddc5da3, 0x709f7b7a,
0x745e66cd, 0x9823b6e0, 0x9ce2ab57, 0x91a18d8e, 0x95609039,
0x8b27c03c, 0x8fe6dd8b, 0x82a5fb52, 0x8664e6e5, 0xbe2b5b58,
0xbaea46ef, 0xb7a96036, 0xb3687d81, 0xad2f2d84, 0xa9ee3033,
0xa4ad16ea, 0xa06c0b5d, 0xd4326d90, 0xd0f37027, 0xddb056fe,
0xd9714b49, 0xc7361b4c, 0xc3f706fb, 0xceb42022, 0xca753d95,
0xf23a8028, 0xf6fb9d9f, 0xfbb8bb46, 0xff79a6f1, 0xe13ef6f4,
0xe5ffeb43, 0xe8bccd9a, 0xec7dd02d, 0x34867077, 0x30476dc0,
0x3d044b19, 0x39c556ae, 0x278206ab, 0x23431b1c, 0x2e003dc5,
0x2ac12072, 0x128e9dcf, 0x164f8078, 0x1b0ca6a1, 0x1fcdabb16,
0x018aeb13, 0x054bf6a4, 0x0808d07d, 0x0cc9cdca, 0x7897ab07,
0x7c56b6b0, 0x71159069, 0x75d48dde, 0x6b93dddb, 0x6f52c06c,
0x6211e6b5, 0x66d0fb02, 0x5e9f46bf, 0x5a5e5b08, 0x571d7ddl,
0x53dc6066, 0x4d9b3063, 0x495a2dd4, 0x44190b0d, 0x40d816ba,
0xaca5c697, 0xa864db20, 0xa527fdf9, 0xa1e6e04e, 0xbfa1b04b,
0xbb60adfc, 0xb6238b25, 0xb2e29692, 0x8aad2b2f, 0x8e6c3698,
0x832f1041, 0x87ee0df6, 0x99a95df3, 0x9d684044, 0x902b669d,
0x94ea7b2a, 0xe0b41de7, 0xe4750050, 0xe9362689, 0xedf73b3e,
0xf3b06b3b, 0xf771768c, 0xfa325055, 0xfef34de2, 0xc6bcf05f,
0xc27dede8, 0xcf3ecb31, 0xcbffd686, 0xd5b88683, 0xd1799b34,
0xdc3abded, 0xd8fba05a, 0x690ce0ee, 0x6dcdfd59, 0x608edb80,
0x644fc637, 0x7a089632, 0x7ec98b85, 0x738aad5c, 0x774bb0eb,
0x4f040d56, 0x4bc510e1, 0x46863638, 0x42472b8f, 0x5c007b8a,
0x58c1663d, 0x558240e4, 0x51435d53, 0x251d3b9e, 0x21dc2629,
0x2c9f00f0, 0x285e1d47, 0x36194d42, 0x32d850f5, 0x3f9b762c,
0x3b5a6b9b, 0x0315d626, 0x07d4cb91, 0x0a97ed48, 0x0e56f0ff,
0x1011a0fa, 0x14d0bd4d, 0x19939b94, 0x1d528623, 0xf12f560e,
0xf5ee4bb9, 0xf8ad6d60, 0xfc6c70d7, 0xe22b20d2, 0xe6ea3d65,
0xeba91bbc, 0xef68060b, 0xd727bbb6, 0xd3e6a601, 0xdea580d8,
0xda649d6f, 0xc423cd6a, 0xc0e2d0dd, 0xcda1f604, 0xc960ebb3,
0xbd3e8d7e, 0xb9ff90c9, 0xb4bcb610, 0xb07daba7, 0xae3afba2,
0xaafbe615, 0xa7b8c0cc, 0xa379dd7b, 0x9b3660c6, 0x9ff77d71,
0x92b45ba8, 0x9675461f, 0x8832161a, 0x8cf30bad, 0x81b02d74,
0x857130c3, 0x5d8a9099, 0x594b8d2e, 0x5408abf7, 0x50c9b640,
0x4e8ee645, 0x4a4ffb2, 0x470cdd2b, 0x43cdc09c, 0x7b827d21,
0x7f436096, 0x7200464f, 0x76c15bf8, 0x68860bfd, 0x6c47164a,
0x61043093, 0x65c52d24, 0x119b4be9, 0x155a565e, 0x18197087,
0x1cd86d30, 0x029f3d35, 0x065e2082, 0x0b1d065b, 0x0fdc1bec,
0x3793a651, 0x3352bbe6, 0x3e119d3f, 0x3ad08088, 0x2497d08d,
```

```

82558 0x2056cd3a, 0x2d15ebe3, 0x29d4f654, 0xc5a92679, 0xc1683bce,
82559 0xcc2b1d17, 0xc8ea00a0, 0xd6ad50a5, 0xd26c4d12, 0xdf2f6bcb,
82560 0xdbee767c, 0xe3a1cbc1, 0xe760d676, 0xea23f0af, 0xee2ed18,
82561 0xf0a5bd1d, 0xf464a0aa, 0xf9278673, 0xfde69bc4, 0x89b8fd09,
82562 0x8d79e0be, 0x803ac667, 0x84fbd0, 0x9abc8bd5, 0x9e7d9662,
82563 0x933eb0bb, 0x97ffad0c, 0xafb010b1, 0xab710d06, 0xa6322bdf,
82564 0xa2f33668, 0xbcb4666d, 0xb8757bda, 0xb5365d03, 0xb1f740b4
82565 };

82566 unsigned long memcrc(const unsigned char *b, size_t n)
82567 {
82568 /* Input arguments:
82569 * const unsigned char* b == byte sequence to checksum
82570 * size_t n == length of sequence
82571 */

82572 register size_t i;
82573 register unsigned c, s = 0;

82574 for (i = n; i > 0; --i) {
82575 c = *b++;
82576 s = (s << 8) ^ crctab[(s >> 24) ^ c];
82577 }

82578 /* Extend with the length of the string. */
82579 while (n != 0) {
82580 c = n & 0377;
82581 n >>= 8;
82582 s = (s << 8) ^ crctab[(s >> 24) ^ c];
82583 }

82584 return ~s;
82585 }

```

The historical practice of writing the number of “blocks” has been changed to writing the number of octets, since the latter is not only more useful, but also since historical implementations have not been consistent in defining what a “block” meant.

The algorithm used was selected to increase the operational robustness of *cksum*. Neither the System V nor BSD *sum* algorithm was selected. Since each of these was different and each was the default behavior on those systems, no realistic compromise was available if either were selected—some set of historical applications would break. Therefore, the name was changed to *cksum*. Although the historical *sum* commands will probably continue to be provided for many years, programs designed for portability across systems should use the new name.

The algorithm selected is based on that used by the ISO/IEC 8802-3:1996 standard (Ethernet) for the frame check sequence field. The algorithm used does not match the technical definition of a *checksum*; the term is used for historical reasons. The length of the file is included in the CRC calculation because this parallels inclusion of a length field by Ethernet in its CRC, but also because it guards against inadvertent collisions between files that begin with different series of zero octets. The chance that two different files produce identical CRCs is much greater when their lengths are not considered. Keeping the length and the checksum of the file itself separate would yield a slightly more robust algorithm, but historical usage has always been that a single number (the checksum as printed) represents the signature of the file. It was decided that historical usage was the more important consideration.



Early proposals contained modifications to the Ethernet algorithm that involved extracting table values whenever an intermediate result became zero. This was demonstrated to be less robust than the current method and mathematically difficult to describe or justify.

The calculation used is identical to that given in pseudo-code in the referenced Sarwate article. The pseudo-code rendition is:

```
X <- 0; Y <- 0;
for i <- m -1 step -1 until 0 do
 begin
 T <- X(1) ^ A[i];
 X(1) <- X(0); X(0) <- Y(1); Y(1) <- Y(0); Y(0) <- 0;
 comment: f[T] and f'[T] denote the T-th words in the
 table f and f' ;
 X <- X ^ f[T]; Y <- Y ^ f'[T];
 end
```

The pseudo-code is reproduced exactly as given; however, note that in the case of *cksum*, **A[i]** represents a byte of the file, the words **X** and **Y** are treated as a single 32-bit value, and the tables **f** and **f'** are a single table containing 32-bit values.

The referenced Sarwate article also discusses generating the table.

#### **FUTURE DIRECTIONS**

None.

#### **SEE ALSO**

XBD [Chapter 8](#) (on page 173)

#### **CHANGE HISTORY**

First released in Issue 4.

#### **Issue 7**

Austin Group Interpretation 1003.1-2001 #092 is applied.

SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

POSIX.1-2008, Technical Corrigendum 1, XCU/TC1-2008/0081 [446] is applied.

+

82633 **NAME**

82634 cmp — compare two files

82635 **SYNOPSIS**82636 cmp [-l|-s] *file1 file2*82637 **DESCRIPTION**

82638 The *cmp* utility shall compare two files. The *cmp* utility shall write no output if the files are the  
 82639 same. Under default options, if they differ, it shall write to standard output the byte and line  
 82640 number at which the first difference occurred. Bytes and lines shall be numbered beginning with  
 82641 1.

82642 **OPTIONS**82643 The *cmp* utility shall conform to XBD [Section 12.2](#) (on page 215).

82644 The following options shall be supported:

82645 -l (Lowercase ell.) Write the byte number (decimal) and the differing bytes (octal) for  
 82646 each difference.

82647 -s Write nothing for differing files; return exit status only.

82648 **OPERANDS**

82649 The following operands shall be supported:

82650 *file1* A pathname of the first file to be compared. If *file1* is '-', the standard input shall  
 82651 be used.

82652 *file2* A pathname of the second file to be compared. If *file2* is '-', the standard input  
 82653 shall be used.

82654 If both *file1* and *file2* refer to standard input or refer to the same FIFO special, block special, or  
 82655 character special file, the results are undefined.

82656 **STDIN**

82657 The standard input shall be used only if the *file1* or *file2* operand refers to standard input. See the  
 82658 INPUT FILES section.

82659 **INPUT FILES**

82660 The input files can be any file type.

82661 **ENVIRONMENT VARIABLES**82662 The following environment variables shall affect the execution of *cmp*:

82663 LANG Provide a default value for the internationalization variables that are unset or null.  
 82664 (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 82665 variables used to determine the values of locale categories.)

82666 LC\_ALL If set to a non-empty string value, override the values of all the other  
 82667 internationalization variables.

82668 LC\_CTYPE Determine the locale for the interpretation of sequences of bytes of text data as  
 82669 characters (for example, single-byte as opposed to multi-byte characters in  
 82670 arguments).

82671 LC\_MESSAGES

82672 Determine the locale that should be used to affect the format and contents of  
 82673 diagnostic messages written to standard error and informative messages written to  
 82674 standard output.

82675 XSI **NLSPATH** Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

82676 **ASYNCHRONOUS EVENTS**

82677 Default.

82678 **STDOUT**

82679 In the POSIX locale, results of the comparison shall be written to standard output. When no

82680 options are used, the format shall be:

82681 "%s %s differ: char %d, line %d\n", *file1*, *file2*,

82682 <*byte number*>, <*line number*>

82683 When the **-l** option is used, the format shall be:

82684 "%d %o %o\n", <*byte number*>, <*differing byte*>,

82685 <*differing byte*>

82686 for each byte that differs. The first <*differing byte*> number is from *file1* while the second is from

82687 *file2*. In both cases, <*byte number*> shall be relative to the beginning of the file, beginning with 1.

82688 No output shall be written to standard output when the **-s** option is used.

82689 **STDERR**

82690 The standard error shall be used only for diagnostic messages. If the **-l** option is used and *file1*

82691 and *file2* differ in length, or if the **-s** option is not used and *file1* and *file2* are identical for the

82692 entire length of the shorter file, in the POSIX locale the following diagnostic message shall be

82693 written:

82694 "cmp: EOF on %s%s\n", <*name of shorter file*>, <*additional info*>

82695 The <*additional info*> field shall either be null or a string that starts with a <blank> and contains

82696 no <newline> characters. Some implementations report on the number of lines in this case.

82697 **OUTPUT FILES**

82698 None.

82699 **EXTENDED DESCRIPTION**

82700 None.

82701 **EXIT STATUS**

82702 The following exit values shall be returned:

82703 0 The files are identical.

82704 1 The files are different; this includes the case where one file is identical to the first part of the

82705 other.

82706 >1 An error occurred.

82707 **CONSEQUENCES OF ERRORS**

82708 Default.

**APPLICATION USAGE**

Although input files to *cmp* can be any type, the results might not be what would be expected on character special device files or on file types not described by the System Interfaces volume of POSIX.1-2008. Since this volume of POSIX.1-2008 does not specify the block size used when doing input, comparisons of character special files need not compare all of the data in those files.

For files which are not text files, line numbers simply reflect the presence of a <newline>, without any implication that the file is organized into lines.

**EXAMPLES**

None.

**RATIONALE**

The global language in [Section 1.4](#) (on page 2310) indicates that using two mutually-exclusive options together produces unspecified results. Some System V implementations consider the option usage:

```
cmp -l -s ...
```

to be an error. They also treat:

```
cmp -s -l ...
```

as if no options were specified. Both of these behaviors are considered bugs, but are allowed.

The word **char** in the standard output format comes from historical usage, even though it is actually a byte number. When *cmp* is supported in other locales, implementations are encouraged to use the word *byte* or its equivalent in another language. Users should not interpret this difference to indicate that the functionality of the utility changed between locales.

Some implementations report on the number of lines in the identical-but-shorter file case. This is allowed by the inclusion of the <additional info> fields in the output format. The restriction on having a leading <blank> and no <newline> characters is to make parsing for the filename easier. It is recognized that some filenames containing white-space characters make parsing difficult anyway, but the restriction does aid programs used on systems where the names are predominantly well behaved.

**FUTURE DIRECTIONS**

None.

**SEE ALSO**

[comm](#), [diff](#)

XBD [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 215)

**CHANGE HISTORY**

First released in Issue 2.

**Issue 7**

SD5-XCU-ERN-96 is applied, updating the STDERR section.

SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

82746 **NAME**

82747       comm — select or reject lines common to two files

82748 **SYNOPSIS**82749       comm [-123] *file1 file2*82750 **DESCRIPTION**

82751       The *comm* utility shall read *file1* and *file2*, which should be ordered in the current collating sequence, and produce three text columns as output: lines only in *file1*, lines only in *file2*, and lines in both files.

82754       If the lines in both files are not ordered according to the collating sequence of the current locale, the results are unspecified.

82756 **OPTIONS**82757       The *comm* utility shall conform to XBD [Section 12.2](#) (on page 215).

82758       The following options shall be supported:

82759       -1       Suppress the output column of lines unique to *file1*.82760       -2       Suppress the output column of lines unique to *file2*.82761       -3       Suppress the output column of lines duplicated in *file1* and *file2*.82762 **OPERANDS**

82763       The following operands shall be supported:

82764       *file1*       A pathname of the first file to be compared. If *file1* is '-', the standard input shall be used.

82766       *file2*       A pathname of the second file to be compared. If *file2* is '-', the standard input shall be used.

82768       If both *file1* and *file2* refer to standard input or to the same FIFO special, block special, or character special file, the results are undefined.

82770 **STDIN**

82771       The standard input shall be used only if one of the *file1* or *file2* operands refers to standard input. See the INPUT FILES section.

82773 **INPUT FILES**

82774       The input files shall be text files.

82775 **ENVIRONMENT VARIABLES**82776       The following environment variables shall affect the execution of *comm*:

82777       LANG       Provide a default value for the internationalization variables that are unset or null. (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization variables used to determine the values of locale categories.)

82780       LC\_ALL       If set to a non-empty string value, override the values of all the other internationalization variables.

82782       LC\_COLLATE

82783       Determine the locale for the collating sequence *comm* expects to have been used when the input files were sorted.

82785       LC\_CTYPE       Determine the locale for the interpretation of sequences of bytes of text data as characters (for example, single-byte as opposed to multi-byte characters in arguments and input files).

82788 *LC\_MESSAGES*  
 82789 Determine the locale that should be used to affect the format and contents of  
 82790 diagnostic messages written to standard error.

82791 XSI *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

82792 **ASYNCHRONOUS EVENTS**  
 82793 Default.

82794 **STDOUT**  
 82795 The *comm* utility shall produce output depending on the options selected. If the **-1**, **-2**, and **-3**  
 82796 options are all selected, *comm* shall write nothing to standard output.

82797 If the **-1** option is not selected, lines contained only in *file1* shall be written using the format:  
 82798 "%s\n", <line in file1>

82799 If the **-2** option is not selected, lines contained only in *file2* are written using the format:  
 82800 "%s%s\n", <lead>, <line in file2>

82801 where the string <lead> is as follows:  
 82802 <tab> The **-1** option is not selected.  
 82803 null string The **-1** option is selected.

82804 If the **-3** option is not selected, lines contained in both files shall be written using the format:  
 82805 "%s%s\n", <lead>, <line in both>

82806 where the string <lead> is as follows:  
 82807 <tab><tab> Neither the **-1** nor the **-2** option is selected.  
 82808 <tab> Exactly one of the **-1** and **-2** options is selected.  
 82809 null string Both the **-1** and **-2** options are selected.

82810 If the input files were ordered according to the collating sequence of the current locale, the lines  
 82811 written shall be in the collating sequence of the original lines.

82812 **STDERR**  
 82813 The standard error shall be used only for diagnostic messages.

82814 **OUTPUT FILES**  
 82815 None.

82816 **EXTENDED DESCRIPTION**  
 82817 None.

82818 **EXIT STATUS**  
 82819 The following exit values shall be returned:  
 82820 0 All input files were successfully output as specified.  
 82821 >0 An error occurred.

82822 **CONSEQUENCES OF ERRORS**  
 82823 Default.

**APPLICATION USAGE**

If the input files are not properly presorted, the output of *comm* might not be useful.

**EXAMPLES**

If a file named **xcu** contains a sorted list of the utilities in this volume of POSIX.1-2008, a file named **xpg3** contains a sorted list of the utilities specified in the X/Open Portability Guide, Issue 3, and a file named **svid89** contains a sorted list of the utilities in the System V Interface Definition Third Edition:

```
comm -23 xcu xpg3 | comm -23 - svid89
```

would print a list of utilities in this volume of POSIX.1-2008 not specified by either of the other documents:

```
comm -12 xcu xpg3 | comm -12 - svid89
```

would print a list of utilities specified by all three documents, and:

```
comm -12 xpg3 svid89 | comm -23 - xcu
```

would print a list of utilities specified by both XPG3 and the SVID, but not specified in this volume of POSIX.1-2008.

**RATIONALE**

None.

**FUTURE DIRECTIONS**

None.

**SEE ALSO**

*cmp*, *diff*, *sort*, *uniq*

XBD [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 215)

**CHANGE HISTORY**

First released in Issue 2.

**Issue 6**

The normative text is reworded to avoid use of the term “must” for application requirements.

**NAME**

`command` — execute a simple command

**SYNOPSIS**

`command` [`-p`] *command\_name* [*argument...*]

`command` [`-p`] [`-v`|`-V`] *command\_name*

**DESCRIPTION**

The *command* utility shall cause the shell to treat the arguments as a simple command, suppressing the shell function lookup that is described in [Section 2.9.1.1](#) (on page 2339), item 1b.

If the *command\_name* is the same as the name of one of the special built-in utilities, the special properties in the enumerated list at the beginning of [Section 2.14](#) (on page 2356) shall not occur. In every other respect, if *command\_name* is not the name of a function, the effect of *command* (with no options) shall be the same as omitting *command*.

When the `-v` or `-V` option is used, the *command* utility shall provide information concerning how a command name is interpreted by the shell.

**OPTIONS**

The *command* utility shall conform to XBD [Section 12.2](#) (on page 215).

The following options shall be supported:

**-p** Perform the command search using a default value for *PATH* that is guaranteed to find all of the standard utilities.

**-v** Write a string to standard output that indicates the pathname or command that will be used by the shell, in the current shell execution environment (see [Section 2.12](#), on page 2353), to invoke *command\_name*, but do not invoke *command\_name*.

- Utilities, regular built-in utilities, *command\_names* including a `<slash>` character, and any implementation-defined functions that are found using the *PATH* variable (as described in [Section 2.9.1.1](#), on page 2339), shall be written as absolute pathnames.
- Shell functions, special built-in utilities, regular built-in utilities not associated with a *PATH* search, and shell reserved words shall be written as just their names.
- An alias shall be written as a command line that represents its alias definition.
- Otherwise, no output shall be written and the exit status shall reflect that the name was not found.

**-V** Write a string to standard output that indicates how the name given in the *command\_name* operand will be interpreted by the shell, in the current shell execution environment (see [Section 2.12](#), on page 2353), but do not invoke *command\_name*. Although the format of this string is unspecified, it shall indicate in which of the following categories *command\_name* falls and shall include the information stated:

- Utilities, regular built-in utilities, and any implementation-defined functions that are found using the *PATH* variable (as described in [Section 2.9.1.1](#), on page 2339), shall be identified as such and include the absolute pathname in the string.



- 82893 • Other shell functions shall be identified as functions.
- 82894 • Aliases shall be identified as aliases and their definitions included in the
- 82895 string.
- 82896 • Special built-in utilities shall be identified as special built-in utilities.
- 82897 • Regular built-in utilities not associated with a *PATH* search shall be identified
- 82898 as regular built-in utilities. (The term “regular” need not be used.)
- 82899 • Shell reserved words shall be identified as reserved words.

## 82900 OPERANDS

82901 The following operands shall be supported:

82902 *argument* One of the strings treated as an argument to *command\_name*.

82903 *command\_name*

82904 The name of a utility or a special built-in utility.

## 82905 STDIN

82906 Not used.

## 82907 INPUT FILES

82908 None.

## 82909 ENVIRONMENT VARIABLES

82910 The following environment variables shall affect the execution of *command*:

82911 *LANG* Provide a default value for the internationalization variables that are unset or null.  
 82912 (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 82913 variables used to determine the values of locale categories.)

82914 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
 82915 internationalization variables.

82916 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
 82917 characters (for example, single-byte as opposed to multi-byte characters in  
 82918 arguments).

82919 *LC\_MESSAGES*

82920 Determine the locale that should be used to affect the format and contents of  
 82921 diagnostic messages written to standard error and informative messages written to  
 82922 standard output.

82923 *XSI* *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

82924 *PATH* Determine the search path used during the command search described in [Section](#)  
 82925 [2.9.1.1](#) (on page 2339), except as described under the **-p** option.

## 82926 ASYNCHRONOUS EVENTS

82927 Default.

## 82928 STDOUT

82929 When the **-v** option is specified, standard output shall be formatted as:

82930 "%s\n", <pathname or command>

82931 When the **-V** option is specified, standard output shall be formatted as:

82932 "%s\n", <unspecified>

82933 **STDERR**

82934 The standard error shall be used only for diagnostic messages.

82935 **OUTPUT FILES**

82936 None.

82937 **EXTENDED DESCRIPTION**

82938 None.

82939 **EXIT STATUS**82940 When the `-v` or `-V` options are specified, the following exit values shall be returned:

82941 0 Successful completion.

82942 >0 The *command\_name* could not be found or an error occurred.

82943 Otherwise, the following exit values shall be returned:

82944 126 The utility specified by *command\_name* was found but could not be invoked.82945 127 An error occurred in the *command* utility or the utility specified by *command\_name* could not  
82946 be found.82947 Otherwise, the exit status of *command* shall be that of the simple command specified by the  
82948 arguments to *command*.82949 **CONSEQUENCES OF ERRORS**

82950 Default.

82951 **APPLICATION USAGE**82952 The order for command search allows functions to override regular built-ins and path searches.  
82953 This utility is necessary to allow functions that have the same name as a utility to call the utility  
82954 (instead of a recursive call to the function).82955 The system default path is available using *getconf*; however, since *getconf* may need to have the  
82956 *PATH* set up before it can be called itself, the following can be used:82957 `command -p getconf PATH`82958 There are some advantages to suppressing the special characteristics of special built-ins on  
82959 occasion. For example:82960 `command exec > unwritable-file`82961 does not cause a non-interactive script to abort, so that the output status can be checked by the  
82962 script.82963 The *command*, *env*, *nohup*, *time*, and *xargs* utilities have been specified to use exit code 127 if an  
82964 error occurs so that applications can distinguish “failure to find a utility” from “invoked utility  
82965 exited with an error indication”. The value 127 was chosen because it is not commonly used for  
82966 other meanings; most utilities use small values for “normal error conditions” and the values  
82967 above 128 can be confused with termination due to receipt of a signal. The value 126 was chosen  
82968 in a similar manner to indicate that the utility could be found, but not invoked. Some scripts  
82969 produce meaningful error messages differentiating the 126 and 127 cases. The distinction  
82970 between exit codes 126 and 127 is based on KornShell practice that uses 127 when all attempts to  
82971 *exec* the utility fail with [ENOENT], and uses 126 when any attempt to *exec* the utility fails for  
82972 any other reason.82973 Since the `-v` and `-V` options of *command* produce output in relation to the current shell execution  
82974 environment, *command* is generally provided as a shell regular built-in. If it is called in a subshell  
82975 or separate utility execution environment, such as one of the following:

```
82976 (PATH=foo command -v)
82977 nohup command -v
```

82978 it does not necessarily produce correct results. For example, when called with *nohup* or an *exec*  
 82979 function, in a separate utility execution environment, most implementations are not able to  
 82980 identify aliases, functions, or special built-ins.

82981 Two types of regular built-ins could be encountered on a system and these are described  
 82982 separately by *command*. The description of command search in [Section 2.9.1.1](#) (on page 2339)  
 82983 allows for a standard utility to be implemented as a regular built-in as long as it is found in the  
 82984 appropriate place in a *PATH* search. So, for example, *command -v true* might yield */bin/true* or  
 82985 some similar pathname. Other implementation-defined utilities that are not defined by this  
 82986 volume of POSIX.1-2008 might exist only as built-ins and have no pathname associated with  
 82987 them. These produce output identified as (regular) built-ins. Applications encountering these are  
 82988 not able to count on *execing* them, using them with *nohup*, overriding them with a different  
 82989 *PATH*, and so on.

## 82990 EXAMPLES

- 82991 1. Make a version of *cd* that always prints out the new working directory exactly once:

```
82992 cd() {
82993 command cd "$@" >/dev/null
82994 pwd
82995 }
```

- 82996 2. Start off a “secure shell script” in which the script avoids being spoofed by its parent:

```
82997 IFS='
82998 '
82999 # The preceding value should be <space><tab><newline>.
83000 # Set IFS to its default value.
83001 \unalias -a
83002 # Unset all possible aliases.
83003 # Note that unalias is escaped to prevent an alias
83004 # being used for unalias.
83005 unset -f command
83006 # Ensure command is not a user function.
83007 PATH="$(command -p getconf PATH):$PATH"
83008 # Put on a reliable PATH prefix.
83009 # ...
```

83010 At this point, given correct permissions on the directories called by *PATH*, the script has  
 83011 the ability to ensure that any utility it calls is the intended one. It is being very cautious  
 83012 because it assumes that implementation extensions may be present that would allow user  
 83013 functions to exist when it is invoked; this capability is not specified by this volume of  
 83014 POSIX.1-2008, but it is not prohibited as an extension. For example, the *ENV* variable  
 83015 precedes the invocation of the script with a user start-up script. Such a script could define  
 83016 functions to spoof the application.

## 83017 RATIONALE

83018 Since *command* is a regular built-in utility it is always found prior to the *PATH* search.

83019 There is nothing in the description of *command* that implies the command line is parsed any  
 83020 differently from that of any other simple command. For example:

command a | b ; c

is not parsed in any special way that causes ' | ' or ' ; ' to be treated other than a pipe operator or <semicolon> or that prevents function lookup on **b** or **c**.

The *command* utility is somewhat similar to the Eighth Edition shell *builtin* command, but since *command* also goes to the file system to search for utilities, the name *builtin* would not be intuitive.

The *command* utility is most likely to be provided as a regular built-in. It is not listed as a special built-in for the following reasons:

- The removal of exportable functions made the special precedence of a special built-in unnecessary.
- A special built-in has special properties (see [Section 2.14](#), on page 2356) that were inappropriate for invoking other utilities. For example, two commands such as:

```
date > unwritable-file
```

```
command date > unwritable-file
```

would have entirely different results; in a non-interactive script, the former would continue to execute the next command, the latter would abort. Introducing this semantic difference along with suppressing functions was seen to be non-intuitive.

The **-p** option is present because it is useful to be able to ensure a safe path search that finds all the standard utilities. This search might not be identical to the one that occurs through one of the *exec* functions (as defined in the System Interfaces volume of POSIX.1-2008) when *PATH* is unset. At the very least, this feature is required to allow the script to access the correct version of *getconf* so that the value of the default path can be accurately retrieved.

The *command* **-v** and **-V** options were added to satisfy requirements from users that are currently accomplished by three different historical utilities: *type* in the System V shell, *whence* in the KornShell, and *which* in the C shell. Since there is no historical agreement on how and what to accomplish here, the POSIX *command* utility was enhanced and the historical utilities were left unmodified. The C shell *which* merely conducts a path search. The KornShell *whence* is more elaborate—in addition to the categories required by POSIX, it also reports on tracked aliases, exported aliases, and undefined functions.

The output format of **-V** was left mostly unspecified because human users are its only audience. Applications should not be written to care about this information; they can use the output of **-v** to differentiate between various types of commands, but the additional information that may be emitted by the more verbose **-V** is not needed and should not be arbitrarily constrained in its verbosity or localization for application parsing reasons.

## FUTURE DIRECTIONS

None.

## SEE ALSO

[Section 2.9.1.1](#) (on page 2339), [Section 2.12](#) (on page 2353), [Section 2.14](#) (on page 2356), *sh*, *type*

XBD [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 215)

XSH *exec*

83061 **CHANGE HISTORY**

83062 First released in Issue 4.

83063 **Issue 7**83064 Austin Group Interpretation 1003.1-2001 #196 is applied, changing the SYNOPSIS to allow **-p** to  
83065 be used with **-v** (or **-V**).

83066 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

83067 The *command* utility is moved from the User Portability Utilities option to the Base. User  
83068 Portability Utilities is now an option for interactive utilities.83069 The APPLICATION USAGE and EXAMPLES are revised to replace the non-standard  
83070 *getconf\_CS\_PATH* with *getconf PATH*.

## 83071 NAME

83072 compress — compress data

## 83073 SYNOPSIS

83074 XSI `compress [-fv] [-b bits] [file...]`83075 `compress [-cfv] [-b bits] [file]`

## 83076 DESCRIPTION

83077 The *compress* utility shall attempt to reduce the size of the named files by using adaptive Lempel-  
83078 Ziv coding algorithm.83079 **Note:** Lempel-Ziv is US Patent 4464650, issued to William Eastman, Abraham Lempel, Jacob Ziv,  
83080 Martin Cohn on August 7th, 1984, and assigned to Sperry Corporation.83081 Lempel-Ziv-Welch compression is covered by US Patent 4558302, issued to Terry A. Welch on  
83082 December 10th, 1985, and assigned to Sperry Corporation.83083 On systems not supporting adaptive Lempel-Ziv coding algorithm, the input files shall not be  
83084 changed and an error value greater than two shall be returned. Except when the output is to the  
83085 standard output, each file shall be replaced by one with the extension **.Z**. If the invoking process  
83086 has appropriate privileges, the ownership, modes, access time, and modification time of the  
83087 original file are preserved. If appending the **.Z** to the filename would make the name exceed  
83088 {NAME\_MAX} bytes, the command shall fail. If no files are specified, the standard input shall be  
83089 compressed to the standard output.

## 83090 OPTIONS

83091 The *compress* utility shall conform to XBD [Section 12.2](#) (on page 215).

83092 The following options shall be supported:

83093 **-b *bits*** Specify the maximum number of bits to use in a code. For a conforming  
83094 application, the *bits* argument shall be:83095  $9 \leq bits \leq 14$ 83096 The implementation may allow *bits* values of greater than 14. The default is 14, 15,  
83097 or 16.83098 **-c** Cause *compress* to write to the standard output; the input file is not changed, and  
83099 no **.Z** files are created.83100 **-f** Force compression of *file*, even if it does not actually reduce the size of the file, or if  
83101 the corresponding *file.Z* file already exists. If the **-f** option is not given, and the  
83102 process is not running in the background, the user is prompted as to whether an  
83103 existing *file.Z* file should be overwritten. If the response is affirmative, the existing  
83104 file will be overwritten.83105 **-v** Write the percentage reduction of each file to standard error.

## 83106 OPERANDS

83107 The following operand shall be supported:

83108 *file* A pathname of a file to be compressed.

## 83109 STDIN

83110 The standard input shall be used only if no *file* operands are specified, or if a *file* operand is **'-'**.

83111 **INPUT FILES**83112 If *file* operands are specified, the input files contain the data to be compressed.83113 **ENVIRONMENT VARIABLES**83114 The following environment variables shall affect the execution of *compress*:

83115 *LANG* Provide a default value for the internationalization variables that are unset or null.  
 83116 (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 83117 variables used to determine the values of locale categories.)

83118 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
 83119 internationalization variables.

83120 *LC\_COLLATE*

83121 Determine the locale for the behavior of ranges, equivalence classes, and multi-  
 83122 character collating elements used in the extended regular expression defined for  
 83123 the **yesexpr** locale keyword in the *LC\_MESSAGES* category.

83124 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
 83125 characters (for example, single-byte as opposed to multi-byte characters in  
 83126 arguments), the behavior of character classes used in the extended regular  
 83127 expression defined for the **yesexpr** locale keyword in the *LC\_MESSAGES* category.

83128 *LC\_MESSAGES*

83129 Determine the locale used to process affirmative responses, and the locale used to  
 83130 affect the format and contents of diagnostic messages, prompts, and the output  
 83131 from the **-v** option written to standard error.

83132 *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

83133 **ASYNCHRONOUS EVENTS**

83134 Default.

83135 **STDOUT**

83136 If no *file* operands are specified, or if a *file* operand is **'-'**, or if the **-c** option is specified, the  
 83137 standard output contains the compressed output.

83138 **STDERR**

83139 The standard error shall be used only for diagnostic and prompt messages and the output from  
 83140 **-v**.

83141 **OUTPUT FILES**

83142 The output files shall contain the compressed output. The format of compressed files is  
 83143 unspecified and interchange of such files between implementations (including access via  
 83144 unspecified file sharing mechanisms) is not required by POSIX.1-2008.

83145 **EXTENDED DESCRIPTION**

83146 None.

83147 **EXIT STATUS**

83148 The following exit values shall be returned:

83149 0 Successful completion.

83150 1 An error occurred.

83151 2 One or more files were not compressed because they would have increased in size (and the  
 83152 **-f** option was not specified).

83153           >2   An error occurred.

83154   **CONSEQUENCES OF ERRORS**

83155           The input file shall remain unmodified.

83156   **APPLICATION USAGE**

83157           The amount of compression obtained depends on the size of the input, the number of *bits* per  
83158           code, and the distribution of common substrings. Typically, text such as source code or English is  
83159           reduced by 50-60%. Compression is generally much better than that achieved by Huffman  
83160           coding or adaptive Huffman coding (*compact*), and takes less time to compute.

83161           Although *compress* strictly follows the default actions upon receipt of a signal or when an error  
83162           occurs, some unexpected results may occur. In some implementations it is likely that a partially  
83163           compressed file is left in place, alongside its uncompressed input file. Since the general  
83164           operation of *compress* is to delete the uncompressed file only after the *.Z* file has been  
83165           successfully filled, an application should always carefully check the exit status of *compress* before  
83166           arbitrarily deleting files that have like-named neighbors with *.Z* suffixes.

83167           The limit of 14 on the *bits* option-argument is to achieve portability to all systems (within the  
83168           restrictions imposed by the lack of an explicit published file format). Some implementations  
83169           based on 16-bit architectures cannot support 15 or 16-bit uncompression.

83170   **EXAMPLES**

83171           None.

83172   **RATIONALE**

83173           None.

83174   **FUTURE DIRECTIONS**

83175           None.

83176   **SEE ALSO**

83177           *uncompress*, *zcat*

83178           XBD [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 215)

83179   **CHANGE HISTORY**

83180           First released in Issue 4.

83181   **Issue 6**

83182           The normative text is reworded to avoid use of the term “must” for application requirements.

83183           An error case is added for systems not supporting adaptive Lempel-Ziv coding.

83184   **Issue 7**

83185           SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

83186           Austin Group Interpretation 1003.1-2001 #125 is applied, revising the ENVIRONMENT  
83187           VARIABLES section.



83188 **NAME**83189 **cp** — copy files83190 **SYNOPSIS**83191 **cp** [-Pfip] *source\_file* *target\_file*83192 **cp** [-Pfip] *source\_file*... *target*83193 **cp** -R [-H|-L|-P] [-fip] *source\_file*... *target*83194 **DESCRIPTION**

83195 The first synopsis form is denoted by two operands, neither of which are existing files of type  
 83196 directory. The *cp* utility shall copy the contents of *source\_file* (or, if *source\_file* is a file of type  
 83197 symbolic link, the contents of the file referenced by *source\_file*) to the destination path named by  
 83198 *target\_file*.

83199 The second synopsis form is denoted by two or more operands where the **-R** option is not  
 83200 specified and the first synopsis form is not applicable. It shall be an error if any *source\_file* is a file  
 83201 of type directory, if *target* does not exist, or if *target* does not name a directory. The *cp* utility shall  
 83202 copy the contents of each *source\_file* (or, if *source\_file* is a file of type symbolic link, the contents of  
 83203 the file referenced by *source\_file*) to the destination path named by the concatenation of *target*, a  
 83204 single <slash> character if *target* did not end in a <slash>, and the last component of *source\_file*.

83205 The third synopsis form is denoted by two or more operands where the **-R** option is specified.  
 83206 The *cp* utility shall copy each file in the file hierarchy rooted in each *source\_file* to a destination  
 83207 path named as follows:

- 83208 • If *target* exists and names an existing directory, the name of the corresponding destination  
 83209 path for each file in the file hierarchy shall be the concatenation of *target*, a single <slash>  
 83210 character if *target* did not end in a <slash>, and the pathname of the file relative to the  
 83211 directory containing *source\_file*.
- 83212 • If *target* does not exist and two operands are specified, the name of the corresponding  
 83213 destination path for *source\_file* shall be *target*; the name of the corresponding destination  
 83214 path for all other files in the file hierarchy shall be the concatenation of *target*, a <slash>  
 83215 character, and the pathname of the file relative to *source\_file*.

83216 It shall be an error if *target* does not exist and more than two operands are specified, or if *target*  
 83217 exists and does not name a directory.

83218 In the following description, the term *dest\_file* refers to the file named by the destination path.  
 83219 The term *source\_file* refers to the file that is being copied, whether specified as an operand or a  
 83220 file in a file hierarchy rooted in a *source\_file* operand. If *source\_file* is a file of type symbolic link:

- 83221 • If the **-R** option was not specified, *cp* shall take actions based on the type and contents of  
 83222 the file referenced by the symbolic link, and not by the symbolic link itself, unless the **-P**  
 83223 option was specified.
- 83224 • If the **-R** option was specified:
  - 83225 — If none of the options **-H**, **-L**, nor **-P** were specified, it is unspecified which of **-H**,  
 83226 **-L**, or **-P** will be used as a default.
  - 83227 — If the **-H** option was specified, *cp* shall take actions based on the type and contents of  
 83228 the file referenced by any symbolic link specified as a *source\_file* operand.
  - 83229 — If the **-L** option was specified, *cp* shall take actions based on the type and contents of  
 83230 the file referenced by any symbolic link specified as a *source\_file* operand or any  
 83231 symbolic links encountered during traversal of a file hierarchy.

- If the **-P** option was specified, *cp* shall copy any symbolic link specified as a *source\_file* operand and any symbolic links encountered during traversal of a file hierarchy, and shall not follow any symbolic links.

For each *source\_file*, the following steps shall be taken:

1. If *source\_file* references the same file as *dest\_file*, *cp* may write a diagnostic message to standard error; it shall do nothing more with *source\_file* and shall go on to any remaining files.
2. If *source\_file* is of type directory, the following steps shall be taken:
  - a. If the **-R** option was not specified, *cp* shall write a diagnostic message to standard error, do nothing more with *source\_file*, and go on to any remaining files.
  - b. If *source\_file* was not specified as an operand and *source\_file* is dot or dot-dot, *cp* shall do nothing more with *source\_file* and go on to any remaining files.
  - c. If *dest\_file* exists and it is a file type not specified by the System Interfaces volume of POSIX.1-2008, the behavior is implementation-defined.
  - d. If *dest\_file* exists and it is not of type directory, *cp* shall write a diagnostic message to standard error, do nothing more with *source\_file* or any files below *source\_file* in the file hierarchy, and go on to any remaining files.
  - e. If the directory *dest\_file* does not exist, it shall be created with file permission bits set to the same value as those of *source\_file*, modified by the file creation mask of the user if the **-p** option was not specified, and then bitwise-inclusively OR'ed with S\_IRWXU. If *dest\_file* cannot be created, *cp* shall write a diagnostic message to standard error, do nothing more with *source\_file*, and go on to any remaining files. It is unspecified if *cp* attempts to copy files in the file hierarchy rooted in *source\_file*.
  - f. The files in the directory *source\_file* shall be copied to the directory *dest\_file*, taking the four steps (1 to 4) listed here with the files as *source\_files*.
  - g. If *dest\_file* was created, its file permission bits shall be changed (if necessary) to be the same as those of *source\_file*, modified by the file creation mask of the user if the **-p** option was not specified.
  - h. The *cp* utility shall do nothing more with *source\_file* and go on to any remaining files.
3. If *source\_file* is of type regular file, the following steps shall be taken:
  - a. The behavior is unspecified if *dest\_file* exists and was written by a previous step. Otherwise, if *dest\_file* exists, the following steps shall be taken:
    - i. If the **-i** option is in effect, the *cp* utility shall write a prompt to the standard error and read a line from the standard input. If the response is not affirmative, *cp* shall do nothing more with *source\_file* and go on to any remaining files.
    - ii. A file descriptor for *dest\_file* shall be obtained by performing actions equivalent to the *open()* function defined in the System Interfaces volume of POSIX.1-2008 called using *dest\_file* as the *path* argument, and the bitwise-inclusive OR of O\_WRONLY and O\_TRUNC as the *oflag* argument.
    - iii. If the attempt to obtain a file descriptor fails and the **-f** option is in effect, *cp* shall attempt to remove the file by performing actions equivalent to the *unlink()* function defined in the System Interfaces volume of POSIX.1-2008

- 83276 called using *dest\_file* as the *path* argument. If this attempt succeeds, *cp* shall  
83277 continue with step 3b.
- 83278 b. If *dest\_file* does not exist, a file descriptor shall be obtained by performing actions  
83279 equivalent to the *open()* function defined in the System Interfaces volume of  
83280 POSIX.1-2008 called using *dest\_file* as the *path* argument, and the bitwise-inclusive  
83281 OR of *O\_WRONLY* and *O\_CREAT* as the *oflag* argument. The file permission bits  
83282 of *source\_file* shall be the *mode* argument.
- 83283 c. If the attempt to obtain a file descriptor fails, *cp* shall write a diagnostic message to  
83284 standard error, do nothing more with *source\_file*, and go on to any remaining files.
- 83285 d. The contents of *source\_file* shall be written to the file descriptor. Any write errors  
83286 shall cause *cp* to write a diagnostic message to standard error and continue to step  
83287 3e.
- 83288 e. The file descriptor shall be closed.
- 83289 f. The *cp* utility shall do nothing more with *source\_file*. If a write error occurred in  
83290 step 3d, it is unspecified if *cp* continues with any remaining files. If no write error  
83291 occurred in step 3d, *cp* shall go on to any remaining files.
- 83292 4. Otherwise, the **-R** option was specified, and the following steps shall be taken:
- 83293 a. The *dest\_file* shall be created with the same file type as *source\_file*.
- 83294 b. If *source\_file* is a file of type FIFO, the file permission bits shall be the same as those  
83295 of *source\_file*, modified by the file creation mask of the user if the **-p** option was not  
83296 specified. Otherwise, the permissions, owner ID, and group ID of *dest\_file* are  
83297 implementation-defined.
- 83298 If this creation fails for any reason, *cp* shall write a diagnostic message to standard  
83299 error, do nothing more with *source\_file*, and go on to any remaining files.
- 83300 c. If *source\_file* is a file of type symbolic link, and the options require the symbolic link  
83301 itself to be acted upon, the pathname contained in *dest\_file* shall be the same as the  
83302 pathname contained in *source\_file*.
- 83303 If this fails for any reason, *cp* shall write a diagnostic message to standard error, do  
83304 nothing more with *source\_file*, and go on to any remaining files.
- 83305 If the implementation provides additional or alternate access control mechanisms (see XBD  
83306 [Section 4.4](#), on page 108), their effect on copies of files is implementation-defined.

## 83307 OPTIONS

83308 The *cp* utility shall conform to XBD [Section 12.2](#) (on page 215).

83309 The following options shall be supported:

- 83310 **-f** If a file descriptor for a destination file cannot be obtained, as described in step  
83311 3.a.ii., attempt to unlink the destination file and proceed.
- 83312 **-H** Take actions based on the type and contents of the file referenced by any symbolic  
83313 link specified as a *source\_file* operand.
- 83314 **-i** Write a prompt to standard error before copying to any existing non-directory  
83315 destination file. If the response from the standard input is affirmative, the copy  
83316 shall be attempted; otherwise, it shall not.

|       |                              |                                                                                                                                                                                                                                      |
|-------|------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 83317 | <b>-L</b>                    | Take actions based on the type and contents of the file referenced by any symbolic link specified as a <i>source_file</i> operand or any symbolic links encountered during traversal of a file hierarchy.                            |
| 83318 |                              |                                                                                                                                                                                                                                      |
| 83319 |                              |                                                                                                                                                                                                                                      |
| 83320 | <b>-P</b>                    | Take actions on any symbolic link specified as a <i>source_file</i> operand or any symbolic link encountered during traversal of a file hierarchy.                                                                                   |
| 83321 |                              |                                                                                                                                                                                                                                      |
| 83322 | <b>-p</b>                    | Duplicate the following characteristics of each source file in the corresponding destination file:                                                                                                                                   |
| 83323 |                              |                                                                                                                                                                                                                                      |
| 83324 |                              | 1. The time of last data modification and time of last access. If this duplication fails for any reason, <i>cp</i> shall write a diagnostic message to standard error.                                                               |
| 83325 |                              |                                                                                                                                                                                                                                      |
| 83326 |                              | 2. The user ID and group ID. If this duplication fails for any reason, it is unspecified whether <i>cp</i> writes a diagnostic message to standard error.                                                                            |
| 83327 |                              |                                                                                                                                                                                                                                      |
| 83328 |                              | 3. The file permission bits and the S_ISUID and S_ISGID bits. Other, implementation-defined, bits may be duplicated as well. If this duplication fails for any reason, <i>cp</i> shall write a diagnostic message to standard error. |
| 83329 |                              |                                                                                                                                                                                                                                      |
| 83330 |                              |                                                                                                                                                                                                                                      |
| 83331 |                              | If the user ID or the group ID cannot be duplicated, the file permission bits S_ISUID and S_ISGID shall be cleared. If these bits are present in the source file but                                                                 |
| 83332 |                              | are not duplicated in the destination file, it is unspecified whether <i>cp</i> writes a                                                                                                                                             |
| 83333 |                              | diagnostic message to standard error.                                                                                                                                                                                                |
| 83334 |                              |                                                                                                                                                                                                                                      |
| 83335 |                              | The order in which the preceding characteristics are duplicated is unspecified. The                                                                                                                                                  |
| 83336 |                              | <i>dest_file</i> shall not be deleted if these characteristics cannot be preserved.                                                                                                                                                  |
| 83337 | <b>-R</b>                    | Copy file hierarchies.                                                                                                                                                                                                               |
| 83338 |                              | Specifying more than one of the mutually-exclusive options <b>-H</b> , <b>-L</b> , and <b>-P</b> shall not be                                                                                                                        |
| 83339 |                              | considered an error. The last option specified shall determine the behavior of the utility.                                                                                                                                          |
| 83340 | <b>OPERANDS</b>              |                                                                                                                                                                                                                                      |
| 83341 |                              | The following operands shall be supported:                                                                                                                                                                                           |
| 83342 | <i>source_file</i>           | A pathname of a file to be copied. If a <i>source_file</i> operand is '-', it shall refer to a                                                                                                                                       |
| 83343 |                              | file named -; implementations shall not treat it as meaning standard input.                                                                                                                                                          |
| 83344 | <i>target_file</i>           | A pathname of an existing or nonexistent file, used for the output when a single                                                                                                                                                     |
| 83345 |                              | file is copied. If a <i>target_file</i> operand is '-', it shall refer to a file named -;                                                                                                                                            |
| 83346 |                              | implementations shall not treat it as meaning standard output.                                                                                                                                                                       |
| 83347 | <i>target</i>                | A pathname of a directory to contain the copied files.                                                                                                                                                                               |
| 83348 | <b>STDIN</b>                 |                                                                                                                                                                                                                                      |
| 83349 |                              | The standard input shall be used to read an input line in response to each prompt specified in                                                                                                                                       |
| 83350 |                              | the STDERR section. Otherwise, the standard input shall not be used.                                                                                                                                                                 |
| 83351 | <b>INPUT FILES</b>           |                                                                                                                                                                                                                                      |
| 83352 |                              | The input files specified as operands may be of any file type.                                                                                                                                                                       |
| 83353 | <b>ENVIRONMENT VARIABLES</b> |                                                                                                                                                                                                                                      |
| 83354 |                              | The following environment variables shall affect the execution of <i>cp</i> :                                                                                                                                                        |
| 83355 | <i>LANG</i>                  | Provide a default value for the internationalization variables that are unset or null.                                                                                                                                               |
| 83356 |                              | (See XBD <a href="#">Section 8.2</a> (on page 174) for the precedence of internationalization                                                                                                                                        |
| 83357 |                              | variables used to determine the values of locale categories.)                                                                                                                                                                        |

|       |                               |                                                                                                                                                                                                                                                                                                                                                          |
|-------|-------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 83358 | <i>LC_ALL</i>                 | If set to a non-empty string value, override the values of all the other internationalization variables.                                                                                                                                                                                                                                                 |
| 83359 |                               |                                                                                                                                                                                                                                                                                                                                                          |
| 83360 | <i>LC_COLLATE</i>             |                                                                                                                                                                                                                                                                                                                                                          |
| 83361 |                               | Determine the locale for the behavior of ranges, equivalence classes, and multi-character collating elements used in the extended regular expression defined for the <b>yesexpr</b> locale keyword in the <i>LC_MESSAGES</i> category.                                                                                                                   |
| 83362 |                               |                                                                                                                                                                                                                                                                                                                                                          |
| 83363 |                               |                                                                                                                                                                                                                                                                                                                                                          |
| 83364 | <i>LC_CTYPE</i>               | Determine the locale for the interpretation of sequences of bytes of text data as characters (for example, single-byte as opposed to multi-byte characters in arguments and input files) and the behavior of character classes used in the extended regular expression defined for the <b>yesexpr</b> locale keyword in the <i>LC_MESSAGES</i> category. |
| 83365 |                               |                                                                                                                                                                                                                                                                                                                                                          |
| 83366 |                               |                                                                                                                                                                                                                                                                                                                                                          |
| 83367 |                               |                                                                                                                                                                                                                                                                                                                                                          |
| 83368 |                               |                                                                                                                                                                                                                                                                                                                                                          |
| 83369 | <i>LC_MESSAGES</i>            |                                                                                                                                                                                                                                                                                                                                                          |
| 83370 |                               | Determine the locale used to process affirmative responses, and the locale used to affect the format and contents of diagnostic messages and prompts written to standard error.                                                                                                                                                                          |
| 83371 |                               |                                                                                                                                                                                                                                                                                                                                                          |
| 83372 |                               |                                                                                                                                                                                                                                                                                                                                                          |
| 83373 | XSI <i>NLSPATH</i>            | Determine the location of message catalogs for the processing of <i>LC_MESSAGES</i> .                                                                                                                                                                                                                                                                    |
| 83374 | <b>ASYNCHRONOUS EVENTS</b>    |                                                                                                                                                                                                                                                                                                                                                          |
| 83375 |                               | Default.                                                                                                                                                                                                                                                                                                                                                 |
| 83376 | <b>STDOUT</b>                 |                                                                                                                                                                                                                                                                                                                                                          |
| 83377 |                               | Not used.                                                                                                                                                                                                                                                                                                                                                |
| 83378 | <b>STDERR</b>                 |                                                                                                                                                                                                                                                                                                                                                          |
| 83379 |                               | A prompt shall be written to standard error under the conditions specified in the DESCRIPTION section. The prompt shall contain the destination pathname, but its format is otherwise unspecified. Otherwise, the standard error shall be used only for diagnostic messages.                                                                             |
| 83380 |                               |                                                                                                                                                                                                                                                                                                                                                          |
| 83381 |                               |                                                                                                                                                                                                                                                                                                                                                          |
| 83382 | <b>OUTPUT FILES</b>           |                                                                                                                                                                                                                                                                                                                                                          |
| 83383 |                               | The output files may be of any type.                                                                                                                                                                                                                                                                                                                     |
| 83384 | <b>EXTENDED DESCRIPTION</b>   |                                                                                                                                                                                                                                                                                                                                                          |
| 83385 |                               | None.                                                                                                                                                                                                                                                                                                                                                    |
| 83386 | <b>EXIT STATUS</b>            |                                                                                                                                                                                                                                                                                                                                                          |
| 83387 |                               | The following exit values shall be returned:                                                                                                                                                                                                                                                                                                             |
| 83388 |                               | 0 All files were copied successfully.                                                                                                                                                                                                                                                                                                                    |
| 83389 |                               | >0 An error occurred.                                                                                                                                                                                                                                                                                                                                    |
| 83390 | <b>CONSEQUENCES OF ERRORS</b> |                                                                                                                                                                                                                                                                                                                                                          |
| 83391 |                               | If <i>cp</i> is prematurely terminated by a signal or error, files or file hierarchies may be only partially copied and files and directories may have incorrect permissions or access and modification times.                                                                                                                                           |
| 83392 |                               |                                                                                                                                                                                                                                                                                                                                                          |
| 83393 |                               |                                                                                                                                                                                                                                                                                                                                                          |

**APPLICATION USAGE**

The set-user-ID and set-group-ID bits are explicitly cleared when files are created. This is to prevent users from creating programs that are set-user-ID or set-group-ID to them when copying files or to make set-user-ID or set-group-ID files accessible to new groups of users. For example, if a file is set-user-ID and the copy has a different group ID than the source, a new group of users has execute permission to a set-user-ID program than did previously. In particular, this is a problem for superusers copying users' trees.

**EXAMPLES**

None.

**RATIONALE**

The `-i` option exists on BSD systems, giving applications and users a way to avoid accidentally removing files when copying. Although the 4.3 BSD version does not prompt if the standard input is not a terminal, the standard developers decided that use of `-i` is a request for interaction, so when the destination path exists, the utility takes instructions from whatever responds on standard input.

The exact format of the interactive prompts is unspecified. Only the general nature of the contents of prompts are specified because implementations may desire more descriptive prompts than those used on historical implementations. Therefore, an application using the `-i` option relies on the system to provide the most suitable dialog directly with the user, based on the behavior specified.

The `-p` option is historical practice on BSD systems, duplicating the time of last data modification and time of last access. This volume of POSIX.1-2008 extends it to preserve the user and group IDs, as well as the file permissions. This requirement has obvious problems in that the directories are almost certainly modified after being copied. This volume of POSIX.1-2008 requires that the modification times be preserved. The statement that the order in which the characteristics are duplicated is unspecified is to permit implementations to provide the maximum amount of security for the user. Implementations should take into account the obvious security issues involved in setting the owner, group, and mode in the wrong order or creating files with an owner, group, or mode different from the final value.

It is unspecified whether `cp` writes diagnostic messages when the user and group IDs cannot be set due to the widespread practice of users using `-p` to duplicate some portion of the file characteristics, indifferent to the duplication of others. Historic implementations only write diagnostic messages on errors other than [EPERM].

Earlier versions of this standard included support for the `-r` option to copy file hierarchies. The `-r` option is historical practice on BSD and BSD-derived systems. This option is no longer specified by POSIX.1-2008 but may be present in some implementations. The `-R` option was added as a close synonym to the `-r` option, selected for consistency with all other options in this volume of POSIX.1-2008 that do recursive directory descent.

The difference between `-R` and the removed `-r` option is in the treatment by `cp` of file types other than regular and directory. It was implementation-defined how the `-` option treated special files to allow both historical implementations and those that chose to support `-r` with the same abilities as `-R` defined by this volume of POSIX.1-2008. The original `-r` flag, for historic reasons, did not handle special files any differently from regular files, but always read the file and copied its contents. This had obvious problems in the presence of special file types; for example, character devices, FIFOs, and sockets.

When a failure occurs during the copying of a file hierarchy, `cp` is required to attempt to copy files that are on the same level in the hierarchy or above the file where the failure occurred. It is unspecified if `cp` shall attempt to copy files below the file where the failure occurred (which

cannot succeed in any case).

Permissions, owners, and groups of created special file types have been deliberately left as implementation-defined. This is to allow systems to satisfy special requirements (for example, allowing users to create character special devices, but requiring them to be owned by a certain group). In general, it is strongly suggested that the permissions, owner, and group be the same as if the user had run the historical *mknod*, *ln*, or other utility to create the file. It is also probable that additional privileges are required to create block, character, or other implementation-defined special file types.

Additionally, the **-p** option explicitly requires that all set-user-ID and set-group-ID permissions be discarded if any of the owner or group IDs cannot be set. This is to keep users from unintentionally giving away special privilege when copying programs.

When creating regular files, historical versions of *cp* use the mode of the source file as modified by the file mode creation mask. Other choices would have been to use the mode of the source file unmodified by the creation mask or to use the same mode as would be given to a new file created by the user (plus the execution bits of the source file) and then modify it by the file mode creation mask. In the absence of any strong reason to change historic practice, it was in large part retained.

When creating directories, historical versions of *cp* use the mode of the source directory, plus read, write, and search bits for the owner, as modified by the file mode creation mask. This is done so that *cp* can copy trees where the user has read permission, but the owner does not. A side-effect is that if the file creation mask denies the owner permissions, *cp* fails. Also, once the copy is done, historical versions of *cp* set the permissions on the created directory to be the same as the source directory, unmodified by the file creation mask.

This behavior has been modified so that *cp* is always able to create the contents of the directory, regardless of the file creation mask. After the copy is done, the permissions are set to be the same as the source directory, as modified by the file creation mask. This latter change from historical behavior is to prevent users from accidentally creating directories with permissions beyond those they would normally set and for consistency with the behavior of *cp* in creating files.

It is not a requirement that *cp* detect attempts to copy a file to itself; however, implementations are strongly encouraged to do so. Historical implementations have detected the attempt in most cases.

There are two methods of copying subtrees in this volume of POSIX.1-2008. The other method is described as part of the *pax* utility (see *pax*). Both methods are historical practice. The *cp* utility provides a simpler, more intuitive interface, while *pax* offers a finer granularity of control. Each provides additional functionality to the other; in particular, *pax* maintains the hard-link structure of the hierarchy, while *cp* does not. It is the intention of the standard developers that the results be similar (using appropriate option combinations in both utilities). The results are not required to be identical; there seemed insufficient gain to applications to balance the difficulty of implementations having to guarantee that the results would be exactly identical.

The wording allowing *cp* to copy a directory to implementation-defined file types not specified by the System Interfaces volume of POSIX.1-2008 is provided so that implementations supporting symbolic links are not required to prohibit copying directories to symbolic links. Other extensions to the System Interfaces volume of POSIX.1-2008 file types may need to use this loophole as well.

83486 **FUTURE DIRECTIONS**

83487 None.

83488 **SEE ALSO**83489 *mv, find, ln, pax*83490 XBD [Section 4.4](#) (on page 108), [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 215)83491 XSH *open()*, *unlink()*83492 **CHANGE HISTORY**

83493 First released in Issue 2.

83494 **Issue 6**83495 The **-r** option is marked obsolescent.83496 The new options **-H**, **-L**, and **-P** are added to align with the IEEE P1003.2b draft standard. These  
83497 options affect the processing of symbolic links.83498 IEEE PASC Interpretation 1003.2 #194 is applied, adding a description of the **-P** option.83499 IEEE Std 1003.1-2001/Cor 1-2002, item XCU/TC1/D6/18 is applied, correcting an error in the  
83500 SEE ALSO section.83501 **Issue 7**83502 Austin Group Interpretation 1003.1-2001 #126 is applied, changing the description of the  
83503 *LC\_MESSAGES* environment variable.

83504 Austin Group Interpretations 1003.1-2001 #092, #164, #165, and #168 are applied.

83505 SD5-XCU-ERN-31 and SD5-XCU-ERN-42 are applied, updating the DESCRIPTION.

83506 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

83507 SD5-XCU-ERN-102 is applied, clarifying the **-i** option within the OPTIONS section.83508 The obsolescent **-r** option is removed.83509 The **-P** option is added to the SYNOPSIS and to the DESCRIPTION with respect to the **-R**  
83510 option.



83511 **NAME**

83512 crontab — schedule periodic background work

83513 **SYNOPSIS**83514 crontab [*file*]83515 UP crontab [**-e** | **-l** | **-r**]83516 **DESCRIPTION**

83517 UP The *crontab* utility shall create, replace, or edit a user's crontab entry; a crontab entry is a list of  
 83518 commands and the times at which they shall be executed. The new crontab entry can be input by  
 83519 UP specifying *file* or input from standard input if no *file* operand is specified, or by using an editor,  
 83520 if **-e** is specified.

83521 Upon execution of a command from a crontab entry, the implementation shall supply a default  
 83522 environment, defining at least the following environment variables:

83523 *HOME* A pathname of the user's home directory.

83524 *LOGNAME* The user's login name.

83525 *PATH* A string representing a search path guaranteed to find all of the standard utilities.

83526 *SHELL* A pathname of the command interpreter. When *crontab* is invoked as specified by  
 83527 this volume of POSIX.1-2008, the value shall be a pathname for *sh*.

83528 The values of these variables when *crontab* is invoked as specified by this volume of  
 83529 POSIX.1-2008 shall not affect the default values provided when the scheduled command is run.

83530 If standard output and standard error are not redirected by commands executed from the  
 83531 crontab entry, any generated output or errors shall be mailed, via an implementation-defined  
 83532 method, to the user.

83533 XSI Users shall be permitted to use *crontab* if their names appear in the file **cron.allow** which is  
 83534 located in an implementation-defined directory. If that file does not exist, the file **cron.deny**,  
 83535 which is located in an implementation-defined directory, shall be checked to determine whether  
 83536 the user shall be denied access to *crontab*. If neither file exists, only a process with appropriate  
 83537 privileges shall be allowed to submit a job. If only **cron.deny** exists and is empty, global usage  
 83538 shall be permitted. The **cron.allow** and **cron.deny** files shall consist of one user name per line.

83539 **OPTIONS**83540 The *crontab* utility shall conform to XBD [Section 12.2](#) (on page 215).

83541 The following options shall be supported:

83542 UP **-e** Edit a copy of the invoking user's crontab entry, or create an empty entry to edit if  
 83543 the crontab entry does not exist. When editing is complete, the entry shall be  
 83544 installed as the user's crontab entry.

83545 **-l** (The letter ell.) List the invoking user's crontab entry.

83546 **-r** Remove the invoking user's crontab entry.

83547 **OPERANDS**

83548 The following operand shall be supported:

83549 *file* The pathname of a file that contains specifications, in the format defined in the  
 83550 INPUT FILES section, for crontab entries.

83551 **STDIN**

83552 See the INPUT FILES section.

83553 **INPUT FILES**

83554 In the POSIX locale, the user or application shall ensure that a crontab entry is a text file  
 83555 consisting of lines of six fields each. The fields shall be separated by <blank> characters. The  
 83556 first five fields shall be integer patterns that specify the following:

- 83557 1. Minute [0,59]
- 83558 2. Hour [0,23]
- 83559 3. Day of the month [1,31]
- 83560 4. Month of the year [1,12]
- 83561 5. Day of the week ([0,6] with 0=Sunday)

83562 Each of these patterns can be either an <asterisk> (meaning all valid values), an element, or a list  
 83563 of elements separated by <comma> characters. An element shall be either a number or two  
 83564 numbers separated by a <hyphen> (meaning an inclusive range). The specification of days can  
 83565 be made by two fields (day of the month and day of the week). If month, day of month, and day  
 83566 of week are all <asterisk> characters, every day shall be matched. If either the month or day of  
 83567 month is specified as an element or list, but the day of week is an <asterisk>, the month and day  
 83568 of month fields shall specify the days that match. If both month and day of month are specified  
 83569 as an <asterisk>, but day of week is an element or list, then only the specified days of the week  
 83570 match. Finally, if either the month or day of month is specified as an element or list, and the day  
 83571 of week is also specified as an element or list, then any day matching either the month and day  
 83572 of month, or the day of week, shall be matched.

83573 The sixth field of a line in a crontab entry is a string that shall be executed by *sh* at the specified  
 83574 times. A <percent-sign> character in this field shall be translated to a <newline>. Any character  
 83575 preceded by a <backslash> (including the ' % ' ) shall cause that character to be treated literally.  
 83576 Only the first line (up to a ' % ' or end-of-line) of the command field shall be executed by the  
 83577 command interpreter. The other lines shall be made available to the command as standard input.

83578 Blank lines and those whose first non-<blank> is ' # ' shall be ignored.

83579 XSI The text files **cron.allow** and **cron.deny**, which are located in an implementation-defined  
 83580 directory, shall contain zero or more user names, one per line, of users who are, respectively,  
 83581 authorized or denied access to the service underlying the *crontab* utility.

83582 **ENVIRONMENT VARIABLES**

83583 The following environment variables shall affect the execution of *crontab*:

- |       |                 |                                                                                                                                                                                                                                                    |
|-------|-----------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 83584 | <i>EDITOR</i>   | Determine the editor to be invoked when the <b>-e</b> option is specified. The default editor shall be <i>vi</i> .                                                                                                                                 |
| 83585 |                 |                                                                                                                                                                                                                                                    |
| 83586 | <i>LANG</i>     | Provide a default value for the internationalization variables that are unset or null. (See XBD <a href="#">Section 8.2</a> (on page 174) for the precedence of internationalization variables used to determine the values of locale categories.) |
| 83587 |                 |                                                                                                                                                                                                                                                    |
| 83588 |                 |                                                                                                                                                                                                                                                    |
| 83589 | <i>LC_ALL</i>   | If set to a non-empty string value, override the values of all the other internationalization variables.                                                                                                                                           |
| 83590 |                 |                                                                                                                                                                                                                                                    |
| 83591 | <i>LC_CTYPE</i> | Determine the locale for the interpretation of sequences of bytes of text data as characters (for example, single-byte as opposed to multi-byte characters in arguments and input files).                                                          |
| 83592 |                 |                                                                                                                                                                                                                                                    |
| 83593 |                 |                                                                                                                                                                                                                                                    |

83594 **LC\_MESSAGES**  
 83595 Determine the locale that should be used to affect the format and contents of  
 83596 diagnostic messages written to standard error.

83597 XSI **NLSPATH** Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

83598 **ASYNCHRONOUS EVENTS**  
 83599 Default.

83600 **STDOUT**  
 83601 If the `-l` option is specified, the crontab entry shall be written to the standard output.

83602 **STDERR**  
 83603 The standard error shall be used only for diagnostic messages.

83604 **OUTPUT FILES**  
 83605 None.

83606 **EXTENDED DESCRIPTION**  
 83607 None.

83608 **EXIT STATUS**  
 83609 The following exit values shall be returned:  
 83610 0 Successful completion.  
 83611 >0 An error occurred.

83612 **CONSEQUENCES OF ERRORS**  
 83613 UP The user's crontab entry is not submitted, removed, **edited**, or listed.

83614 **APPLICATION USAGE**  
 83615 The format of the crontab entry shown here is guaranteed only for the POSIX locale. Other  
 83616 cultures may be supported with substantially different interfaces, although implementations are  
 83617 encouraged to provide comparable levels of functionality.

83618 The default settings of the *HOME*, *LOGNAME*, *PATH*, and *SHELL* variables that are given to the  
 83619 scheduled job are not affected by the settings of those variables when *crontab* is run; as stated,  
 83620 they are defaults. The text about "invoked as specified by this volume of POSIX.1-2008" means  
 83621 that the implementation may provide extensions that allow these variables to be affected at  
 83622 runtime, but that the user has to take explicit action in order to access the extension, such as give  
 83623 a new option flag or modify the format of the crontab entry.

83624 A typical user error is to type only *crontab*; this causes the system to wait for the new crontab  
 83625 entry on standard input. If end-of-file is typed (generally <control>-D), the crontab entry is  
 83626 replaced by an empty file. In this case, the user should type the interrupt character, which  
 83627 prevents the crontab entry from being replaced.

83628 **EXAMPLES**  
 83629 1. Clean up **core** files every weekday morning at 3:15 am:  
 83630 15 3 \* \* 1-5 find "\$HOME" -name core -exec rm -f {} + 2>/dev/null  
 83631 2. Mail a birthday greeting:  
 83632 0 12 14 2 \* mailx john%Happy Birthday!%Time for lunch.  
 83633 3. As an example of specifying the two types of days:  
 83634 0 0 1,15 \* 1

83635 would run a command on the first and fifteenth of each month, as well as on every  
83636 Monday. To specify days by only one field, the other field should be set to `'*'`; for  
83637 example:

83638 0 0 \* \* 1

83639 would run a command only on Mondays.

#### 83640 RATIONALE

83641 All references to a *cron* daemon and to *cron files* have been omitted. Although historical  
83642 implementations have used this arrangement, there is no reason to limit future implementations.

83643 This description of *crontab* is designed to support only users with normal privileges. The format  
83644 of the input is based on the System V *crontab*; however, there is no requirement here that the  
83645 actual system database used by the *cron* daemon (or a similar mechanism) use this format  
83646 internally. For example, systems derived from BSD are likely to have an additional field  
83647 appended that indicates the user identity to be used when the job is submitted.

83648 The `-e` option was adopted from the SVID as a user convenience, although it does not exist in all  
83649 historical implementations.

#### 83650 FUTURE DIRECTIONS

83651 None.

#### 83652 SEE ALSO

83653 *at*

83654 XBD [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 215)

#### 83655 CHANGE HISTORY

83656 First released in Issue 2.

#### 83657 Issue 6

83658 This utility is marked as part of the User Portability Utilities option.

83659 The normative text is reworded to avoid use of the term “must” for application requirements.

#### 83660 Issue 7

83661 The *crontab* utility (except for the `-e` option) is moved from the User Portability Utilities option  
83662 to the Base. User Portability Utilities is now an option for interactive utilities.

83663 SD5-XCU-ERN-95 is applied, removing the references to fixed locations for the files referenced  
83664 by the *crontab* utility.

83665 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

83666 The first example is changed to remove the unreliable use of *find* | *xargs*.

83667 **NAME**

83668       csplit — split files based on context

83669 **SYNOPSIS**83670       csplit [-ks] [-f *prefix*] [-n *number*] *file arg...*83671 **DESCRIPTION**83672       The *csplit* utility shall read the file named by the *file* operand, write all or part of that file into  
83673       other files as directed by the *arg* operands, and write the sizes of the files.83674 **OPTIONS**83675       The *csplit* utility shall conform to XBD [Section 12.2](#) (on page 215).

83676       The following options shall be supported:

83677       **-f *prefix***     Name the created files *prefix00*, *prefix01*, ..., *prefixn*. The default is *xx00* ... *xxn*. If  
83678       the *prefix* argument would create a filename exceeding {NAME\_MAX} bytes, an  
83679       error shall result, *csplit* shall exit with a diagnostic message, and no files shall be  
83680       created.83681       **-k**            Leave previously created files intact. By default, *csplit* shall remove created files if  
83682       an error occurs.83683       **-n *number***    Use *number* decimal digits to form filenames for the file pieces. The default shall be  
83684       2.83685       **-s**            Suppress the output of file size messages.83686 **OPERANDS**

83687       The following operands shall be supported:

83688       *file*           The pathname of a text file to be split. If *file* is '-', the standard input shall be  
83689       used.83690       Each *arg* operand can be one of the following:83691       /*regexp*/[*offset*]83692       A file shall be created using the content of the lines from the current line up to, but  
83693       not including, the line that results from the evaluation of the regular expression  
83694       with *offset*, if any, applied. The regular expression *regexp* shall follow the rules for  
83695       basic regular expressions described in XBD [Section 9.3](#) (on page 183). The  
83696       application shall use the sequence "\/" to specify a <slash> character within the  
83697       *regexp*. The optional offset shall be a positive or negative integer value representing a  
83698       number of lines. A positive integer value can be preceded by '+'. If the selection  
83699       of lines from an *offset* expression of this type would create a file with zero lines, or  
83700       one with greater than the number of lines left in the input file, the results are  
83701       unspecified. After the section is created, the current line shall be set to the line that  
83702       results from the evaluation of the regular expression with any offset applied. If the  
83703       current line is the first line in the file and a regular expression operation has not yet  
83704       been performed, the pattern match of *regexp* shall be applied from the current line to  
83705       the end of the file. Otherwise, the pattern match of *regexp* shall be applied from the  
83706       line following the current line to the end of the file.83707       %*regexp*%[*offset*]83708       Equivalent to /*regexp*/[*offset*], except that no file shall be created for the selected  
83709       section of the input file. The application shall use the sequence "\%" to specify a  
83710       <percent-sign> character within the *regexp*.

83711 *line\_no* Create a file from the current line up to (but not including) the line number *line\_no*.  
 83712 Lines in the file shall be numbered starting at one. The current line becomes  
 83713 *line\_no*.

83714 {*num*} Repeat operand. This operand can follow any of the operands described  
 83715 previously. If it follows a *rexp* type operand, that operand shall be applied *num*  
 83716 more times. If it follows a *line\_no* operand, the file shall be split every *line\_no* lines,  
 83717 *num* times, from that point.

83718 An error shall be reported if an operand does not reference a line between the current position  
 83719 and the end of the file.

83720 **STDIN**

83721 See the INPUT FILES section.

83722 **INPUT FILES**

83723 The input file shall be a text file.

83724 **ENVIRONMENT VARIABLES**

83725 The following environment variables shall affect the execution of *csplit*:

83726 *LANG* Provide a default value for the internationalization variables that are unset or null.  
 83727 (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 83728 variables used to determine the values of locale categories.)

83729 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
 83730 internationalization variables.

83731 *LC\_COLLATE*  
 83732 Determine the locale for the behavior of ranges, equivalence classes, and multi-  
 83733 character collating elements within regular expressions.

83734 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
 83735 characters (for example, single-byte as opposed to multi-byte characters in  
 83736 arguments and input files) and the behavior of character classes within regular  
 83737 expressions.

83738 *LC\_MESSAGES*  
 83739 Determine the locale that should be used to affect the format and contents of  
 83740 diagnostic messages written to standard error.

83741 XSI *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

83742 **ASYNCHRONOUS EVENTS**

83743 If the *-k* option is specified, created files shall be retained. Otherwise, the default action occurs.

83744 **STDOUT**

83745 Unless the *-s* option is used, the standard output shall consist of one line per file created, with a  
 83746 format as follows:

83747 "%d\n", <*file size in bytes*>

83748 **STDERR**

83749 The standard error shall be used only for diagnostic messages.

83750 **OUTPUT FILES**

83751 The output files shall contain portions of the original input file; otherwise, unchanged.

83752 **EXTENDED DESCRIPTION**

83753 None.

83754 **EXIT STATUS**

83755 The following exit values shall be returned:

83756 0 Successful completion.

83757 &gt;0 An error occurred.

83758 **CONSEQUENCES OF ERRORS**

83759 By default, created files shall be removed if an error occurs. When the **-k** option is specified,  
 83760 created files shall not be removed if an error occurs.

83761 **APPLICATION USAGE**

83762 None.

83763 **EXAMPLES**83764 1. This example creates four files, **cobol00** ... **cobol03**:83765 `csplit -f cobol file '/procedure division/' /par5./ /par16./`

83766 After editing the split files, they can be recombined as follows:

83767 `cat cobol0[0-3] > file`

83768 Note that this example overwrites the original file.

83769 2. This example would split the file after the first 99 lines, and every 100 lines thereafter, up  
 83770 to 9999 lines; this is because lines in the file are numbered from 1 rather than zero, for  
 83771 historical reasons:

83772 `csplit -k file 100 {99}`

83773 3. Assuming that **prog.c** follows the C-language coding convention of ending routines with  
 83774 a `' } '` at the beginning of the line, this example creates a file containing each separate C  
 83775 routine (up to 21) in **prog.c**:

83776 `csplit -k prog.c '%main(%' '/^}'+1' {20}`83777 **RATIONALE**83778 The **-n** option was added to extend the range of filenames that could be handled.

83779 Consideration was given to adding a **-a** flag to use the alphabetic filename generation used by  
 83780 the historical *split* utility, but the functionality added by the **-n** option was deemed to make  
 83781 alphabetic naming unnecessary.

83782 **FUTURE DIRECTIONS**

83783 None.

83784 **SEE ALSO**83785 *sed*, *split*83786 XBD [Chapter 8](#) (on page 173), [Section 9.3](#) (on page 183), [Section 12.2](#) (on page 215)83787 **CHANGE HISTORY**

83788 First released in Issue 2.

**83789 Issue 5**

83790 The FUTURE DIRECTIONS section is added.

**83791 Issue 6**

83792 This utility is marked as part of the User Portability Utilities option.

83793 The APPLICATION USAGE section is added.

83794 The description of regular expression operands is changed to align with the IEEE P1003.2b draft  
83795 standard.

83796 The normative text is reworded to avoid use of the term “must” for application requirements.

**83797 Issue 7**

83798 The *csplit* utility is moved from the User Portability Utilities option to the Base. User Portability  
83799 Utilities is now an option for interactive utilities.

83800 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

83801 The SYNOPSIS and OPERANDS sections are revised to use a single *arg* to split a file into two  
83802 pieces.



83803 **NAME**83804 `ctags` — create a tags file (**DEVELOPMENT**, **FORTTRAN**)83805 **SYNOPSIS**83806 SD `ctags [-a] [-f tagsfile] pathname...`83807 `ctags -x pathname...`83808 **DESCRIPTION**

83809 The *ctags* utility shall be provided on systems that support the the Software Development  
 83810 Utilities option, and either or both of the C-Language Development Utilities option and  
 83811 FORTRAN Development Utilities option. On other systems, it is optional.

83812 The *ctags* utility shall write a *tagsfile* or an index of objects from C-language or FORTRAN source  
 83813 files specified by the *pathname* operands. The *tagsfile* shall list the locators of language-specific  
 83814 objects within the source files. A locator consists of a name, *pathname*, and either a search  
 83815 pattern or a line number that can be used in searching for the object definition. The objects that  
 83816 shall be recognized are specified in the EXTENDED DESCRIPTION section.

83817 **OPTIONS**83818 The *ctags* utility shall conform to XBD [Section 12.2](#) (on page 215).

83819 The following options shall be supported:

83820 **-a** Append to *tagsfile*.

83821 **-f** *tagsfile* Write the object locator lists into *tagsfile* instead of the default file named **tags** in the  
 83822 current directory.

83823 **-x** Produce a list of object names, the line number, and filename in which each is  
 83824 defined, as well as the text of that line, and write this to the standard output. A  
 83825 *tagsfile* shall not be created when **-x** is specified.

83826 **OPERANDS**83827 The following *pathname* operands are supported:

83828 *file.c* Files with basenames ending with the **.c** suffix shall be treated as C-language  
 83829 source code. Such files that are not valid input to *c99* produce unspecified results.

83830 *file.h* Files with basenames ending with the **.h** suffix shall be treated as C-language  
 83831 source code. Such files that are not valid input to *c99* produce unspecified results.

83832 *file.f* Files with basenames ending with the **.f** suffix shall be treated as FORTRAN-  
 83833 language source code. Such files that are not valid input to *fort77* produce  
 83834 unspecified results.

83835 The handling of other files is implementation-defined.

83836 **STDIN**

83837 See the INPUT FILES section.

83838 **INPUT FILES**

83839 The input files shall be text files containing source code in the language indicated by the  
 83840 operand filename suffixes.

83841 **ENVIRONMENT VARIABLES**83842 The following environment variables shall affect the execution of *ctags*:

83843 *LANG* Provide a default value for the internationalization variables that are unset or null.  
 83844 (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 83845 variables used to determine the values of locale categories.)

83846 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
 83847 internationalization variables.

83848 *LC\_COLLATE*

83849 Determine the order in which output is sorted for the *-x* option. The POSIX locale  
 83850 determines the order in which the *tagsfile* is written.

83851 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
 83852 characters (for example, single-byte as opposed to multi-byte characters in  
 83853 arguments and input files). When processing C-language source code, if the locale  
 83854 is not compatible with the C locale described by the ISO C standard, the results are  
 83855 unspecified.

83856 *LC\_MESSAGES*

83857 Determine the locale that should be used to affect the format and contents of  
 83858 diagnostic messages written to standard error.

83859 XSI *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

83860 **ASYNCHRONOUS EVENTS**

83861 Default.

83862 **STDOUT**

83863 The list of object name information produced by the *-x* option shall be written to standard  
 83864 output in the following format:

83865 "%s %d %s %s", <object-name>, <line-number>, <filename>, <text>

83866 where <text> is the text of line <line-number> of file <filename>.

83867 **STDERR**

83868 The standard error shall be used only for diagnostic messages.

83869 **OUTPUT FILES**83870 When the *-x* option is not specified, the format of the output file shall be:

83871 "%s\t%s\t/%s/\n", <identifier>, <filename>, <pattern>

83872 where <pattern> is a search pattern that could be used by an editor to find the defining instance  
 83873 of <identifier> in <filename> (where *defining instance* is indicated by the declarations listed in the  
 83874 EXTENDED DESCRIPTION).

83875 An optional <circumflex> ('*^*') can be added as a prefix to <pattern>, and an optional <dollar-  
 83876 sign> can be appended to <pattern> to indicate that the pattern is anchored to the beginning  
 83877 (end) of a line of text. Any <slash> or <backslash> characters in <pattern> shall be preceded by a  
 83878 <backslash> character. The anchoring <circumflex>, <dollar-sign>, and escaping <backslash>  
 83879 characters shall not be considered part of the search pattern. All other characters in the search  
 83880 pattern shall be considered literal characters.

An alternative format is:

```
"%s\t%s\t%s?\n", <identifier>, <filename>, <pattern>
```

which is identical to the first format except that <slash> characters in <pattern> shall not be preceded by escaping <backslash> characters, and <question-mark> characters in <pattern> shall be preceded by <backslash> characters.

A second alternative format is:

```
"%s\t%s\t%d\n", <identifier>, <filename>, <lineno>
```

where <lineno> is a decimal line number that could be used by an editor to find <identifier> in <filename>.

Neither alternative format shall be produced by *ctags* when it is used as described by POSIX.1-2008, but the standard utilities that process tags files shall be able to process those formats as well as the first format.

In any of these formats, the file shall be sorted by identifier, based on the collation sequence in the POSIX locale.

## EXTENDED DESCRIPTION

If the operand identifies C-language source, the *ctags* utility shall attempt to produce an output line for each of the following objects:

- Function definitions
- Type definitions
- Macros with arguments

It may also produce output for any of the following objects:

- Function prototypes
- Structures
- Unions
- Global variable definitions
- Enumeration types
- Macros without arguments
- **#define** statements
- **#line** statements

Any **#if** and **#ifdef** statements shall produce no output. The tag **main** is treated specially in C programs. The tag formed shall be created by prefixing **M** to the name of the file, with the trailing **.c**, and leading pathname components (if any) removed.

On systems that do not support the C-Language Development Utilities option, *ctags* produces unspecified results for C-language source code files. It should write to standard error a message identifying this condition and cause a non-zero exit status to be produced.

If the operand identifies FORTRAN source, the *ctags* utility shall produce an output line for each function definition. It may also produce output for any of the following objects:

- Subroutine definitions

- 83919 • COMMON statements
- 83920 • PARAMETER statements
- 83921 • DATA and BLOCK DATA statements
- 83922 • Statement numbers

83923 On systems that do not support the FORTRAN Development Utilities option, *ctags* produces  
 83924 unspecified results for FORTRAN source code files. It should write to standard error a message  
 83925 identifying this condition and cause a non-zero exit status to be produced.

83926 It is implementation-defined what other objects (including duplicate identifiers) produce output.

## 83927 EXIT STATUS

83928 The following exit values shall be returned:

- 83929 0 Successful completion.
- 83930 >0 An error occurred.

## 83931 CONSEQUENCES OF ERRORS

83932 Default.

## 83933 APPLICATION USAGE

83934 The output with *-x* is meant to be a simple index that can be written out as an off-line readable  
 83935 function index. If the input files to *ctags* (such as *.c* files) were not created using the same locale  
 83936 as that in effect when *ctags -x* is run, results might not be as expected.

83937 The description of C-language processing says “attempts to” because the C language can be  
 83938 greatly confused, especially through the use of *#defines*, and this utility would be of no use if  
 83939 the real C preprocessor were run to identify them. The output from *ctags* may be fooled and  
 83940 incorrect for various constructs.

## 83941 EXAMPLES

83942 None.

## 83943 RATIONALE

83944 The option list was significantly reduced from that provided by historical implementations. The  
 83945 *-F* option was omitted as redundant, since it is the default. The *-B* option was omitted as being  
 83946 of very limited usefulness. The *-t* option was omitted since the recognition of *typedefs* is now  
 83947 required for C source files. The *-u* option was omitted because the update function was judged  
 83948 to be not only inefficient, but also rarely needed.

83949 An early proposal included a *-w* option to suppress warning diagnostics. Since the types of such  
 83950 diagnostics could not be described, the option was omitted as being not useful.

83951 The text for *LC\_CTYPE* about compatibility with the C locale acknowledges that the ISO C  
 83952 standard imposes requirements on the locale used to process C source. This could easily be a  
 83953 superset of that known as “the C locale” by way of implementation extensions, or one of a few  
 83954 alternative locales for systems supporting different codesets. No statement is made for  
 83955 FORTRAN because the ANSI X3.9-1978 standard (FORTRAN 77) does not (yet) define a similar  
 83956 locale concept. However, a general rule in this volume of POSIX.1-2008 is that any time that  
 83957 locales do not match (preparing a file for one locale and processing it in another), the results are  
 83958 suspect.

83959 The collation sequence of the tags file is not affected by *LC\_COLLATE* because it is typically not  
 83960 used by human readers, but only by programs such as *vi* to locate the tag within the source files.  
 83961 Using the POSIX locale eliminates some of the problems of coordinating locales between the  
 83962 *ctags* file creator and the *vi* file reader.

Historically, the tags file has been used only by *ex* and *vi*. However, the format of the tags file has been published to encourage other programs to use the tags in new ways. The format allows either patterns or line numbers to find the identifiers because the historical *vi* recognizes either. The *ctags* utility does not produce the format using line numbers because it is not useful following any source file changes that add or delete lines. The documented search patterns match historical practice. It should be noted that literal leading `<circumflex>` or trailing `<dollar-sign>` characters in the search pattern will only behave correctly if anchored to the beginning of the line or end of the line by an additional `<circumflex>` or `<dollar-sign>` character.

Historical implementations also understand the objects used by the languages Pascal and sometimes LISP, and they understand the C source output by *lex* and *yacc*. The *ctags* utility is not required to accommodate these languages, although implementors are encouraged to do so.

The following historical option was not specified, as *vgrind* is not included in this volume of POSIX.1-2008:

**-v** If the **-v** flag is given, an index of the form expected by *vgrind* is produced on the standard output. This listing contains the function name, filename, and page number (assuming 64-line pages). Since the output is sorted into lexicographic order, it may be desired to run the output through *sort -f*. Sample use:

```
ctags -v files | sort -f > index vgrind -x index
```

The special treatment of the tag **main** makes the use of *ctags* practical in directories with more than one program.

## FUTURE DIRECTIONS

None.

## SEE ALSO

[c99](#), [fort77](#), [vi](#)

XBD [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 215)

## CHANGE HISTORY

First released in Issue 4.

### Issue 5

The FUTURE DIRECTIONS section is added.

### Issue 6

This utility is marked as part of the User Portability Utilities option.

The OUTPUT FILES section is changed to align with the IEEE P1003.2b draft standard.

The normative text is reworded to avoid use of the term “must” for application requirements.

IEEE PASC Interpretation 1003.2 #168 is applied, changing “create” to “write” in the DESCRIPTION.

### Issue 7

The *ctags* utility is no longer dependent on support for the User Portability Utilities option.

SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

84001 **NAME**

84002       cut — cut out selected fields of each line of a file

84003 **SYNOPSIS**84004       cut -b *list* [-n] [*file...*]84005       cut -c *list* [*file...*]84006       cut -f *list* [-d *delim*] [-s] [*file...*]84007 **DESCRIPTION**

84008       The *cut* utility shall cut out bytes (**-b** option), characters (**-c** option), or character-delimited fields  
 84009       (**-f** option) from each line in one or more files, concatenate them, and write them to standard  
 84010       output.

84011 **OPTIONS**84012       The *cut* utility shall conform to XBD [Section 12.2](#) (on page 215).

84013       The application shall ensure that the option-argument *list* (see options **-b**, **-c**, and **-f** below) is a  
 84014       <comma>-separated list or <blank>-separated list of positive numbers and ranges. Ranges can  
 84015       be in three forms. The first is two positive numbers separated by a <hyphen> (*low-high*), which  
 84016       represents all fields from the first number to the second number. The second is a positive  
 84017       number preceded by a <hyphen> (*-high*), which represents all fields from field number 1 to that  
 84018       number. The third is a positive number followed by a <hyphen> (*low-*), which represents that  
 84019       number to the last field, inclusive. The elements in *list* can be repeated, can overlap, and can be  
 84020       specified in any order, but the bytes, characters, or fields selected shall be written in the order of  
 84021       the input data. If an element appears in the selection list more than once, it shall be written  
 84022       exactly once.

84023       The following options shall be supported:

84024       **-b list**       Cut based on a *list* of bytes. Each selected byte shall be output unless the **-n** option  
 84025       is also specified. It shall not be an error to select bytes not present in the input line.

84026       **-c list**       Cut based on a *list* of characters. Each selected character shall be output. It shall  
 84027       not be an error to select characters not present in the input line.

84028       **-d delim**      Set the field delimiter to the character *delim*. The default is the <tab>.

84029       **-f list**       Cut based on a *list* of fields, assumed to be separated in the file by a delimiter  
 84030       character (see **-d**). Each selected field shall be output. Output fields shall be  
 84031       separated by a single occurrence of the field delimiter character. Lines with no field  
 84032       delimiters shall be passed through intact, unless **-s** is specified. It shall not be an  
 84033       error to select fields not present in the input line.

84034       **-n**           Do not split characters. When specified with the **-b** option, each element in *list* of  
 84035       the form *low-high* (<hyphen>-separated numbers) shall be modified as follows:

- 84036           • If the byte selected by *low* is not the first byte of a character, *low* shall be  
 84037           decremented to select the first byte of the character originally selected by *low*.  
 84038           If the byte selected by *high* is not the last byte of a character, *high* shall be  
 84039           decremented to select the last byte of the character prior to the character  
 84040           originally selected by *high*, or zero if there is no prior character. If the  
 84041           resulting range element has *high* equal to zero or *low* greater than *high*, the list  
 84042           element shall be dropped from *list* for that input line without causing an  
 84043           error.

84044       Each element in *list* of the form *low-* shall be treated as above with *high* set to the  
 84045       number of bytes in the current line, not including the terminating <newline>. Each

84046 element in *list* of the form *-high* shall be treated as above with *low* set to 1. Each  
 84047 element in *list* of the form *num* (a single number) shall be treated as above with *low*  
 84048 set to *num* and *high* set to *num*.

84049 **-s** Suppress lines with no delimiter characters, when used with the *-f* option. Unless  
 84050 specified, lines with no delimiters shall be passed through untouched.

## 84051 OPERANDS

84052 The following operand shall be supported:

84053 *file* A pathname of an input file. If no *file* operands are specified, or if a *file* operand is  
 84054 '*-*', the standard input shall be used.

## 84055 STDIN

84056 The standard input shall be used only if no *file* operands are specified, or if a *file* operand is '*-*'.  
 84057 See the INPUT FILES section.

## 84058 INPUT FILES

84059 The input files shall be text files, except that line lengths shall be unlimited.

## 84060 ENVIRONMENT VARIABLES

84061 The following environment variables shall affect the execution of *cut*:

84062 *LANG* Provide a default value for the internationalization variables that are unset or null.  
 84063 (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 84064 variables used to determine the values of locale categories.)

84065 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
 84066 internationalization variables.

84067 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
 84068 characters (for example, single-byte as opposed to multi-byte characters in  
 84069 arguments and input files).

84070 *LC\_MESSAGES*

84071 Determine the locale that should be used to affect the format and contents of  
 84072 diagnostic messages written to standard error.

84073 XSI *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

## 84074 ASYNCHRONOUS EVENTS

84075 Default.

## 84076 STDOUT

84077 The *cut* utility output shall be a concatenation of the selected bytes, characters, or fields (one of  
 84078 the following):

84079 "%s\n", <concatenation of bytes>

84080 "%s\n", <concatenation of characters>

84081 "%s\n", <concatenation of fields and field delimiters>

## 84082 STDERR

84083 The standard error shall be used only for diagnostic messages.

## 84084 OUTPUT FILES

84085 None.

## 84086 EXTENDED DESCRIPTION

84087 None.

## 84088 EXIT STATUS

84089 The following exit values shall be returned:

84090 0 All input files were output successfully.

84091 &gt;0 An error occurred.

## 84092 CONSEQUENCES OF ERRORS

84093 Default.

## 84094 APPLICATION USAGE

84095 The *cut* and *fold* utilities can be used to create text files out of files with arbitrary line lengths.  
 84096 The *cut* utility should be used when the number of lines (or records) needs to remain constant.  
 84097 The *fold* utility should be used when the contents of long lines need to be kept contiguous.

84098 Earlier versions of the *cut* utility worked in an environment where bytes and characters were  
 84099 considered equivalent (modulo <backspace> and <tab> processing in some implementations). In  
 84100 the extended world of multi-byte characters, the new **-b** option has been added. The **-n** option  
 84101 (used with **-b**) allows it to be used to act on bytes rounded to character boundaries. The  
 84102 algorithm specified for **-n** guarantees that:

84103 `cut -b 1-500 -n file > file1`84104 `cut -b 501- -n file > file2`

84105 ends up with all the characters in **file** appearing exactly once in **file1** or **file2**. (There is,  
 84106 however, a <newline> in both **file1** and **file2** for each <newline> in **file**.)

## 84107 EXAMPLES

84108 Examples of the option qualifier list:

84109 1,4,7 Select the first, fourth, and seventh bytes, characters, or fields and field delimiters.

84110 1-3,8 Equivalent to 1,2,3,8.

84111 -5,10 Equivalent to 1,2,3,4,5,10.

84112 3- Equivalent to third to last, inclusive.

84113 The *low-high* forms are not always equivalent when used with **-b** and **-n** and multi-byte  
 84114 characters; see the description of **-n**.

84115 The following command:

84116 `cut -d : -f 1,6 /etc/passwd`

84117 reads the System V password file (user database) and produces lines of the form:

84118 `<user ID>:<home directory>`

84119 Most utilities in this volume of POSIX.1-2008 work on text files. The *cut* utility can be used to  
 84120 turn files with arbitrary line lengths into a set of text files containing the same data. The *paste*  
 84121 utility can be used to create (or recreate) files with arbitrary line lengths. For example, if **file**  
 84122 contains long lines:

84123 `cut -b 1-500 -n file > file1`84124 `cut -b 501- -n file > file2`

84125 creates **file1** (a text file) with lines no longer than 500 bytes (plus the <newline>) and **file2** that  
 84126 contains the remainder of the data from **file**. (Note that **file2** is not a text file if there are lines in



**file** that are longer than 500 + {LINE\_MAX} bytes.) The original file can be recreated from **file1** and **file2** using the command:

```
paste -d "\0" file1 file2 > file
```

## RATIONALE

Some historical implementations do not count <backspace> characters in determining character counts with the **-c** option. This may be useful for using *cut* for processing *nroff* output. It was deliberately decided not to have the **-c** option treat either <backspace> or <tab> characters in any special fashion. The *fold* utility does treat these characters specially.

Unlike other utilities, some historical implementations of *cut* exit after not finding an input file, rather than continuing to process the remaining *file* operands. This behavior is prohibited by this volume of POSIX.1-2008, where only the exit status is affected by this problem.

The behavior of *cut* when provided with either mutually-exclusive options or options that do not work logically together has been deliberately left unspecified in favor of global wording in [Section 1.4](#) (on page 2310).

The OPTIONS section was changed in response to IEEE PASC Interpretation 1003.2 #149. The change represents historical practice on all known systems. The original standard was ambiguous on the nature of the output.

The *list* option-arguments are historically used to select the portions of the line to be written, but do not affect the order of the data. For example:

```
echo abcdefghi | cut -c6,2,4-7,1
```

yields "abdefg".

A proposal to enhance *cut* with the following option:

**-o** Preserve the selected field order. When this option is specified, each byte, character, or field (or ranges of such) shall be written in the order specified by the *list* option-argument, even if this requires multiple outputs of the same bytes, characters, or fields.

was rejected because this type of enhancement is outside the scope of the IEEE P1003.2b draft standard.

## FUTURE DIRECTIONS

None.

## SEE ALSO

[Section 2.5](#) (on page 2323), *fold*, *grep*, *paste*

XBD [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 215)

## CHANGE HISTORY

First released in Issue 2.

### Issue 6

The OPTIONS section is changed to align with the IEEE P1003.2b draft standard.

The normative text is reworded to avoid use of the term “must” for application requirements.

### Issue 7

SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

SD5-XCU-ERN-171 is applied, adding APPLICATION USAGE.

84167 **NAME**84168 cxref — generate a C-language program cross-reference table (**DEVELOPMENT**)84169 **SYNOPSIS**

```
84170 XSI cxref [-cs] [-o file] [-w num] [-D name[=def]]... [-I dir]...
84171 [-U name]... file...
```

84172 **DESCRIPTION**

84173 The *cxref* utility shall analyze a collection of C-language *files* and attempt to build a cross-  
 84174 reference table. Information from **#define** lines shall be included in the symbol table. A sorted  
 84175 listing shall be written to standard output of all symbols (auto, static, and global) in each *file*  
 84176 separately, or with the **-c** option, in combination. Each symbol shall contain an <asterisk> before  
 84177 the declaring reference.

84178 **OPTIONS**

84179 The *cxref* utility shall conform to XBD [Section 12.2](#) (on page 215), except that the order of the **-D**,  
 84180 **-I**, and **-U** options (which are identical to their interpretation by *c99*) is significant. The  
 84181 following options shall be supported:

- 84182 **-c** Write a combined cross-reference of all input files.
- 84183 **-s** Operate silently; do not print input filenames.
- 84184 **-o file** Direct output to named *file*.
- 84185 **-w num** Format output no wider than *num* (decimal) columns. This option defaults to 80 if  
 84186 *num* is not specified or is less than 51.
- 84187 **-D** Equivalent to *c99*.
- 84188 **-I** Equivalent to *c99*.
- 84189 **-U** Equivalent to *c99*.

84190 **OPERANDS**

84191 The following operand shall be supported:

- 84192 *file* A pathname of a C-language source file.

84193 **STDIN**

84194 Not used.

84195 **INPUT FILES**

84196 The input files are C-language source files.

84197 **ENVIRONMENT VARIABLES**

84198 The following environment variables shall affect the execution of *cxref*:

- 84199 **LANG** Provide a default value for the internationalization variables that are unset or null.  
 84200 (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 84201 variables used to determine the values of locale categories.)
- 84202 **LC\_ALL** If set to a non-empty string value, override the values of all the other  
 84203 internationalization variables.
- 84204 **LC\_COLLATE**  
 84205 Determine the locale for the ordering of the output.
- 84206 **LC\_CTYPE** Determine the locale for the interpretation of sequences of bytes of text data as  
 84207 characters (for example, single-byte as opposed to multi-byte characters in  
 84208 arguments and input files).

**84209 LC\_MESSAGES**

84210 Determine the locale that should be used to affect the format and contents of  
84211 diagnostic messages written to standard error.

84212 **NLSPATH** Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

**84213 ASYNCHRONOUS EVENTS**

84214 Default.

**84215 STDOUT**

84216 The standard output shall be used for the cross-reference listing, unless the **-o** option is used to  
84217 select a different output file.

84218 The format of standard output is unspecified, except that the following information shall be  
84219 included:

- 84220 • If the **-c** option is not specified, each portion of the listing shall start with the name of the  
84221 input file on a separate line.
- 84222 • The name line shall be followed by a sorted list of symbols, each with its associated  
84223 location pathname, the name of the function in which it appears (if it is not a function  
84224 name itself), and line number references.
- 84225 • Each line number may be preceded by an <asterisk> ('\*') flag, meaning that this is the  
84226 declaring reference. Other single-character flags, with implementation-defined meanings,  
84227 may be included.

**84228 STDERR**

84229 The standard error shall be used only for diagnostic messages.

**84230 OUTPUT FILES**

84231 The output file named by the **-o** option shall be used instead of standard output.

**84232 EXTENDED DESCRIPTION**

84233 None.

**84234 EXIT STATUS**

84235 The following exit values shall be returned:

- 84236 0 Successful completion.
- 84237 >0 An error occurred.

**84238 CONSEQUENCES OF ERRORS**

84239 Default.

**84240 APPLICATION USAGE**

84241 None.

**84242 EXAMPLES**

84243 None.

**84244 RATIONALE**

84245 None.

**84246 FUTURE DIRECTIONS**

84247 None.

84248 **SEE ALSO**84249 [c99](#)84250 XBD [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 215)84251 **CHANGE HISTORY**

84252 First released in Issue 2.

84253 **Issue 5**84254 In the SYNOPSIS, `[-U dir]` is changed to `[-U name]`.84255 **Issue 6**

84256 The APPLICATION USAGE section is added.

84257 **Issue 7**

84258 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

84259 **NAME**84260 `date` — write the date and time84261 **SYNOPSIS**84262 `date [-u] [+format]`84263 XSI `date [-u] mmddhhmm[[cc]yy]`84264 **DESCRIPTION**

84265 XSI The *date* utility shall write the date and time to standard output or attempt to set the system  
 84266 *date and time*. By default, the current date and time shall be written. If an operand beginning  
 84267 with '+' is specified, the output format of *date* shall be controlled by the conversion  
 84268 specifications and other text in the operand.

84269 **OPTIONS**84270 The *date* utility shall conform to XBD [Section 12.2](#) (on page 215).

84271 The following option shall be supported:

84272 **-u** Perform operations as if the *TZ* environment variable was set to the string "UTC0",  
 84273 or its equivalent historical value of "GMT0". Otherwise, *date* shall use the timezone  
 84274 indicated by the *TZ* environment variable or the system default if that variable is  
 84275 unset or null.

84276 **OPERANDS**

84277 The following operands shall be supported:

84278 **+format** When the format is specified, each conversion specifier shall be replaced in the  
 84279 standard output by its corresponding value. All other characters shall be copied to  
 84280 the output without change. The output shall always be terminated with a  
 84281 <newline>.

84282 **Conversion Specifications**

84283 %a Locale's abbreviated weekday name.  
 84284 %A Locale's full weekday name.  
 84285 %b Locale's abbreviated month name.  
 84286 %B Locale's full month name.  
 84287 %c Locale's appropriate date and time representation.  
 84288 %C Century (a year divided by 100 and truncated to an integer) as a decimal  
 84289 number [00,99].  
 84290 %d Day of the month as a decimal number [01,31].  
 84291 %D Date in the format *mm/dd/yy*.  
 84292 %e Day of the month as a decimal number [1,31] in a two-digit field with  
 84293 leading <space> character fill.  
 84294 %h A synonym for %b.  
 84295 %H Hour (24-hour clock) as a decimal number [00,23].  
 84296 %I Hour (12-hour clock) as a decimal number [01,12].

|       |    |                                                                                                                                                                                                                                                                                               |
|-------|----|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 84297 | %j | Day of the year as a decimal number [001,366].                                                                                                                                                                                                                                                |
| 84298 | %m | Month as a decimal number [01,12].                                                                                                                                                                                                                                                            |
| 84299 | %M | Minute as a decimal number [00,59].                                                                                                                                                                                                                                                           |
| 84300 | %n | A <newline>.                                                                                                                                                                                                                                                                                  |
| 84301 | %p | Locale's equivalent of either AM or PM.                                                                                                                                                                                                                                                       |
| 84302 | %r | 12-hour clock time [01,12] using the AM/PM notation; in the POSIX locale, this shall be equivalent to %I:%M:%S %p.                                                                                                                                                                            |
| 84303 |    |                                                                                                                                                                                                                                                                                               |
| 84304 | %S | Seconds as a decimal number [00,60].                                                                                                                                                                                                                                                          |
| 84305 | %t | A <tab>.                                                                                                                                                                                                                                                                                      |
| 84306 | %T | 24-hour clock time [00,23] in the format <i>HH:MM:SS</i> .                                                                                                                                                                                                                                    |
| 84307 | %u | Weekday as a decimal number [1,7] (1=Monday).                                                                                                                                                                                                                                                 |
| 84308 | %U | Week of the year (Sunday as the first day of the week) as a decimal number [00,53]. All days in a new year preceding the first Sunday shall be considered to be in week 0.                                                                                                                    |
| 84309 |    |                                                                                                                                                                                                                                                                                               |
| 84310 |    |                                                                                                                                                                                                                                                                                               |
| 84311 | %V | Week of the year (Monday as the first day of the week) as a decimal number [01,53]. If the week containing January 1 has four or more days in the new year, then it shall be considered week 1; otherwise, it shall be the last week of the previous year, and the next week shall be week 1. |
| 84312 |    |                                                                                                                                                                                                                                                                                               |
| 84313 |    |                                                                                                                                                                                                                                                                                               |
| 84314 |    |                                                                                                                                                                                                                                                                                               |
| 84315 | %w | Weekday as a decimal number [0,6] (0=Sunday).                                                                                                                                                                                                                                                 |
| 84316 | %W | Week of the year (Monday as the first day of the week) as a decimal number [00,53]. All days in a new year preceding the first Monday shall be considered to be in week 0.                                                                                                                    |
| 84317 |    |                                                                                                                                                                                                                                                                                               |
| 84318 |    |                                                                                                                                                                                                                                                                                               |
| 84319 | %x | Locale's appropriate date representation.                                                                                                                                                                                                                                                     |
| 84320 | %X | Locale's appropriate time representation.                                                                                                                                                                                                                                                     |
| 84321 | %y | Year within century [00,99].                                                                                                                                                                                                                                                                  |
| 84322 | %Y | Year with century as a decimal number.                                                                                                                                                                                                                                                        |
| 84323 | %Z | Timezone name, or no characters if no timezone is determinable.                                                                                                                                                                                                                               |
| 84324 | %% | A <percent-sign> character.                                                                                                                                                                                                                                                                   |

See XBD [Section 7.3.5](#) (on page 159) for the conversion specifier values in the POSIX locale.

### Modified Conversion Specifications

Some conversion specifiers can be modified by the **E** and **O** modifier characters to indicate a different format or specification as specified in the *LC\_TIME* locale description (see XBD [Section 7.3.5](#), on page 159). If the corresponding keyword (see **era**, **era\_year**, **era\_d\_fmt**, and **alt\_digits** in XBD [Section 7.3.5](#), on page 159) is not specified or not supported for the current locale, the unmodified conversion specifier value shall be used.

|       |     |                                                                |
|-------|-----|----------------------------------------------------------------|
| 84334 | %EC | Locale's alternative appropriate date and time representation. |
|-------|-----|----------------------------------------------------------------|

|       |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|-------|--------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 84335 | %EC          | The name of the base year (period) in the locale's alternative representation.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 84336 |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 84337 | %Ex          | Locale's alternative date representation.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 84338 | %EX          | Locale's alternative time representation.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 84339 | %Ey          | Offset from %EC (year only) in the locale's alternative representation.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 84340 | %EY          | Full alternative year representation.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 84341 | %Od          | Day of month using the locale's alternative numeric symbols.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 84342 | %Oe          | Day of month using the locale's alternative numeric symbols.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 84343 | %OH          | Hour (24-hour clock) using the locale's alternative numeric symbols.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| 84344 | %OI          | Hour (12-hour clock) using the locale's alternative numeric symbols.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| 84345 | %Om          | Month using the locale's alternative numeric symbols.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 84346 | %OM          | Minutes using the locale's alternative numeric symbols.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 84347 | %OS          | Seconds using the locale's alternative numeric symbols.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 84348 | %Ou          | Weekday as a number in the locale's alternative representation (Monday = 1).                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 84349 |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 84350 | %OU          | Week number of the year (Sunday as the first day of the week) using the locale's alternative numeric symbols.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 84351 |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 84352 | %OV          | Week number of the year (Monday as the first day of the week, rules corresponding to %V), using the locale's alternative numeric symbols.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 84353 |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 84354 | %Ow          | Weekday as a number in the locale's alternative representation (Sunday = 0).                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 84355 |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 84356 | %OW          | Week number of the year (Monday as the first day of the week) using the locale's alternative numeric symbols.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 84357 |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 84358 | %Oy          | Year (offset from %C) in alternative representation.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| 84359 | XSI          | <b><code>mmdhmm[[cc]yy]</code></b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 84360 |              | Attempt to set the system date and time from the value given in the operand. This is only possible if the user has appropriate privileges and the system permits the setting of the system date and time. The first <i>mm</i> is the month (number); <i>dd</i> is the day (number); <i>hh</i> is the hour (number, 24-hour system); the second <i>mm</i> is the minute (number); <i>cc</i> is the century and is the first two digits of the year (this is optional); <i>yy</i> is the last two digits of the year and is optional. If century is not specified, then values in the range [69,99] shall refer to years 1969 to 1999 inclusive, and values in the range [00,68] shall refer to years 2000 to 2068 inclusive. The current year is the default if <i>yy</i> is omitted. |
| 84361 |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 84362 |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 84363 |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 84364 |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 84365 |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 84366 |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 84367 |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 84368 |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 84369 |              | <b>Note:</b> It is expected that in a future version of this standard the default century inferred from a 2-digit year will change. (This would apply to all commands accepting a 2-digit year as input.)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 84370 |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 84371 |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 84372 | <b>STDIN</b> |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 84373 |              | Not used.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |

84374 **INPUT FILES**

84375 None.

84376 **ENVIRONMENT VARIABLES**84377 The following environment variables shall affect the execution of *date*:

84378 *LANG* Provide a default value for the internationalization variables that are unset or null.  
 84379 (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 84380 variables used to determine the values of locale categories.)

84381 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
 84382 internationalization variables.

84383 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
 84384 characters (for example, single-byte as opposed to multi-byte characters in  
 84385 arguments).

84386 *LC\_MESSAGES*

84387 Determine the locale that should be used to affect the format and contents of  
 84388 diagnostic messages written to standard error.

84389 *LC\_TIME* Determine the format and contents of date and time strings written by *date*.

84390 XSI *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

84391 *TZ* Determine the timezone in which the time and date are written, unless the *-u*  
 84392 option is specified. If the *TZ* variable is unset or null and *-u* is not specified, an  
 84393 unspecified system default timezone is used.

84394 **ASYNCHRONOUS EVENTS**

84395 Default.

84396 **STDOUT**

84397 When no formatting operand is specified, the output in the POSIX locale shall be equivalent to  
 84398 specifying:

84399 *date* "+%a %b %e %H:%M:%S %Z %Y"84400 **STDERR**

84401 The standard error shall be used only for diagnostic messages.

84402 **OUTPUT FILES**

84403 None.

84404 **EXTENDED DESCRIPTION**

84405 None.

84406 **EXIT STATUS**

84407 The following exit values shall be returned:

84408 0 The date was written successfully.

84409 &gt;0 An error occurred.

84410 **CONSEQUENCES OF ERRORS**

84411 Default.



## APPLICATION USAGE

Conversion specifiers are of unspecified format when not in the POSIX locale. Some of them can contain <newline> characters in some locales, so it may be difficult to use the format shown in standard output for parsing the output of *date* in those locales.

The range of values for %S extends from 0 to 60 seconds to accommodate the occasional leap second.

Although certain of the conversion specifiers in the POSIX locale (such as the name of the month) are shown with initial capital letters, this need not be the case in other locales. Programs using these fields may need to adjust the capitalization if the output is going to be used at the beginning of a sentence.

The date string formatting capabilities are intended for use in Gregorian-style calendars, possibly with a different starting year (or years). The %x and %c conversion specifications, however, are intended for local representation; these may be based on a different, non-Gregorian calendar.

The %C conversion specification was introduced to allow a fallback for the %EC (alternative year format base year); it can be viewed as the base of the current subdivision in the Gregorian calendar. The century number is calculated as the year divided by 100 and truncated to an integer; it should not be confused with the use of ordinal numbers for centuries (for example, “twenty-first century”). Both the %EY and %Y can then be viewed as the offset from %EC and %C, respectively.

The E and O modifiers modify the traditional conversion specifiers, so that they can always be used, even if the implementation (or the current locale) does not support the modifier.

The E modifier supports alternative date formats, such as the Japanese Emperor’s Era, as long as these are based on the Gregorian calendar system. Extending the E modifiers to other date elements may provide an implementation-defined extension capable of supporting other calendar systems, especially in combination with the O modifier.

The O modifier supports time and date formats using the locale’s alternative numerical symbols, such as Kanji or Hindi digits or ordinal number representation.

Non-European locales, whether they use Latin digits in computational items or not, often have local forms of the digits for use in date formats. This is not totally unknown even in Europe; a variant of dates uses Roman numerals for the months: the third day of September 1991 would be written as 3.IX.1991. In Japan, Kanji digits are regularly used for dates; in Arabic-speaking countries, Hindi digits are used. The %d, %e, %H, %I, %m, %S, %U, %w, %W, and %Y conversion specifications always return the date and time field in Latin digits (that is, 0 to 9). The %O modifier was introduced to support the use for display purposes of non-Latin digits. In the LC\_TIME category in *localedef*, the optional **alt\_digits** keyword is intended for this purpose. As an example, assume the following (partial) *localedef* source:

```
alt_digits " "; "I"; "II"; "III"; "IV"; "V"; "VI"; "VII"; "VIII" \
 "IX"; "X"; "XI"; "XII"
d_fmt "%e.%Om.%Y"
```

With the above date, the command:

```
date "+%x"
```

would yield 3.IX.1991. With the same **d\_fmt**, but without the **alt\_digits**, the command would yield 3.9.1991.

## EXAMPLES

1. The following are input/output examples of *date* used at arbitrary times in the POSIX locale:

```
$ date
Tue Jun 26 09:58:10 PDT 1990

$ date "+DATE: %m/%d/%y%nTIME: %H:%M:%S"
DATE: 11/02/91
TIME: 13:36:16

$ date "+TIME: %r"
TIME: 01:36:32 PM
```

2. Examples for Denmark, where the default date and time format is %a %d %b %Y %T %Z:

```
$ LANG=da_DK.iso_8859-1 date
ons 02 okt 1991 15:03:32 CET

$ LANG=da_DK.iso_8859-1 \
date "+DATO: %A den %e. %B %Y%nKLOKKEN: %H:%M:%S"
DATO: onsdag den 2. oktober 1991
KLOKKEN: 15:03:56
```

3. Examples for Germany, where the default date and time format is %a %d.%h.%Y, %T %Z:

```
$ LANG=De_DE.88591 date
Mi 02.Okt.1991, 15:01:21 MEZ

$ LANG=De_DE.88591 date "+DATUM: %A, %d. %B %Y%nZEIT: %H:%M:%S"
DATUM: Mittwoch, 02. Oktober 1991
ZEIT: 15:02:02
```

4. Examples for France, where the default date and time format is %a %d %h %Y %Z %T:

```
$ LANG=Fr_FR.88591 date
Mer 02 oct 1991 MET 15:03:32

$ LANG=Fr_FR.88591 date "+JOUR: %A %d %B %Y%nHEURE: %H:%M:%S"
JOUR: Mercredi 02 octobre 1991
HEURE: 15:03:56
```

## RATIONALE

Some of the new options for formatting are from the ISO C standard. The **-u** option was introduced to allow portable access to Coordinated Universal Time (UTC). The string "GMT0" is allowed as an equivalent TZ value to be compatible with all of the systems using the BSD implementation, where this option originated.

The %e format conversion specification (adopted from System V) was added because the ISO C standard conversion specifications did not provide any way to produce the historical default *date* output during the first nine days of any month.

There are two varieties of day and week numbering supported (in addition to any others created with the locale-dependent %E and %O modifier characters):

- The historical variety in which Sunday is the first day of the week and the weekdays preceding the first Sunday of the year are considered week 0. These are represented by %w and %U. A variant of this is %W, using Monday as the first day of the week, but still referring to week 0. This view of the calendar was retained because so many historical

|       |                                                                                                                                             |
|-------|---------------------------------------------------------------------------------------------------------------------------------------------|
| 84499 | applications depend on it and the ISO C standard <i>strftime()</i> function, on which many <i>date</i>                                      |
| 84500 | implementations are based, was defined in this way.                                                                                         |
| 84501 | <ul style="list-style-type: none"> <li>The international standard, based on the ISO 8601:2004 standard where Monday is the first</li> </ul> |
| 84502 |                                                                                                                                             |
| 84503 |                                                                                                                                             |
| 84504 |                                                                                                                                             |
| 84505 |                                                                                                                                             |
| 84506 |                                                                                                                                             |
| 84507 | <b>FUTURE DIRECTIONS</b>                                                                                                                    |
| 84508 | None.                                                                                                                                       |
| 84509 | <b>SEE ALSO</b>                                                                                                                             |
| 84510 | XBD <a href="#">Section 7.3.5</a> (on page 159), <a href="#">Chapter 8</a> (on page 173), <a href="#">Section 12.2</a> (on page 215)        |
| 84511 | XSH <a href="#">fprintf()</a> , <a href="#">strftime()</a>                                                                                  |
| 84512 | <b>CHANGE HISTORY</b>                                                                                                                       |
| 84513 | First released in Issue 2.                                                                                                                  |
| 84514 | <b>Issue 5</b>                                                                                                                              |
| 84515 | Changes are made for Year 2000 alignment.                                                                                                   |
| 84516 | <b>Issue 6</b>                                                                                                                              |
| 84517 | The following new requirements on POSIX implementations derive from alignment with the                                                      |
| 84518 |                                                                                                                                             |
| 84519 | Single UNIX Specification:                                                                                                                  |
| 84520 | <ul style="list-style-type: none"> <li>The <code>%EX</code> modified conversion specification is added.</li> </ul>                          |
| 84521 | The Open Group Corrigendum U048/2 is applied, correcting the examples.                                                                      |
| 84522 | The DESCRIPTION is updated to refer to conversion specifications, instead of field descriptors                                              |
| 84523 | for consistency with the <code>LC_TIME</code> category.                                                                                     |
| 84524 | A clarification is made such that the current year is the default if the <code>yy</code> argument is omitted                                |
| 84525 | when setting the system date and time.                                                                                                      |
| 84526 | IEEE Std 1003.1-2001/Cor 1-2002, item XCU/TC1/D6/19 is applied, correcting the CHANGE                                                       |
|       | HISTORY section.                                                                                                                            |

## 84527 NAME

84528 dd — convert and copy a file

## 84529 SYNOPSIS

84530 dd [*operand*...]

## 84531 DESCRIPTION

84532 The *dd* utility shall copy the specified input file to the specified output file with possible  
 84533 conversions using specific input and output block sizes. It shall read the input one block at a  
 84534 time, using the specified input block size; it shall then process the block of data actually  
 84535 returned, which could be smaller than the requested block size. It shall apply any conversions  
 84536 that have been specified and write the resulting data to the output in blocks of the specified  
 84537 output block size. If the **bs=expr** operand is specified and no conversions other than **sync**,  
 84538 **noerror**, or **notrunc** are requested, the data returned from each input block shall be written as a  
 84539 separate output block; if the read returns less than a full block and the **sync** conversion is not  
 84540 specified, the resulting output block shall be the same size as the input block. If the **bs=expr**  
 84541 operand is not specified, or a conversion other than **sync**, **noerror**, or **notrunc** is requested, the  
 84542 input shall be processed and collected into full-sized output blocks until the end of the input is  
 84543 reached.

84544 The processing order shall be as follows:

- 84545 1. An input block is read.
- 84546 2. If the input block is shorter than the specified input block size and the **sync** conversion is  
 84547 specified, null bytes shall be appended to the input data up to the specified size. (If either  
 84548 **block** or **unblock** is also specified, <space> characters shall be appended instead of null  
 84549 bytes.) The remaining conversions and output shall include the pad characters as if they  
 84550 had been read from the input.
- 84551 3. If the **bs=expr** operand is specified and no conversion other than **sync** or **noerror** is  
 84552 requested, the resulting data shall be written to the output as a single block, and the  
 84553 remaining steps are omitted.
- 84554 4. If the **swab** conversion is specified, each pair of input data bytes shall be swapped. If  
 84555 there is an odd number of bytes in the input block, the last byte in the input record shall  
 84556 not be swapped.
- 84557 5. Any remaining conversions (**block**, **unblock**, **lcase**, and **ucase**) shall be performed. These  
 84558 conversions shall operate on the input data independently of the input blocking; an input  
 84559 or output fixed-length record may span block boundaries.
- 84560 6. The data resulting from input or conversion or both shall be aggregated into output  
 84561 blocks of the specified size. After the end of input is reached, any remaining output shall  
 84562 be written as a block without padding if **conv=sync** is not specified; thus, the final output  
 84563 block may be shorter than the output block size.

## 84564 OPTIONS

84565 None.

## 84566 OPERANDS

84567 All of the operands shall be processed before any input is read. The following operands shall be  
 84568 supported:

- 84569 **if=file** Specify the input pathname; the default is standard input.
- 84570 **of=file** Specify the output pathname; the default is standard output. If the **seek=expr**  
 84571 conversion is not also specified, the output file shall be truncated before the copy  
 84572 begins if an explicit **of=file** operand is specified, unless **conv=notrunc** is specified.

|       |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|-------|-------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 84573 |                               | If <b>seek=expr</b> is specified, but <b>conv=notrunc</b> is not, the effect of the copy shall be to preserve the blocks in the output file over which <i>dd</i> seeks, but no other portion of the output file shall be preserved. (If the size of the seek plus the size of the input file is less than the previous size of the output file, the output file shall be shortened by the copy. If the input file is empty and either the size of the seek is greater than the previous size of the output file or the output file did not previously exist, the size of the output file shall be set to the file offset after the seek.)                          |
| 84574 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 84575 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 84576 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 84577 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 84578 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 84579 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 84580 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 84581 | <b>ibs=expr</b>               | Specify the input block size, in bytes, by <i>expr</i> (default is 512).                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| 84582 | <b>obs=expr</b>               | Specify the output block size, in bytes, by <i>expr</i> (default is 512).                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 84583 | <b>bs=expr</b>                | Set both input and output block sizes to <i>expr</i> bytes, superseding <b>ibs=</b> and <b>obs=</b> . If no conversion other than <b>sync</b> , <b>noerror</b> , and <b>notrunc</b> is specified, each input block shall be copied to the output as a single block without aggregating short blocks.                                                                                                                                                                                                                                                                                                                                                               |
| 84584 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 84585 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 84586 | <b>cbs=expr</b>               | Specify the conversion block size for <b>block</b> and <b>unblock</b> in bytes by <i>expr</i> (default is zero). If <b>cbs=</b> is omitted or given a value of zero, using <b>block</b> or <b>unblock</b> produces unspecified results.                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 84587 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 84588 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 84589 | XSI                           | The application shall ensure that this operand is also specified if the <b>conv=</b> operand is specified with a value of <b>ascii</b> , <b>ebcdic</b> , or <b>ibm</b> . For a <b>conv=</b> operand with an <b>ascii</b> value, the input is handled as described for the <b>unblock</b> value, except that characters are converted to ASCII before any trailing <space> characters are deleted. For <b>conv=</b> operands with <b>ebcdic</b> or <b>ibm</b> values, the input is handled as described for the <b>block</b> value except that the characters are converted to EBCDIC or IBM EBCDIC, respectively, after any trailing <space> characters are added. |
| 84590 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 84591 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 84592 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 84593 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 84594 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 84595 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 84596 | <b>skip=n</b>                 | Skip <i>n</i> input blocks (using the specified input block size) before starting to copy. On seekable files, the implementation shall read the blocks or seek past them; on non-seekable files, the blocks shall be read and the data shall be discarded.                                                                                                                                                                                                                                                                                                                                                                                                         |
| 84597 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 84598 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 84599 | <b>seek=n</b>                 | Skip <i>n</i> blocks (using the specified output block size) from the beginning of the output file before copying. On non-seekable files, existing blocks shall be read and space from the current end-of-file to the specified offset, if any, filled with null bytes; on seekable files, the implementation shall seek to the specified offset or read the blocks as described for non-seekable files.                                                                                                                                                                                                                                                           |
| 84600 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 84601 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 84602 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 84603 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 84604 | <b>count=n</b>                | Copy only <i>n</i> input blocks.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 84605 | <b>conv=value[,value ...]</b> |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 84606 |                               | Where <i>values</i> are <comma>-separated symbols from the following list:                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 84607 | XSI                           | <b>ascii</b> Convert EBCDIC to ASCII; see <a href="#">Table 4-7</a> (on page 2609).                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 84608 | XSI                           | <b>ebcdic</b> Convert ASCII to EBCDIC; see <a href="#">Table 4-7</a> (on page 2609).                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| 84609 | XSI                           | <b>ibm</b> Convert ASCII to a different EBCDIC set; see <a href="#">Table 4-8</a> (on page 2610).                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 84610 | XSI                           | The <b>ascii</b> , <b>ebcdic</b> , and <b>ibm</b> values are mutually-exclusive.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 84611 | <b>block</b>                  | Treat the input as a sequence of <newline>-terminated or end-of-file-terminated variable-length records independent of the input block boundaries. Each record shall be converted to a record with a fixed length specified by the conversion block size. Any <newline> shall be removed from the input line; <space> characters shall be appended to lines that are shorter than their conversion block size to fill the block. Lines that are longer than the conversion block size shall be truncated                                                                                                                                                           |
| 84612 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 84613 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 84614 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 84615 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 84616 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 84617 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |

|       |                |                                                                                    |
|-------|----------------|------------------------------------------------------------------------------------|
| 84618 |                | to the largest number of characters that fit into that size; the number of         |
| 84619 |                | truncated lines shall be reported (see the STDERR section).                        |
| 84620 |                | The <b>block</b> and <b>unblock</b> values are mutually-exclusive.                 |
| 84621 | <b>unblock</b> | Convert fixed-length records to variable length. Read a number of bytes            |
| 84622 |                | equal to the conversion block size (or the number of bytes remaining in            |
| 84623 |                | the input, if less than the conversion block size), delete all trailing            |
| 84624 |                | <space> characters, and append a <newline>.                                        |
| 84625 | <b>lcase</b>   | Map uppercase characters specified by the <i>LC_CTYPE</i> keyword                  |
| 84626 |                | <b>tolower</b> to the corresponding lowercase character. Characters for which      |
| 84627 |                | no mapping is specified shall not be modified by this conversion.                  |
| 84628 |                | The <b>lcase</b> and <b>ucase</b> symbols are mutually-exclusive.                  |
| 84629 | <b>ucase</b>   | Map lowercase characters specified by the <i>LC_CTYPE</i> keyword                  |
| 84630 |                | <b>toupper</b> to the corresponding uppercase character. Characters for            |
| 84631 |                | which no mapping is specified shall not be modified by this conversion.            |
| 84632 | <b>swab</b>    | Swap every pair of input bytes.                                                    |
| 84633 | <b>noerror</b> | Do not stop processing on an input error. When an input error occurs, a            |
| 84634 |                | diagnostic message shall be written on standard error, followed by the             |
| 84635 |                | current input and output block counts in the same format as used at                |
| 84636 |                | completion (see the STDERR section). If the <b>sync</b> conversion is              |
| 84637 |                | specified, the missing input shall be replaced with null bytes and                 |
| 84638 |                | processed normally; otherwise, the input block shall be omitted from               |
| 84639 |                | the output.                                                                        |
| 84640 | <b>notrunc</b> | Do not truncate the output file. Preserve blocks in the output file not            |
| 84641 |                | explicitly written by this invocation of the <i>dd</i> utility. (See also the      |
| 84642 |                | preceding <b>of=file</b> operand.)                                                 |
| 84643 | <b>sync</b>    | Pad every input block to the size of the <b>ibs=</b> buffer, appending null        |
| 84644 |                | bytes. (If either <b>block</b> or <b>unblock</b> is also specified, append <space> |
| 84645 |                | characters, rather than null bytes.)                                               |

The behavior is unspecified if operands other than **conv=** are specified more than once.

For the **bs=**, **cbs=**, **ibs=**, and **obs=** operands, the application shall supply an expression specifying a size in bytes. The expression, *expr*, can be:

1. A positive decimal number
2. A positive decimal number followed by *k*, specifying multiplication by 1 024
3. A positive decimal number followed by *b*, specifying multiplication by 512
4. Two or more positive decimal numbers (with or without *k* or *b*) separated by *x*, specifying the product of the indicated values

All of the operands are processed before any input is read.

XSI

The following two tables display the octal number character values used for the **ascii** and **ebcdic** conversions (first table) and for the **ibm** conversion (second table). In both tables, the ASCII values are the row and column headers and the EBCDIC values are found at their intersections. For example, ASCII 0012 (LF) is the second row, third column, yielding 0045 in EBCDIC. The inverted tables (for EBCDIC to ASCII conversion) are not shown, but are in one-to-one correspondence with these tables. The differences between the two tables are highlighted by

84661

small boxes drawn around five entries.

84662

Table 4-7 ASCII to EBCDIC Conversion

|      | 0   | 1   | 2   | 3   | 4   | 5   | 6   | 7   |
|------|-----|-----|-----|-----|-----|-----|-----|-----|
| 0000 | NUL | SOH | STX | ETX | EOT | ENQ | ACK | BEL |
| 0010 | BS  | HT  | LF  | VT  | FF  | CR  | SO  | SI  |
| 0020 | DLE | DC1 | DC2 | DC3 | DC4 | NAK | SYN | ETB |
| 0030 | CAN | EM  | SUB | ESC | IFS | IGS | IRS | ITB |
| 0040 | Sp  | !   | "   | #   | \$  | %   | &   | .   |
| 0050 | (   | )   | *   | +   | ,   | -   | .   | /   |
| 0060 | 0   | 1   | 2   | 3   | 4   | 5   | 6   | 7   |
| 0070 | 8   | 9   | :   | ;   | <   | =   | >   | ?   |
| 0100 | @   | A   | B   | C   | D   | E   | F   | G   |
| 0110 | H   | I   | J   | K   | L   | M   | N   | O   |
| 0120 | P   | Q   | R   | S   | T   | U   | V   | W   |
| 0130 | X   | Y   | Z   | [   | \   | ]   | ^   | _   |
| 0140 | ,   | a   | b   | c   | d   | e   | f   | g   |
| 0150 | h   | i   | j   | k   | l   | m   | n   | o   |
| 0160 | p   | q   | r   | s   | t   | u   | v   | w   |
| 0170 | x   | y   | z   | {   |     | }   | ~   | DEL |
| 0200 | DS  | SOS | FS  | WUS | BYP | NL  | RNL | POC |
| 0210 | SA  | SFE | SM  | CSP | MFA | SPS | RPT | CU1 |
| 0220 |     |     | UBS | IR  | PP  | TRN | NBS | GE  |
| 0230 | SBS | IT  | RFF | CU3 | SEL | RES |     |     |
| 0240 |     |     |     |     |     |     |     |     |
| 0250 |     |     |     |     |     |     |     |     |
| 0260 |     |     |     |     |     |     |     |     |
| 0270 |     |     |     |     |     |     |     |     |
| 0300 |     |     |     |     |     |     |     |     |
| 0310 |     |     |     |     |     |     |     |     |
| 0320 |     |     |     |     |     |     |     |     |
| 0330 |     |     |     |     |     |     |     |     |
| 0340 |     |     |     |     |     |     |     |     |
| 0350 |     |     |     |     |     |     |     |     |
| 0360 |     |     |     |     |     |     |     |     |
| 0370 |     |     |     |     |     |     |     |     |

84663

Table 4-8 ASCII to IBM EBCDIC Conversion

|      | 0    | 1   | 2    | 3   | 4    | 5   | 6    | 7   |      |     |      |     |      |     |      |     |
|------|------|-----|------|-----|------|-----|------|-----|------|-----|------|-----|------|-----|------|-----|
| 0000 | 0000 | NUL | 0001 | SOH | 0002 | STX | 0003 | ETX | 0004 | EOT | 0005 | ENQ | 0006 | ACK | 0007 | BEL |
| 0010 | 0026 | BS  | 0005 | HT  | 0045 | LF  | 0013 | VT  | 0014 | FF  | 0015 | CR  | 0016 | SO  | 0017 | SI  |
| 0020 | 0020 | DLE | 0021 | DC1 | 0022 | DC2 | 0023 | DC3 | 0074 | DC4 | 0075 | NAK | 0062 | SYN | 0046 | ETB |
| 0030 | 0030 | CAN | 0031 | EM  | 0077 | SUB | 0047 | ESC | 0034 | IFS | 0035 | IGS | 0036 | IRS | 0037 | ITB |
| 0040 | 0100 | Sp  | 0132 | !   | 0177 | "   | 0173 | #   | 0133 | \$  | 0154 | %   | 0120 | &   | 0175 | ,   |
| 0050 | 0115 | (   | 0135 | )   | 0134 | *   | 0116 | +   | 0153 | ,   | 0140 | -   | 0113 | .   | 0141 | /   |
| 0060 | 0360 | 0   | 0361 | 1   | 0362 | 2   | 0363 | 3   | 0364 | 4   | 0365 | 5   | 0366 | 6   | 0367 | 7   |
| 0070 | 0370 | 8   | 0371 | 9   | 0172 | :   | 0136 | ;   | 0114 | <   | 0176 | =   | 0156 | >   | 0157 | ?   |
| 0100 | 0174 | @   | 0301 | A   | 0302 | B   | 0303 | C   | 0304 | D   | 0305 | E   | 0306 | F   | 0307 | G   |
| 0110 | 0310 | H   | 0311 | I   | 0321 | J   | 0322 | K   | 0323 | L   | 0324 | M   | 0325 | N   | 0326 | O   |
| 0120 | 0327 | P   | 0330 | Q   | 0331 | R   | 0342 | S   | 0343 | T   | 0344 | U   | 0345 | V   | 0346 | W   |
| 0130 | 0347 | X   | 0350 | Y   | 0351 | Z   | 0255 | [   | 0340 | \   | 0275 | ]   | 0137 | ^   | 0155 | _   |
| 0140 | 0171 | ,   | 0201 | a   | 0202 | b   | 0203 | c   | 0204 | d   | 0205 | e   | 0206 | f   | 0207 | g   |
| 0150 | 0210 | h   | 0211 | i   | 0221 | j   | 0222 | k   | 0223 | l   | 0224 | m   | 0225 | n   | 0226 | o   |
| 0160 | 0227 | p   | 0230 | q   | 0231 | r   | 0242 | s   | 0243 | t   | 0244 | u   | 0245 | v   | 0246 | w   |
| 0170 | 0247 | x   | 0250 | y   | 0251 | z   | 0300 | {   | 0117 |     | 0320 | }   | 0241 | ~   | 0007 | DEL |
| 0200 | 0040 | DS  | 0041 | SOS | 0042 | FS  | 0043 | WUS | 0044 | BYP | 0025 | NL  | 0006 | RNL | 0027 | POC |
| 0210 | 0050 | SA  | 0051 | SFE | 0052 | SM  | 0053 | CSP | 0054 | MFA | 0011 | SPS | 0012 | RPT | 0033 | CU1 |
| 0220 | 0060 |     | 0061 |     | 0032 | UBS | 0063 | IR  | 0064 | PP  | 0065 | TRN | 0066 | NBS | 0010 | GE  |
| 0230 | 0070 | SBS | 0071 | IT  | 0072 | RFF | 0073 | CU3 | 0004 | SEL | 0024 | RES | 0076 |     | 0341 |     |
| 0240 | 0101 |     | 0102 |     | 0103 |     | 0104 |     | 0105 |     | 0106 |     | 0107 |     | 0110 |     |
| 0250 | 0111 |     | 0121 |     | 0122 |     | 0123 |     | 0124 |     | 0125 |     | 0126 |     | 0127 |     |
| 0260 | 0130 |     | 0131 |     | 0142 |     | 0143 |     | 0144 |     | 0145 |     | 0146 |     | 0147 |     |
| 0270 | 0150 |     | 0151 |     | 0160 |     | 0161 |     | 0162 |     | 0163 |     | 0164 |     | 0165 |     |
| 0300 | 0166 |     | 0167 |     | 0170 |     | 0200 |     | 0212 |     | 0213 |     | 0214 |     | 0215 |     |
| 0310 | 0216 |     | 0217 |     | 0220 |     | 0232 |     | 0233 |     | 0234 |     | 0235 |     | 0236 |     |
| 0320 | 0237 |     | 0240 |     | 0252 |     | 0253 |     | 0254 |     | 0255 | [   | 0256 |     | 0257 |     |
| 0330 | 0260 |     | 0261 |     | 0262 |     | 0263 |     | 0264 |     | 0265 |     | 0266 |     | 0267 |     |
| 0340 | 0270 |     | 0271 |     | 0272 |     | 0273 |     | 0274 |     | 0275 | ]   | 0276 |     | 0277 |     |
| 0350 | 0312 |     | 0313 |     | 0314 | J   | 0315 |     | 0316 | y   | 0317 |     | 0332 |     | 0333 |     |
| 0360 | 0334 |     | 0335 |     | 0336 |     | 0337 |     | 0352 |     | 0353 |     | 0354 | H   | 0355 |     |
| 0370 | 0356 |     | 0357 |     | 0372 | I   | 0373 |     | 0374 |     | 0375 |     | 0376 |     | 0377 | EO  |



84664 **STDIN**84665 If no **if=** operand is specified, the standard input shall be used. See the INPUT FILES section.84666 **INPUT FILES**

84667 The input file can be any file type.

84668 **ENVIRONMENT VARIABLES**84669 The following environment variables shall affect the execution of *dd*:

84670 **LANG** Provide a default value for the internationalization variables that are unset or null.  
 84671 (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 84672 variables used to determine the values of locale categories.)

84673 **LC\_ALL** If set to a non-empty string value, override the values of all the other  
 84674 internationalization variables.

84675 **LC\_CTYPE** Determine the locale for the interpretation of sequences of bytes of text data as  
 84676 characters (for example, single-byte as opposed to multi-byte characters in  
 84677 arguments and input files), the classification of characters as uppercase or  
 84678 lowercase, and the mapping of characters from one case to the other.

84679 **LC\_MESSAGES**

84680 Determine the locale that should be used to affect the format and contents of  
 84681 diagnostic messages written to standard error and informative messages written to  
 84682 standard output.

84683 **XSI NLSPATH** Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

84684 **ASYNCHRONOUS EVENTS**

84685 For SIGINT, the *dd* utility shall interrupt its current processing, write status information to  
 84686 standard error, and exit as though terminated by SIGINT. It shall take the standard action for all  
 84687 other signals; see the ASYNCHRONOUS EVENTS section in [Section 1.4](#) (on page 2310).

84688 **STDOUT**

84689 If no **of=** operand is specified, the standard output shall be used. The nature of the output  
 84690 depends on the operands selected.

84691 **STDERR**

84692 On completion, *dd* shall write the number of input and output blocks to standard error. In the  
 84693 POSIX locale the following formats shall be used:

84694 "%u+%u records in\n", <number of whole input blocks>,  
 84695 <number of partial input blocks>

84696 "%u+%u records out\n", <number of whole output blocks>,  
 84697 <number of partial output blocks>

84698 A partial input block is one for which *read()* returned less than the input block size. A partial  
 84699 output block is one that was written with fewer bytes than specified by the output block size.

84700 In addition, when there is at least one truncated block, the number of truncated blocks shall be  
 84701 written to standard error. In the POSIX locale, the format shall be:

84702 "%u truncated %s\n", <number of truncated blocks>, "record" (if  
 84703 <number of truncated blocks> is one) "records" (otherwise)

84704 Diagnostic messages may also be written to standard error.

**OUTPUT FILES**

If the **of=** operand is used, the output shall be the same as described in the STDOUT section.

**EXTENDED DESCRIPTION**

None.

**EXIT STATUS**

The following exit values shall be returned:

0 The input file was copied successfully.

>0 An error occurred.

**CONSEQUENCES OF ERRORS**

If an input error is detected and the **noerror** conversion has not been specified, any partial output block shall be written to the output file, a diagnostic message shall be written, and the copy operation shall be discontinued. If some other error is detected, a diagnostic message shall be written and the copy operation shall be discontinued.

**APPLICATION USAGE**

The input and output block size can be specified to take advantage of raw physical I/O.

There are many different versions of the EBCDIC codesets. The ASCII and EBCDIC conversions specified for the *dd* utility perform conversions for the version specified by the tables.

**EXAMPLES**

The following command:

```
dd if=/dev/rmt0h of=/dev/rmt1h
```

copies from tape drive 0 to tape drive 1, using a common historical device naming convention.

The following command:

```
dd ibs=10 skip=1
```

strips the first 10 bytes from standard input.

This example reads an EBCDIC tape blocked ten 80-byte EBCDIC card images per block into the ASCII file *x*:

```
dd if=/dev/tape of=x ibs=800 cbs=80 conv=ascii,lcase
```

**RATIONALE**

The **OPTIONS** section is listed as “None” because there are no options recognized by historical *dd* utilities. Certainly, many of the operands could have been designed to use the Utility Syntax Guidelines, which would have resulted in the classic hyphenated option letters. In this version of this volume of POSIX.1-2008, *dd* retains its curious JCL-like syntax due to the large number of applications that depend on the historical implementation.

A suggested implementation technique for **conv=noerror, sync** is to zero (or <space>-fill, if **blocking** or **unblocking**) the input buffer before each read and to write the contents of the input buffer to the output even after an error. In this manner, any data transferred to the input buffer before the error was detected is preserved. Another point is that a failed read on a regular file or a disk generally does not increment the file offset, and *dd* must then seek past the block on which the error occurred; otherwise, the input error occurs repetitively. When the input is a magnetic tape, however, the tape normally has passed the block containing the error when the error is reported, and thus no seek is necessary.

The default **ibs=** and **obs=** sizes are specified as 512 bytes because there are historical (largely portable) scripts that assume these values. If they were left unspecified, unusual results could

occur if an implementation chose an odd block size.

Historical implementations of *dd* used *creat()* when processing *of=file*. This makes the **seek=** operand unusable except on special files. The **conv=notrunc** feature was added because more recent BSD-based implementations use *open()* (without *O\_TRUNC*) instead of *creat()*, but they fail to delete output file contents after the data copied.

The *w* multiplier (historically meaning *word*), is used in System V to mean 2 and in 4.2 BSD to mean 4. Since *word* is inherently non-portable, its use is not supported by this volume of POSIX.1-2008.

Standard EBCDIC does not have the characters ' [ ' and ' ] '. The values used in the table are taken from a common print train that does contain them. Other than those characters, the print train values are not filled in, but appear to provide some of the motivation for the historical choice of translations reflected here.

The Standard EBCDIC table provides a 1:1 translation for all 256 bytes.

The IBM EBCDIC table does not provide such a translation. The marked cells in the tables differ in such a way that:

1. EBCDIC 0112 ( ' ¢ ' ) and 0152 (broken pipe) do not appear in the table.
2. EBCDIC 0137 ( ' ¬ ' ) translates to/from ASCII 0236 ( ' ^ ' ). In the standard table, EBCDIC 0232 (no graphic) is used.
3. EBCDIC 0241 ( ' ~ ' ) translates to/from ASCII 0176 ( ' ¨ ' ). In the standard table, EBCDIC 0137 ( ' ¬ ' ) is used.
4. 0255 ( ' [ ' ) and 0275 ( ' ] ' ) appear twice, once in the same place as for the standard table and once in place of 0112 ( ' ¢ ' ) and 0241 ( ' ~ ' ).

In net result:

EBCDIC 0275 ( ' ] ' ) displaced EBCDIC 0241 ( ' ~ ' ) in cell 0345.

That displaced EBCDIC 0137 ( ' ¬ ' ) in cell 0176.

That displaced EBCDIC 0232 (no graphic) in cell 0136.

That replaced EBCDIC 0152 (broken pipe) in cell 0313.

EBCDIC 0255 ( ' [ ' ) replaced EBCDIC 0112 ( ' ¢ ' ).

This translation, however, reflects historical practice that (ASCII) ' ~ ' and ' ¬ ' were often mapped to each other, as were ' [ ' and ' ¢ ' ; and ' ] ' and (EBCDIC) ' ~ ' .

The **cbs** operand is required if any of the **ascii**, **ebcdic**, or **ibm** operands are specified. For the **ascii** operand, the input is handled as described for the **unblock** operand except that characters are converted to ASCII before the trailing <space> characters are deleted. For the **ebcdic** and **ibm** operands, the input is handled as described for the **block** operand except that the characters are converted to EBCDIC or IBM EBCDIC after the trailing <space> characters are added.

The **block** and **unblock** keywords are from historical BSD practice.

The consistent use of the word **record** in standard error messages matches most historical practice. An earlier version of System V used **block**, but this has been updated in more recent releases.

Early proposals only allowed two numbers separated by *x* to be used in a product when specifying **bs=**, **cbs=**, **ibs=**, and **obs=** sizes. This was changed to reflect the historical practice of

- 84789 allowing multiple numbers in the product as provided by Version 7 and all releases of System V  
84790 and BSD.
- 84791 A change to the **swab** conversion is required to match historical practice and is the result of IEEE  
84792 PASC Interpretations 1003.2 #03 and #04, submitted for the ISO POSIX-2: 1993 standard.
- 84793 A change to the handling of SIGINT is required to match historical practice and is the result of  
84794 IEEE PASC Interpretation 1003.2 #06 submitted for the ISO POSIX-2: 1993 standard.
- 84795 **FUTURE DIRECTIONS**
- 84796 None.
- 84797 **SEE ALSO**
- 84798 [Section 1.4](#) (on page 2310), [sed](#), [tr](#)
- 84799 XBD [Chapter 8](#) (on page 173)
- 84800 **CHANGE HISTORY**
- 84801 First released in Issue 2.
- 84802 **Issue 5**
- 84803 The second paragraph of the **cbs=** description is reworded and marked EX.
- 84804 The FUTURE DIRECTIONS section is added.
- 84805 **Issue 6**
- 84806 Changes are made to **swab** conversion and SIGINT handling to align with the IEEE P1003.2b  
84807 draft standard.
- 84808 The normative text is reworded to avoid use of the term “must” for application requirements.
- 84809 IEEE PASC Interpretation 1003.2 #209 is applied, clarifying the interaction between **dd of=file** and  
84810 **conv=notrunc**.
- 84811 **Issue 7**
- 84812 Austin Group Interpretation 1003.1-2001 #102 is applied.
- 84813 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

84814 **NAME**84815       delta — make a delta (change) to an SCCS file (**DEVELOPMENT**)84816 **SYNOPSIS**84817 XSI       delta [-nps] [-g *list*] [-m *mrlist*] [-r *SID*] [-y[*comment*]] *file...*84818 **DESCRIPTION**84819       The *delta* utility shall be used to permanently introduce into the named SCCS files changes that  
84820       were made to the files retrieved by *get* (called the *g-files*, or generated files).84821 **OPTIONS**84822       The *delta* utility shall conform to XBD [Section 12.2](#) (on page 215), except that the *-y* option has an  
84823       optional option-argument. This optional option-argument shall not be presented as a separate  
84824       argument.

84825       The following options shall be supported:

84826       *-r SID*       Uniquely identify which delta is to be made to the SCCS file. The use of this option  
84827       shall be necessary only if two or more outstanding *get* commands for editing (*get*  
84828       *-e*) on the same SCCS file were done by the same person (login name). The *SID*  
84829       value specified with the *-r* option can be either the *SID* specified on the *get*  
84830       command line or the *SID* to be made as reported by the *get* utility; see *get* (on page  
84831       2788).84832       *-s*           Suppress the report to standard output of the activity associated with each *file*. See  
84833       the STDOUT section.84834       *-n*           Specify retention of the edited *g-file* (normally removed at completion of delta  
84835       processing).84836       *-g list*       Specify a *list* (see *get* for the definition of *list*) of deltas that shall be ignored when  
84837       the file is accessed at the change level (*SID*) created by this delta.84838       *-m mrlist*   Specify a modification request (MR) number that the application shall supply as  
84839       the reason for creating the new delta. This shall be used if the SCCS file has the *v*  
84840       flag set; see *admin*.84841       If *-m* is not used and '*-*' is not specified as a file argument, and the standard  
84842       input is a terminal, the prompt described in the STDOUT section shall be written  
84843       to standard output before the standard input is read; if the standard input is not a  
84844       terminal, no prompt shall be issued.84845       MRs in a list shall be separated by <blank> characters or escaped <newline>  
84846       characters. An unescaped <newline> shall terminate the MR list. The escape  
84847       character is <backslash>.84848       If the *v* flag has a value, it shall be taken to be the name of a program which  
84849       validates the correctness of the MR numbers. If a non-zero exit status is returned  
84850       from the MR number validation program, the *delta* utility shall terminate. (It is  
84851       assumed that the MR numbers were not all valid.)84852       *-y[comment]* Describe the reason for making the delta. The *comment* shall be an arbitrary group  
84853       of lines that would meet the definition of a text file. Implementations shall support  
84854       *comments* from zero to 512 bytes and may support longer values. A null string  
84855       (specified as either *-y*, *-y*"", or in response to a prompt for a comment) shall be  
84856       considered a valid *comment*.84857       If *-y* is not specified and '*-*' is not specified as a file argument, and the standard

84858 input is a terminal, the prompt described in the STDOUT section shall be written  
 84859 to standard output before the standard input is read; if the standard input is not a  
 84860 terminal, no prompt shall be issued. An unescaped <newline> shall terminate the  
 84861 comment text. The escape character is <backslash>.

84862 The `-y` option shall be required if the *file* operand is specified as `'-'`.

84863 **-p** Write (to standard output) the SCCS file differences before and after the delta is  
 84864 applied in *diff* format; see *diff*.

## 84865 OPERANDS

84866 The following operand shall be supported:

84867 *file* A pathname of an existing SCCS file or a directory. If *file* is a directory, the *delta*  
 84868 utility shall behave as though each file in the directory were specified as a named  
 84869 file, except that non-SCCS files (last component of the pathname does not begin  
 84870 with *s.*) and unreadable files shall be silently ignored.

84871 If exactly one *file* operand appears, and it is `'-'`, the standard input shall be read;  
 84872 each line of the standard input shall be taken to be the name of an SCCS file to be  
 84873 processed. Non-SCCS files and unreadable files shall be silently ignored.

## 84874 STDIN

84875 The standard input shall be a text file used only in the following cases:

- 84876 • To read an *mrlist* or a *comment* (see the `-m` and `-y` options).
- 84877 • A *file* operand shall be specified as `'-'`. In this case, the `-y` option must be used to specify  
 84878 the comment, and if the SCCS file has the *v* flag set, the `-m` option must also be used to  
 84879 specify the MR list.

## 84880 INPUT FILES

84881 Input files shall be text files whose data is to be included in the SCCS files. If the first character of  
 84882 any line of an input file is <SOH> in the POSIX locale, the results are unspecified. If this file  
 84883 contains more than 99 999 lines, the number of lines recorded in the header for this file shall be  
 84884 99 999 for this delta.

## 84885 ENVIRONMENT VARIABLES

84886 The following environment variables shall affect the execution of *delta*:

84887 *LANG* Provide a default value for the internationalization variables that are unset or null.  
 84888 (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 84889 variables used to determine the values of locale categories.)

84890 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
 84891 internationalization variables.

84892 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
 84893 characters (for example, single-byte as opposed to multi-byte characters in  
 84894 arguments and input files).

84895 *LC\_MESSAGES*

84896 Determine the locale that should be used to affect the format and contents of  
 84897 diagnostic messages written to standard error, and informative messages written  
 84898 to standard output.

84899 *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

84900 **TZ** Determine the timezone in which the time and date are written in the SCCS file. If  
 84901 the *TZ* variable is unset or NULL, an unspecified system default timezone is used.

#### 84902 **ASYNCHRONOUS EVENTS**

84903 If SIGINT is caught, temporary files shall be cleaned up and *delta* shall exit with a non-zero exit  
 84904 code. The standard action shall be taken for all other signals; see [Section 1.4](#) (on page 2310).

#### 84905 **STDOUT**

84906 The standard output shall be used only for the following messages in the POSIX locale:

- 84907 • Prompts (see the **-m** and **-y** options) in the following formats:

84908 "MRs? "

84909 "comments? "

84910 The MR prompt, if written, shall always precede the comments prompt.

- 84911 • A report of each file's activities (unless the **-s** option is specified) in the following format:

84912 "%s\n%d inserted\n%d deleted\n%d unchanged\n", <New SID>,  
 84913 <number of lines inserted>, <number of lines deleted>,  
 84914 <number of lines unchanged>

#### 84915 **STDERR**

84916 The standard error shall be used only for diagnostic messages.

#### 84917 **OUTPUT FILES**

84918 Any SCCS files updated shall be files of an unspecified format.

#### 84919 **EXTENDED DESCRIPTION**

##### 84920 **System Date and Time**

84921 When a delta is added to an SCCS file, the system date and time shall be recorded for the new  
 84922 delta. If a *get* is performed using an SCCS file with a date recorded apparently in the future, the  
 84923 behavior is unspecified.

#### 84924 **EXIT STATUS**

84925 The following exit values shall be returned:

84926 0 Successful completion.

84927 >0 An error occurred.

#### 84928 **CONSEQUENCES OF ERRORS**

84929 Default.

#### 84930 **APPLICATION USAGE**

84931 Problems can arise if the system date and time have been modified (for example, put forward  
 84932 and then back again, or unsynchronized clocks across a network) and can also arise when  
 84933 different values of the *TZ* environment variable are used.

84934 Problems of a similar nature can also arise for the operation of the *get* utility, which records the  
 84935 date and time in the file body.

#### 84936 **EXAMPLES**

84937 None.

84938 **RATIONALE**

84939 None.

84940 **FUTURE DIRECTIONS**

84941 None.

84942 **SEE ALSO**84943 [Section 1.4](#) (on page 2310), *admin*, *diff*, *get*, *prs*, *rmdel*84944 XBD [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 215)84945 **CHANGE HISTORY**

84946 First released in Issue 2.

84947 **Issue 5**

84948 The output format description in the STDOUT section is corrected.

84949 **Issue 6**

84950 The APPLICATION USAGE section is added.

84951 The normative text is reworded to avoid use of the term “must” for application requirements.

84952 The Open Group Base Resolution bwg2001-007 is applied as follows:

- 84953 • The use of ‘-’ as a file argument is clarified.
- 84954 • The use of STDIN is added.
- 84955 • The ASYNCHRONOUS EVENTS section is updated to remove the implicit requirement that implementations re-signal themselves when catching a normally fatal signal.
- 84956
- 84957 • New text is added to the INPUT FILES section warning that the maximum lines recorded
- 84958 in the file is 99 999.

84959 New text is added to the EXTENDED DESCRIPTION and APPLICATION USAGE sections  
84960 regarding how the system date and time may be taken into account, and the TZ environment  
84961 variable is added to the ENVIRONMENT VARIABLES section as per The Open Group Base  
84962 Resolution bwg2001-007.

84963 **Issue 7**

84964 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.



84965 **NAME**

84966           df — report free disk space

84967 **SYNOPSIS**84968 XSI       df [-k] [-P|-t] [*file...*]84969 **DESCRIPTION**

84970 XSI       The *df* utility shall write the amount of available space and file slots for file systems on which  
 84971 the invoking user has appropriate read access. File systems shall be specified by the *file*  
 84972 operands; when none are specified, information shall be written for all file systems. The format  
 84973 of the default output from *df* is unspecified, but all space figures are reported in 512-byte units,  
 84974 unless the **-k** option is specified. This output shall contain at least the file system names, amount  
 84975 XSI of available space on each of these file systems, and, if no options other than **-t** are specified, the  
 84976 number of free file slots, or *inodes*, available; when **-t** is specified, the output shall contain the  
 84977 total allocated space as well.

84978 **OPTIONS**84979           The *df* utility shall conform to XBD [Section 12.2](#) (on page 215).

84980           The following options shall be supported:

84981           **-k**           Use 1024-byte units, instead of the default 512-byte units, when writing space  
 84982 figures.

84983           **-P**           Produce output in the format described in the STDOUT section.

84984 XSI       **-t**           Include total allocated-space figures in the output.

84985 **OPERANDS**

84986           The following operand shall be supported:

84987           *file*       A pathname of a file within the hierarchy of the desired file system. If a file other  
 84988 XSI than a FIFO, a regular file, a directory, or a special file representing the device  
 84989 containing the file system (for example, */dev/dsk/0s1*) is specified, the results are  
 84990 unspecified. If the *file* operand names a file other than a special file containing a file  
 84991 system, *df* shall write the amount of free space in the file system containing the  
 84992 XSI specified *file* operand. Otherwise, *df* shall write the amount of free space in that  
 84993 file system.

84994 **STDIN**

84995           Not used.

84996 **INPUT FILES**

84997           None.

84998 **ENVIRONMENT VARIABLES**84999           The following environment variables shall affect the execution of *df*:

85000           **LANG**       Provide a default value for the internationalization variables that are unset or null.  
 85001 (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 85002 variables used to determine the values of locale categories.)

85003           **LC\_ALL**      If set to a non-empty string value, override the values of all the other  
 85004 internationalization variables.

85005           **LC\_CTYPE**   Determine the locale for the interpretation of sequences of bytes of text data as  
 85006 characters (for example, single-byte as opposed to multi-byte characters in  
 85007 arguments).

85008 *LC\_MESSAGES*

85009 Determine the locale that should be used to affect the format and contents of

85010 diagnostic messages written to standard error and informative messages written to

85011 standard output.

85012 XSI *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

85013 **ASYNCHRONOUS EVENTS**

85014 Default.

85015 **STDOUT**

85016 When both the **-k** and **-P** options are specified, the following header line shall be written (in the

85017 POSIX locale):

85018 "Filesystem 1024-blocks Used Available Capacity Mounted on\n"

85019 When the **-P** option is specified without the **-k** option, the following header line shall be written

85020 (in the POSIX locale):

85021 "Filesystem 512-blocks Used Available Capacity Mounted on\n"

85022 The implementation may adjust the spacing of the header line and the individual data lines so

85023 that the information is presented in orderly columns.

85024 The remaining output with **-P** shall consist of one line of information for each specified file

85025 system. These lines shall be formatted as follows:

85026 "%s %d %d %d %d%% %s\n", <file system name>, <total space>,  
85027 <space used>, <space free>, <percentage used>,  
85028 <file system root>

85029 In the following list, all quantities expressed in 512-byte units (1 024-byte when **-k** is specified)

85030 shall be rounded up to the next higher unit. The fields are:

85031 <file system name>

85032 The name of the file system, in an implementation-defined format.

85033 <total space> The total size of the file system in 512-byte units. The exact meaning of this figure

85034 is implementation-defined, but should include <space used>, <space free>, plus any

85035 space reserved by the system not normally available to a user.

85036 <space used> The total amount of space allocated to existing files in the file system, in 512-byte

85037 units.

85038 <space free> The total amount of space available within the file system for the creation of new

85039 files by unprivileged users, in 512-byte units. When this figure is less than or equal

85040 to zero, it shall not be possible to create any new files on the file system without

85041 first deleting others, unless the process has appropriate privileges. The figure

85042 written may be less than zero.

85043 <percentage used>

85044 The percentage of the normally available space that is currently allocated to all files

85045 on the file system. This shall be calculated using the fraction:

85046 
$$\frac{\text{<space used>}}{(\text{<space used>} + \text{<space free>})}$$

85047 expressed as a percentage. This percentage may be greater than 100 if <space free>

85048 is less than zero. The percentage value shall be expressed as a positive integer, with

85049 any fractional result causing it to be rounded to the next highest integer.

85050 <file system root>

85051 The directory below which the file system hierarchy appears.

85052 XSI The output format is unspecified when **-t** is used.

## 85053 STDERR

85054 The standard error shall be used only for diagnostic messages.

## 85055 OUTPUT FILES

85056 None.

## 85057 EXTENDED DESCRIPTION

85058 None.

## 85059 EXIT STATUS

85060 The following exit values shall be returned:

85061 0 Successful completion.

85062 >0 An error occurred.

## 85063 CONSEQUENCES OF ERRORS

85064 Default.

## 85065 APPLICATION USAGE

85066 On most systems, the “name of the file system, in an implementation-defined format” is the  
85067 special file on which the file system is mounted.

85068 On large file systems, the calculation specified for percentage used can create huge rounding  
85069 errors.

## 85070 EXAMPLES

85071 1. The following example writes portable information about the **/usr** file system:

85072 `df -P /usr`

85073 2. Assuming that **/usr/src** is part of the **/usr** file system, the following produces the same  
85074 output as the previous example:

85075 `df -P /usr/src`

## 85076 RATIONALE

85077 The behavior of *df* with the **-P** option is the default action of the 4.2 BSD *df* utility. The uppercase  
85078 **-P** was selected to avoid collision with a known industry extension using **-p**.

85079 Historical *df* implementations vary considerably in their default output. It was therefore  
85080 necessary to describe the default output in a loose manner to accommodate all known historical  
85081 implementations and to add a portable option (**-P**) to provide information in a portable format.

85082 The use of 512-byte units is historical practice and maintains compatibility with *ls* and other  
85083 utilities in this volume of POSIX.1-2008. This does not mandate that the file system itself be  
85084 based on 512-byte blocks. The **-k** option was added as a compromise measure. It was agreed by  
85085 the standard developers that 512 bytes was the best default unit because of its complete  
85086 historical consistency on System V (*versus* the mixed 512/1024-byte usage on BSD systems), and  
85087 that a **-k** option to switch to 1024-byte units was a good compromise. Users who prefer the  
85088 more logical 1024-byte quantity can easily alias *df* to *df -k* without breaking many historical  
85089 scripts relying on the 512-byte units.

85090 It was suggested that *df* and the various related utilities be modified to access a *BLOCKSIZE*  
85091 environment variable to achieve consistency and user acceptance. Since this is not historical

85092 practice on any system, it is left as a possible area for system extensions and will be re-evaluated  
 85093 in a future version if it is widely implemented.

#### 85094 **FUTURE DIRECTIONS**

85095 None.

#### 85096 **SEE ALSO**

85097 *find*

85098 XBD [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 215)

#### 85099 **CHANGE HISTORY**

85100 First released in Issue 2.

#### 85101 **Issue 6**

85102 This utility is marked as part of the User Portability Utilities option.

#### 85103 **Issue 7**

85104 Austin Group Interpretation 1003.1-2001 #099 is applied.

85105 The *df* utility is removed from the User Portability Utilities option. User Portability Utilities is  
 85106 now an option for interactive utilities.

85107 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

85108 POSIX.1-2008, Technical Corrigendum 1, XCU/TC1-2008/0082 [156] is applied.

+

85109 **NAME**85110 `diff` — compare two files85111 **SYNOPSIS**85112 `diff [-c|-e|-f|-u|-C n|-U n] [-br] file1 file2`85113 **DESCRIPTION**

85114 The *diff* utility shall compare the contents of *file1* and *file2* and write to standard output a list of  
 85115 changes necessary to convert *file1* into *file2*. This list should be minimal. No output shall be  
 85116 produced if the files are identical.

85117 **OPTIONS**85118 The *diff* utility shall conform to XBD [Section 12.2](#) (on page 215).

85119 The following options shall be supported:

85120 **-b** Cause any amount of white space at the end of a line to be treated as a single  
 85121 <newline> (that is, the white-space characters preceding the <newline> are  
 85122 ignored) and other strings of white-space characters, not including <newline>  
 85123 characters, to compare equal.

85124 **-c** Produce output in a form that provides three lines of copied context.

85125 **-C n** Produce output in a form that provides *n* lines of copied context (where *n* shall be  
 85126 interpreted as a positive decimal integer).

85127 **-e** Produce output in a form suitable as input for the *ed* utility, which can then be used  
 85128 to convert *file1* into *file2*.

85129 **-f** Produce output in an alternative form, similar in format to **-e**, but not intended to  
 85130 be suitable as input for the *ed* utility, and in the opposite order.

85131 **-r** Apply *diff* recursively to files and directories of the same name when *file1* and *file2*  
 85132 are both directories.

85133 The *diff* utility shall detect infinite loops; that is, entering a previously visited  
 85134 directory that is an ancestor of the last file encountered. When it detects an infinite  
 85135 loop, *diff* shall write a diagnostic message to standard error and shall either recover  
 85136 its position in the hierarchy or terminate.

85137 **-u** Produce output in a form that provides three lines of unified context.

85138 **-U n** Produce output in a form that provides *n* lines of unified context (where *n* shall be  
 85139 interpreted as a non-negative decimal integer).

85140 **OPERANDS**

85141 The following operands shall be supported:

85142 *file1, file2* A pathname of a file to be compared. If either the *file1* or *file2* operand is `'-'`, the  
 85143 standard input shall be used in its place.

85144 If both *file1* and *file2* are directories, *diff* shall not compare block special files, character special  
 85145 files, or FIFO special files to any files and shall not compare regular files to directories. Further  
 85146 details are as specified in [Diff Directory Comparison Format](#) (on page 2624). The behavior of *diff*  
 85147 on other file types is implementation-defined when found in directories.

85148 If only one of *file1* and *file2* is a directory, *diff* shall be applied to the non-directory file and the file  
 85149 contained in the directory file with a filename that is the same as the last component of the non-  
 85150 directory file.

85151 **STDIN**

85152 The standard input shall be used only if one of the *file1* or *file2* operands references standard  
 85153 input. See the INPUT FILES section.

85154 **INPUT FILES**

85155 The input files may be of any type.

85156 **ENVIRONMENT VARIABLES**

85157 The following environment variables shall affect the execution of *diff*:

85158 *LANG* Provide a default value for the internationalization variables that are unset or null.  
 85159 (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 85160 variables used to determine the values of locale categories.)

85161 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
 85162 internationalization variables.

85163 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
 85164 characters (for example, single-byte as opposed to multi-byte characters in  
 85165 arguments and input files).

85166 *LC\_MESSAGES*

85167 Determine the locale that should be used to affect the format and contents of  
 85168 diagnostic messages written to standard error and informative messages written to  
 85169 standard output.

85170 *LC\_TIME* Determine the locale for affecting the format of file timestamps written with the *-C*  
 85171 and *-c* options.

85172 XSI *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

85173 *TZ* Determine the timezone used for calculating file timestamps written with a context  
 85174 format. If *TZ* is unset or null, an unspecified default timezone shall be used.

85175 **ASYNCHRONOUS EVENTS**

85176 Default.

85177 **STDOUT**85178 **Diff Directory Comparison Format**

85179 If both *file1* and *file2* are directories, the following output formats shall be used.

85180 In the POSIX locale, each file that is present in only one directory shall be reported using the  
 85181 following format:

85182 "Only in %s: %s\n", <directory pathname>, <filename>

85183 In the POSIX locale, subdirectories that are common to the two directories may be reported with  
 85184 the following format:

85185 "Common subdirectories: %s and %s\n", <directory1 pathname>,  
 85186 <directory2 pathname>

85187 For each file common to the two directories, if the two files are not to be compared: if the two  
 85188 files have the same device ID and file serial number, or are both block special files that refer to  
 85189 the same device, or are both character special files that refer to the same device, in the POSIX  
 85190 locale the output format is unspecified. Otherwise, in the POSIX locale an unspecified format  
 85191 shall be used that contains the pathnames of the two files.

85192 For each file common to the two directories, if the files are compared and are identical, no

85193 output shall be written. If the two files differ, the following format is written:

85194 "diff %s %s %s\n", <diff\_options>, <filename1>, <filename2>

85195 where <diff\_options> are the options as specified on the command line.

85196 All directory pathnames listed in this section shall be relative to the original command line  
85197 arguments. All other names of files listed in this section shall be filenames (pathname  
85198 components).

### 85199 **Diff Binary Output Format**

85200 In the POSIX locale, if one or both of the files being compared are not text files, it is  
85201 implementation-defined whether *diff* uses the binary file output format or the other formats as  
85202 specified below. The binary file output format shall contain the pathnames of two files being  
85203 compared and the string "differ".

85204 If both files being compared are text files, depending on the options specified, one of the  
85205 following formats shall be used to write the differences.

### 85206 **Diff Default Output Format**

85207 The default (without **-e**, **-f**, **-c**, **-C**, **-u**, or **-U** options) *diff* utility output shall contain lines of  
85208 these forms:

85209 "%da%d\n", <num1>, <num2>

85210 "%da%d,%d\n", <num1>, <num2>, <num3>

85211 "%dd%d\n", <num1>, <num2>

85212 "%d,%dd%d\n", <num1>, <num2>, <num3>

85213 "%dc%d\n", <num1>, <num2>

85214 "%d,%dc%d\n", <num1>, <num2>, <num3>

85215 "%dc%d,%d\n", <num1>, <num2>, <num3>

85216 "%d,%dc%d,%d\n", <num1>, <num2>, <num3>, <num4>

85217 These lines resemble *ed* subcommands to convert *file1* into *file2*. The line numbers before the  
85218 action letters shall pertain to *file1*; those after shall pertain to *file2*. Thus, by exchanging *a* for *d*  
85219 and reading the line in reverse order, one can also determine how to convert *file2* into *file1*. As in  
85220 *ed*, identical pairs (where *num1*=*num2*) are abbreviated as a single number.

85221 Following each of these lines, *diff* shall write to standard output all lines affected in the first file  
85222 using the format:

85223 "<Δ%s", <line>

85224 and all lines affected in the second file using the format:

85225 ">Δ%s", <line>

85226 If there are lines affected in both *file1* and *file2* (as with the **c** subcommand), the changes are  
85227 separated with a line consisting of three <hyphen> characters:

85228 "---\n"

**Diff -e Output Format**

With the **-e** option, a script shall be produced that shall, when provided as input to *ed*, along with an appended **w** (write) command, convert *file1* into *file2*. Only the **a** (append), **c** (change), **d** (delete), **i** (insert), and **s** (substitute) commands of *ed* shall be used in this script. Text lines, except those consisting of the single character <period> ('.'), shall be output as they appear in the file.

**Diff -f Output Format**

With the **-f** option, an alternative format of script shall be produced. It is similar to that produced by **-e**, with the following differences:

1. It is expressed in reverse sequence; the output of **-e** orders changes from the end of the file to the beginning; the **-f** from beginning to end.
2. The command form <lines> <command-letter> used by **-e** is reversed. For example, 10c with **-e** would be c10 with **-f**.
3. The form used for ranges of line numbers is <space>-separated, rather than <comma>-separated.

**Diff -c or -C Output Format**

With the **-c** or **-C** option, the output format shall consist of affected lines along with surrounding lines of context. The affected lines shall show which ones need to be deleted or changed in *file1*, and those added from *file2*. With the **-c** option, three lines of context, if available, shall be written before and after the affected lines. With the **-C** option, the user can specify how many lines of context are written. The exact format follows.

The name and last modification time of each file shall be output in the following format:

```
*** %s %s\n", file1, <file1 timestamp>
--- %s %s\n", file2, <file2 timestamp>
```

Each <file> field shall be the pathname of the corresponding file being compared. The pathname written for standard input is unspecified.

In the POSIX locale, each <timestamp> field shall be equivalent to the output from the following command:

```
date "+%a %b %e %T %Y"
```

without the trailing <newline>, executed at the time of last modification of the corresponding file (or the current time, if the file is standard input).

Then, the following output formats shall be applied for every set of changes.

First, a line shall be written in the following format:

```
*****\n"
```

Next, the range of lines in *file1* shall be written in the following format if the range contains two or more lines:

```
*** %d,%d *****\n", <beginning line number>, <ending line number>
```

and the following format otherwise:

```
*** %d *****\n", <ending line number>
```

The ending line number of an empty range shall be the number of the preceding line, or 0 if the



85269 range is at the start of the file.

85270 Next, the affected lines along with lines of context (unaffected lines) shall be written. Unaffected  
85271 lines shall be written in the following format:

85272 " $\Delta\Delta$ %s", <unaffected\_line>

85273 Deleted lines shall be written as:

85274 " $-\Delta$ %s", <deleted\_line>

85275 Changed lines shall be written as:

85276 " $!\Delta$ %s", <changed\_line>

85277 Next, the range of lines in *file2* shall be written in the following format if the range contains two  
85278 or more lines:

85279 "--- %d,%d ----\n", <beginning line number>, <ending line number>

85280 and the following format otherwise:

85281 "--- %d ----\n", <ending line number>

85282 Then, lines of context and changed lines shall be written as described in the previous formats.  
85283 Lines added from *file2* shall be written in the following format:

85284 "+ $\Delta$ %s", <added\_line>

#### 85285 **Diff -u or -U Output Format**

85286 The -u or -U options behave like the -c or -C options, except that the context lines are not  
85287 repeated; instead, the context, deleted, and added lines are shown together, interleaved. The  
85288 exact format follows.

85289 The name and last modification time of each file shall be output in the following format:

85290 "--- $\Delta$ %s%s% $\Delta$ %s0, file1, <file1 timestamp>, <file1 frac>, <file1 zone>  
85291 "+++ $\Delta$ %s%s% $\Delta$ %s0, file2, <file2 timestamp>, <file2 frac>, <file2 zone>

85292 Each <file> field shall be the pathname of the corresponding file being compared, or the single  
85293 character '-' if standard input is being compared. However, if the pathname contains a <tab>  
85294 or a <newline>, or if it does not consist entirely of characters taken from the portable character  
85295 set, the behavior is implementation-defined.

85296 Each <timestamp> field shall be equivalent to the output from the following command:

85297 date '+%Y-%m-%d $\Delta$ %H:%M:%S'

85298 without the trailing <newline>, executed at the time of last modification of the corresponding  
85299 file (or the current time, if the file is standard input).

85300 Each <frac> field shall be either empty, or a decimal point followed by at least one decimal digit,  
85301 indicating the fractional-seconds part (if any) of the file timestamp. The number of fractional  
85302 digits shall be at least the number needed to represent the file's timestamp without loss of  
85303 information.

85304 Each <zone> field shall be of the form "shhmm", where "shh" is a signed two-digit decimal  
85305 number in the range -24 through +25, and "mm" is an unsigned two-digit decimal number in the  
85306 range 00 through 59. It represents the timezone of the timestamp as the number of hours (hh)  
85307 and minutes (mm) east (+) or west (-) of UTC for the timestamp. If the hours and minutes are  
85308 both zero, the sign shall be '+'. However, if the timezone is not an integral number of minutes

85309 away from UTC, the `<zone>` field is implementation-defined.

85310 Then, the following output formats shall be applied for every set of changes.

85311 First, the range of lines in each file shall be written in the following format:

85312 `"@@Δ-%sΔ+%sΔ@@", <file1 range>, <file2 range>`

85313 Each `<range>` field shall be of the form:

85314 `"%ld", <beginning line number>`

85315 if the range contains exactly one line, and:

85316 `"%ld,%ld", <beginning line number>, <number of lines>`

85317 otherwise. If a range is empty, its beginning line number shall be the number of the line just

85318 before the range, or 0 if the empty range starts the file.

85319 Next, the affected lines along with lines of context shall be written. Each non-empty unaffected

85320 line shall be written in the following format:

85321 `"Δ%s", <unaffected_line>`

85322 where the contents of the unaffected line shall be taken from *file1*. It is implementation-defined

85323 whether an empty unaffected line is written as an empty line or a line containing a single

85324 `<space>` character. This line also represents the same line of *file2*, even though *file2*'s line may

85325 contain different contents due to the `-b`. Deleted lines shall be written as:

85326 `"- %s", <deleted_line>`

85327 Added lines shall be written as:

85328 `"+ %s", <added_line>`

85329 The order of lines written shall be the same as that of the corresponding file. A deleted line shall

85330 never be written immediately after an added line.

85331 If `-U n` is specified, the output shall contain no more than *n* consecutive unaffected lines; and if

85332 the output contains an affected line and this line is adjacent to up to *n* consecutive unaffected

85333 lines in the corresponding file, the output shall contain these unaffected lines. `-u` shall act like

85334 `-U3`.

85335 **STDERR**

85336 The standard error shall be used only for diagnostic messages.

85337 **OUTPUT FILES**

85338 None.

85339 **EXTENDED DESCRIPTION**

85340 None.

85341 **EXIT STATUS**

85342 The following exit values shall be returned:

85343 0 No differences were found.

85344 1 Differences were found.

85345 >1 An error occurred.

## CONSEQUENCES OF ERRORS

Default.

## APPLICATION USAGE

If lines at the end of a file are changed and other lines are added, *diff* output may show this as a delete and add, as a change, or as a change and add; *diff* is not expected to know which happened and users should not care about the difference in output as long as it clearly shows the differences between the files.

## EXAMPLES

If **dir1** is a directory containing a directory named **x**, **dir2** is a directory containing a directory named **x**, **dir1/x** and **dir2/x** both contain files named **date.out**, and **dir2/x** contains a file named **y**, the command:

```
diff -r dir1 dir2
```

could produce output similar to:

```
Common subdirectories: dir1/x and dir2/x
Only in dir2/x: y
diff -r dir1/x/date.out dir2/x/date.out
1c1
< Mon Jul 2 13:12:16 PDT 1990

> Tue Jun 19 21:41:39 PDT 1990
```

## RATIONALE

The **-h** option was omitted because it was insufficiently specified and does not add to applications portability.

Historical implementations employ algorithms that do not always produce a minimum list of differences; the current language about making every effort is the best this volume of POSIX.1-2008 can do, as there is no metric that could be employed to judge the quality of implementations against any and all file contents. The statement “This list should be minimal” clearly implies that implementations are not expected to provide the following output when comparing two 100-line files that differ in only one character on a single line:

```
1,100c1,100
all 100 lines from file1 preceded with "< "

all 100 lines from file2 preceded with "> "
```

The “Only in” messages required when the **-r** option is specified are not used by most historical implementations if the **-e** option is also specified. It is required here because it provides useful information that must be provided to update a target directory hierarchy to match a source hierarchy. The “Common subdirectories” messages are written by System V and 4.3 BSD when the **-r** option is specified. They are allowed here but are not required because they are reporting on something that is the same, not reporting a difference, and are not needed to update a target hierarchy.

The **-c** option, which writes output in a format using lines of context, has been included. The format is useful for a variety of reasons, among them being much improved readability and the ability to understand difference changes when the target file has line numbers that differ from another similar, but slightly different, copy. The *patch* utility is most valuable when working with difference listings using a context format. The BSD version of **-c** takes an optional argument specifying the amount of context. Rather than overloading **-c** and breaking the Utility Syntax Guidelines for *diff*, the standard developers decided to add a separate option for

specifying a context diff with a specified amount of context (`-C`). Also, the format for context *diffs* was extended slightly in 4.3 BSD to allow multiple changes that are within context lines from each other to be merged together. The output format contains an additional four `<asterisk>` characters after the range of affected lines in the first filename. This was to provide a flag for old programs (like old versions of *patch*) that only understand the old context format. The version of context described here does not require that multiple changes within context lines be merged, but it does not prohibit it either. The extension is upwards-compatible, so any vendors that wish to retain the old version of *diff* can do so by adding the extra four `<asterisk>` characters (that is, utilities that currently use *diff* and understand the new merged format will also understand the old unmerged format, but not *vice versa*).

The `-u` and `-U` options of GNU *diff* have been included. Their output format, designed by Wayne Davison, takes up less space than `-c` and `-C` format, and in many cases is easier to read. The format's timestamps do not vary by locale, so `LC_TIME` does not affect it. The format's line numbers are rendered with the `%ld` format, not `%d`, because the file format notation rules would allow extra `<blank>` characters to appear around the numbers.

The substitute command was added as an additional format for the `-e` option. This was added to provide implementations with a way to fix the classic "dot alone on a line" bug present in many versions of *diff*. Since many implementations have fixed this bug, the standard developers decided not to standardize broken behavior, but rather to provide the necessary tool for fixing the bug. One way to fix this bug is to output two periods whenever a lone period is needed, then terminate the append command with a period, and then use the substitute command to convert the two periods into one period.

The BSD-derived `-r` option was added to provide a mechanism for using *diff* to compare two file system trees. This behavior is useful, is standard practice on all BSD-derived systems, and is not easily reproducible with the *find* utility.

The requirement that *diff* not compare files in some circumstances, even though they have the same name, is based on the actual output of historical implementations. The specified behavior precludes the problems arising from running into FIFOs and other files that would cause *diff* to hang waiting for input with no indication to the user that *diff* was hung. An earlier version of this standard specified the output format more precisely, but in practice this requirement was widely ignored and the benefit of standardization seemed small, so it is now unspecified. In most common usage, *diff -r* should indicate differences in the file hierarchies, not the difference of contents of devices pointed to by the hierarchies.

Many early implementations of *diff* require seekable files. Since the System Interfaces volume of POSIX.1-2008 supports named pipes, the standard developers decided that such a restriction was unreasonable. Note also that the allowed filename – almost always refers to a pipe.

No directory search order is specified for *diff*. The historical ordering is, in fact, not optimal, in that it prints out all of the differences at the current level, including the statements about all common subdirectories before recursing into those subdirectories.

The message:

```
"diff %s %s %s\n", <diff_options>, <filename1>, <filename2>
```

does not vary by locale because it is the representation of a command, not an English sentence.

## FUTURE DIRECTIONS

None.

85437 **SEE ALSO**85438 *cmp, comm, ed, find*85439 XBD [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 215)85440 **CHANGE HISTORY**

85441 First released in Issue 2.

85442 **Issue 5**

85443 The FUTURE DIRECTIONS section is added.

85444 **Issue 6**85445 The following new requirements on POSIX implementations derive from alignment with the  
85446 Single UNIX Specification:

- 85447
- The `-f` option is added.

85448 The output format for `-c` or `-C` format is changed to align with changes to the IEEE P1003.2b  
85449 draft standard resulting from IEEE PASC Interpretation 1003.2 #71.

85450 The normative text is reworded to avoid use of the term “must” for application requirements.

85451 IEEE Std 1003.1-2001/Cor 1-2002, item XCU/TC1/D6/20 is applied, changing the STDOUT  
85452 section. This changes the specification of *diff* `-c` so that it agrees with existing practice when  
85453 contexts contain zero lines or one line.85454 **Issue 7**

85455 Austin Group Interpretations 1003.1-2001 #115 and #114 are applied.

85456 Austin Group Interpretation 1003.1-2001 #192 is applied, clarifying the behavior if both files are  
85457 non-text files.

85458 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

85459 SD5-XCU-ERN-103 and SD5-XCU-ERN-120 are applied, adding the `-u` option.

**NAME**

dirname — return the directory portion of a pathname

**SYNOPSIS**

dirname *string*

**DESCRIPTION**

The *string* operand shall be treated as a pathname, as defined in XBD [Section 3.267](#) (on page 75). The string *string* shall be converted to the name of the directory containing the filename corresponding to the last pathname component in *string*, performing actions equivalent to the following steps in order:

1. If *string* is *//*, skip steps 2 to 5.
2. If *string* consists entirely of <slash> characters, *string* shall be set to a single <slash> character. In this case, skip steps 3 to 8.
3. If there are any trailing <slash> characters in *string*, they shall be removed.
4. If there are no <slash> characters remaining in *string*, *string* shall be set to a single <period> character. In this case, skip steps 5 to 8.
5. If there are any trailing non-<slash> characters in *string*, they shall be removed.
6. If the remaining *string* is *//*, it is implementation-defined whether steps 7 and 8 are skipped or processed.
7. If there are any trailing <slash> characters in *string*, they shall be removed.
8. If the remaining *string* is empty, *string* shall be set to a single <slash> character.

The resulting string shall be written to standard output.

**OPTIONS**

None.

**OPERANDS**

The following operand shall be supported:

*string*            A string.

**STDIN**

Not used.

**INPUT FILES**

None.

**ENVIRONMENT VARIABLES**

The following environment variables shall affect the execution of *dirname*:

- |                 |                                                                                                                                                                                                                                                    |
|-----------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>LANG</b>     | Provide a default value for the internationalization variables that are unset or null. (See XBD <a href="#">Section 8.2</a> (on page 174) for the precedence of internationalization variables used to determine the values of locale categories.) |
| <b>LC_ALL</b>   | If set to a non-empty string value, override the values of all the other internationalization variables.                                                                                                                                           |
| <b>LC_CTYPE</b> | Determine the locale for the interpretation of sequences of bytes of text data as characters (for example, single-byte as opposed to multi-byte characters in arguments).                                                                          |

85500 **LC\_MESSAGES**  
 85501 Determine the locale that should be used to affect the format and contents of  
 85502 diagnostic messages written to standard error.

85503 XSI **NLSPATH** Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

85504 **ASYNCHRONOUS EVENTS**  
 85505 Default.

85506 **STDOUT**  
 85507 The *dirname* utility shall write a line to the standard output in the following format:  
 85508 "%s\n", <resulting string>

85509 **STDERR**  
 85510 The standard error shall be used only for diagnostic messages.

85511 **OUTPUT FILES**  
 85512 None.

85513 **EXTENDED DESCRIPTION**  
 85514 None.

85515 **EXIT STATUS**  
 85516 The following exit values shall be returned:  
 85517 0 Successful completion.  
 85518 >0 An error occurred.

85519 **CONSEQUENCES OF ERRORS**  
 85520 Default.

85521 **APPLICATION USAGE**  
 85522 The definition of *pathname* specifies implementation-defined behavior for pathnames starting  
 85523 with two <slash> characters. Therefore, applications shall not arbitrarily add <slash> characters  
 85524 to the beginning of a pathname unless they can ensure that there are more or less than two or are  
 85525 prepared to deal with the implementation-defined consequences.

## 85526 EXAMPLES

| Command                 | Results     |
|-------------------------|-------------|
| <i>dirname</i> /        | /           |
| <i>dirname</i> //       | / or //     |
| <i>dirname</i> /a/b/    | /a          |
| <i>dirname</i> //a//b// | //a         |
| <i>dirname</i>          | Unspecified |
| <i>dirname</i> a        | . (\$? = 0) |
| <i>dirname</i> ""       | . (\$? = 0) |
| <i>dirname</i> /a       | /           |
| <i>dirname</i> /a/b     | /a          |
| <i>dirname</i> a/b      | a           |

85538 See also the examples for the *basename* utility.

+

85539 **RATIONALE**  
 85540 The *dirname* utility originated in System III. It has evolved through the System V releases to a  
 85541 version that matches the requirements specified in this description in System V Release 3. 4.3  
 85542 BSD and earlier versions did not include *dirname*.

85543 The behaviors of *basename* and *dirname* in this volume of POSIX.1-2008 have been coordinated so  
85544 that when *string* is a valid pathname:

85545 `$ (basename -- "string")` |

85546 would be a valid filename for the file in the directory:

85547 `$ (dirname -- "string")` |

85548 This would not work for the versions of these utilities in early proposals due to the way  
85549 processing of trailing <slash> characters was specified. Consideration was given to leaving  
85550 processing unspecified if there were trailing <slash> characters, but this cannot be done; XBD  
85551 [Section 3.267](#) (on page 75) allows trailing <slash> characters. The *basename* and *dirname* utilities  
85552 have to specify consistent handling for all valid pathnames.

#### 85553 FUTURE DIRECTIONS

85554 None.

#### 85555 SEE ALSO

85556 [Section 2.5](#) (on page 2323), *basename*

85557 XBD [Section 3.267](#) (on page 75), [Chapter 8](#) (on page 173)

#### 85558 CHANGE HISTORY

85559 First released in Issue 2.

#### 85560 Issue 7

85561 POSIX.1-2008, Technical Corrigendum 1, XCU/TC1-2008/0083 [192,430], XCU/TC1-2008/0084 +  
85562 [192], and XCU/TC1-2008/0085 [192] are applied.



85563 **NAME**85564 `du` — estimate file space usage85565 **SYNOPSIS**85566 `du [-a|-s] [-kx] [-H|-L] [file...]`85567 **DESCRIPTION**

85568 By default, the *du* utility shall write to standard output the size of the file space allocated to, and  
 85569 the size of the file space allocated to each subdirectory of, the file hierarchy rooted in each of the  
 85570 specified files. By default, when a symbolic link is encountered on the command line or in the  
 85571 file hierarchy, *du* shall count the size of the symbolic link (rather than the file referenced by the  
 85572 link), and shall not follow the link to another portion of the file hierarchy. The size of the file  
 85573 space allocated to a file of type directory shall be defined as the sum total of space allocated to  
 85574 all files in the file hierarchy rooted in the directory plus the space allocated to the directory itself.

85575 When *du* cannot *stat()* files or *stat()* or read directories, it shall report an error condition and the  
 85576 final exit status is affected. Files with multiple links shall be counted and written for only one  
 85577 entry. The directory entry that is selected in the report is unspecified. By default, file sizes shall  
 85578 be written in 512-byte units, rounded up to the next 512-byte unit.

85579 **OPTIONS**85580 The *du* utility shall conform to XBD [Section 12.2](#) (on page 215).

85581 The following options shall be supported:

85582 **-a** In addition to the default output, report the size of each file not of type directory in  
 85583 the file hierarchy rooted in the specified file. Regardless of the presence of the **-a**  
 85584 option, non-directories given as *file* operands shall always be listed.

85585 **-H** If a symbolic link is specified on the command line, *du* shall count the size of the  
 85586 file or file hierarchy referenced by the link.

85587 **-k** Write the files sizes in units of 1 024 bytes, rather than the default 512-byte units.

85588 **-L** If a symbolic link is specified on the command line or encountered during the  
 85589 traversal of a file hierarchy, *du* shall count the size of the file or file hierarchy  
 85590 referenced by the link.

85591 **-s** Instead of the default output, report only the total sum for each of the specified  
 85592 files.

85593 **-x** When evaluating file sizes, evaluate only those files that have the same device as  
 85594 the file specified by the *file* operand.

85595 Specifying more than one of the mutually-exclusive options **-H** and **-L** shall not be considered  
 85596 an error. The last option specified shall determine the behavior of the utility.

85597 **OPERANDS**

85598 The following operand shall be supported:

85599 *file* The pathname of a file whose size is to be written. If no *file* is specified, the current  
 85600 directory shall be used.

85601 **STDIN**

85602 Not used.

85603 **INPUT FILES**

85604 None.

85605 **ENVIRONMENT VARIABLES**85606 The following environment variables shall affect the execution of *du*:

85607 *LANG* Provide a default value for the internationalization variables that are unset or null.  
 85608 (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 85609 variables used to determine the values of locale categories.)

85610 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
 85611 internationalization variables.

85612 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
 85613 characters (for example, single-byte as opposed to multi-byte characters in  
 85614 arguments).

85615 *LC\_MESSAGES*

85616 Determine the locale that should be used to affect the format and contents of  
 85617 diagnostic messages written to standard error.

85618 *XSI* *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

85619 **ASYNCHRONOUS EVENTS**

85620 Default.

85621 **STDOUT**

85622 The output from *du* shall consist of the amount of space allocated to a file and the name of the  
 85623 file, in the following format:

85624 "%d %s\n", &lt;size&gt;, &lt;pathname&gt;

85625 **STDERR**

85626 The standard error shall be used only for diagnostic messages.

85627 **OUTPUT FILES**

85628 None.

85629 **EXTENDED DESCRIPTION**

85630 None.

85631 **EXIT STATUS**

85632 The following exit values shall be returned:

85633 0 Successful completion.

85634 &gt;0 An error occurred.

85635 **CONSEQUENCES OF ERRORS**

85636 Default.

85637 **APPLICATION USAGE**

85638 None.

85639 **EXAMPLES**

85640 None.

85641 **RATIONALE**

85642 The use of 512-byte units is historical practice and maintains compatibility with *ls* and other  
 85643 utilities in this volume of POSIX.1-2008. This does not mandate that the file system itself be  
 85644 based on 512-byte blocks. The *-k* option was added as a compromise measure. It was agreed by  
 85645 the standard developers that 512 bytes was the best default unit because of its complete  
 85646 historical consistency on System V (*versus* the mixed 512/1024-byte usage on BSD systems), and  
 85647 that a *-k* option to switch to 1024-byte units was a good compromise. Users who prefer the

1 024-byte quantity can easily alias *du* to *du -k* without breaking the many historical scripts relying on the 512-byte units.

The **-b** option was added to an early proposal to provide a resolution to the situation where System V and BSD systems give figures for file sizes in *blocks*, which is an implementation-defined concept. (In common usage, the block size is 512 bytes for System V and 1 024 bytes for BSD systems.) However, **-b** was later deleted, since the default was eventually decided as 512-byte units.

Historical file systems provided no way to obtain exact figures for the space allocation given to files. There are two known areas of inaccuracies in historical file systems: cases of *indirect blocks* being used by the file system or *sparse* files yielding incorrectly high values. An indirect block is space used by the file system in the storage of the file, but that need not be counted in the space allocated to the file. A *sparse* file is one in which an *lseek()* call has been made to a position beyond the end of the file and data has subsequently been written at that point. A file system need not allocate all the intervening zero-filled blocks to such a file. It is up to the implementation to define exactly how accurate its methods are.

The **-a** and **-s** options were mutually-exclusive in the original version of *du*. The POSIX Shell and Utilities description is implied by the language in the SVID where **-s** is described as causing “only the grand total” to be reported. Some systems may produce output for **-sa**, but a Strictly Conforming POSIX Shell and Utilities Application cannot use that combination.

The **-a** and **-s** options were adopted from the SVID except that the System V behavior of not listing non-directories explicitly given as operands, unless the **-a** option is specified, was considered a bug; the BSD-based behavior (report for all operands) is mandated. The default behavior of *du* in the SVID with regard to reporting the failure to read files (it produces no messages) was considered counter-intuitive, and thus it was specified that the POSIX Shell and Utilities default behavior shall be to produce such messages. These messages can be turned off with shell redirection to achieve the System V behavior.

The **-x** option is historical practice on recent BSD systems. It has been adopted by this volume of POSIX.1-2008 because there was no other historical method of limiting the *du* search to a single file hierarchy. This limitation of the search is necessary to make it possible to obtain file space usage information about a file system on which other file systems are mounted, without having to resort to a lengthy *find* and *awk* script.

## FUTURE DIRECTIONS

None.

## SEE ALSO

*ls*

XBD [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 215)

XSH [fstatat\(\)](#)

## CHANGE HISTORY

First released in Issue 2.

## Issue 6

This utility is marked as part of the User Portability Utilities option.

The APPLICATION USAGE section is added.

The obsolescent **-r** option is removed.

The Open Group Corrigendum U025/3 is applied. The *du* utility is reinstated, as it had incorrectly been marked LEGACY in Issue 5.

85693 The **-H** and **-L** options for symbolic links are added as described in the IEEE P1003.2b draft  
85694 standard.

85695 **Issue 7**

85696 The *du* utility is moved from the User Portability Utilities option to the Base. User Portability  
85697 Utilities is now an option for interactive utilities.

85698 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

85699 **NAME**85700 `echo` — write arguments to standard output85701 **SYNOPSIS**85702 `echo` [*string*...]85703 **DESCRIPTION**85704 The *echo* utility writes its arguments to standard output, followed by a <newline>. If there are  
85705 no arguments, only the <newline> is written.85706 **OPTIONS**85707 The *echo* utility shall not recognize the "--" argument in the manner specified by Guideline 10  
85708 of XBD [Section 12.2](#) (on page 215); "--" shall be recognized as a string operand.

85709 Implementations shall not support any options.

85710 **OPERANDS**

85711 The following operands shall be supported:

85712 *string* A string to be written to standard output. If the first operand is `-n`, or if any of the  
85713 operands contain a <backslash> character, the results are implementation-defined.85714 XSI On XSI-conformant systems, if the first operand is `-n`, it shall be treated as a string,  
85715 not an option. The following character sequences shall be recognized on XSI-  
85716 conformant systems within any of the arguments:

|       |                    |                                                                                    |
|-------|--------------------|------------------------------------------------------------------------------------|
| 85717 | <code>\a</code>    | Write an <alert>.                                                                  |
| 85718 | <code>\b</code>    | Write a <backspace>.                                                               |
| 85719 | <code>\c</code>    | Suppress the <newline> that otherwise follows the final argument in the            |
| 85720 |                    | output. All characters following the ' <code>\c</code> ' in the arguments shall be |
| 85721 |                    | ignored.                                                                           |
| 85722 | <code>\f</code>    | Write a <form-feed>.                                                               |
| 85723 | <code>\n</code>    | Write a <newline>.                                                                 |
| 85724 | <code>\r</code>    | Write a <carriage-return>.                                                         |
| 85725 | <code>\t</code>    | Write a <tab>.                                                                     |
| 85726 | <code>\v</code>    | Write a <vertical-tab>.                                                            |
| 85727 | <code>\\</code>    | Write a <backslash> character.                                                     |
| 85728 | <code>\0num</code> | Write an 8-bit value that is the zero, one, two, or three-digit octal number       |
| 85729 |                    | <i>num</i> .                                                                       |

85730 **STDIN**

85731 Not used.

85732 **INPUT FILES**

85733 None.

85734 **ENVIRONMENT VARIABLES**85735 The following environment variables shall affect the execution of *echo*:85736 *LANG* Provide a default value for the internationalization variables that are unset or null.  
85737 (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
85738 variables used to determine the values of locale categories.)

85739 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
 85740 internationalization variables.

85741 XSI *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
 85742 characters (for example, single-byte as opposed to multi-byte characters in  
 85743 arguments).

85744 *LC\_MESSAGES*  
 85745 Determine the locale that should be used to affect the format and contents of  
 85746 diagnostic messages written to standard error.

85747 XSI *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

85748 **ASYNCHRONOUS EVENTS**  
 85749 Default.

85750 **STDOUT**  
 85751 The *echo* utility arguments shall be separated by single <space> characters and a <newline>  
 85752 XSI character shall follow the last argument. Output transformations shall occur based on the  
 85753 escape sequences in the input. See the OPERANDS section.

85754 **STDERR**  
 85755 The standard error shall be used only for diagnostic messages.

85756 **OUTPUT FILES**  
 85757 None.

85758 **EXTENDED DESCRIPTION**  
 85759 None.

85760 **EXIT STATUS**  
 85761 The following exit values shall be returned:  
 85762 0 Successful completion.  
 85763 >0 An error occurred.

85764 **CONSEQUENCES OF ERRORS**  
 85765 Default.

85766 **APPLICATION USAGE**  
 85767 It is not possible to use *echo* portably across all POSIX systems unless both *-n* (as the first  
 85768 argument) and escape sequences are omitted.

85769 The *printf* utility can be used portably to emulate any of the traditional behaviors of the *echo*  
 85770 utility as follows (assuming that *IFS* has its standard value or is unset):

- 85771 • The historic System V *echo* and the requirements on XSI implementations in this volume of  
 85772 POSIX.1-2008 are equivalent to:

```
85773 printf "%b\n" "$*"
85774
```

- 85774 • The BSD *echo* is equivalent to:

```
85775 if ["X$1" = "X-n"]
85776 then
85777 shift
85778 printf "%s" "$*"
85779 else
85780 printf "%s\n" "$*"
85781 fi
```

85782 New applications are encouraged to use *printf* instead of *echo*.

85783 **EXAMPLES**

85784 None.

85785 **RATIONALE**

85786 The *echo* utility has not been made obsolescent because of its extremely widespread use in  
85787 historical applications. Conforming applications that wish to do prompting without <newline>  
85788 characters or that could possibly be expecting to echo a **-n**, should use the *printf* utility derived  
85789 from the Ninth Edition system.

85790 As specified, *echo* writes its arguments in the simplest of ways. The two different historical  
85791 versions of *echo* vary in fatally incompatible ways.

85792 The BSD *echo* checks the first argument for the string **-n** which causes it to suppress the  
85793 <newline> that would otherwise follow the final argument in the output.

85794 The System V *echo* does not support any options, but allows escape sequences within its  
85795 operands, as described for XSI implementations in the OPERANDS section.

85796 The *echo* utility does not support Utility Syntax Guideline 10 because historical applications  
85797 depend on *echo* to echo *all* of its arguments, except for the **-n** option in the BSD version.

85798 **FUTURE DIRECTIONS**

85799 None.

85800 **SEE ALSO**

85801 *printf*

85802 XBD [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 215)

85803 **CHANGE HISTORY**

85804 First released in Issue 2.

85805 **Issue 5**

85806 In the OPTIONS section, the last sentence is changed to indicate that implementations “do not”  
85807 support any options; in the previous issue this said “need not”.

85808 **Issue 6**

85809 The following new requirements on POSIX implementations derive from alignment with the  
85810 Single UNIX Specification:

- 85811 • A set of character sequences is defined as *string* operands.
- 85812 • *LC\_CTYPE* is added to the list of environment variables affecting *echo*.
- 85813 • In the OPTIONS section, implementations shall not support any options.

85814 IEEE Std 1003.1-2001/Cor 1-2002, item XCU/TC1/D6/21 is applied, so that the *echo* utility can  
85815 accommodate historical BSD behavior.

85816 **Issue 7**

85817 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

85818 **NAME**

85819           ed — edit text

85820 **SYNOPSIS**85821           ed [-p *string*] [-s] [*file*]85822 **DESCRIPTION**

85823           The *ed* utility is a line-oriented text editor that uses two modes: *command mode* and *input mode*. In  
 85824           command mode the input characters shall be interpreted as commands, and in input mode they  
 85825           shall be interpreted as text. See the EXTENDED DESCRIPTION section.

85826           If an operand is *'-'*, the results are unspecified.85827 **OPTIONS**

85828           The *ed* utility shall conform to XBD [Section 12.2](#) (on page 215), except for the unspecified usage  
 85829           of *'-'*.

85830           The following options shall be supported:

85831           -p *string*    Use *string* as the prompt string when in command mode. By default, there shall be  
 85832                           no prompt string.

85833           -s            Suppress the writing of byte counts by **e**, **E**, **r**, and **w** commands and of the *'!'*  
 85834                           prompt after a *!command*.

85835 **OPERANDS**

85836           The following operand shall be supported:

85837           *file*        If the *file* argument is given, *ed* shall simulate an **e** command on the file named by  
 85838                           the pathname, *file*, before accepting commands from the standard input.

85839 **STDIN**

85840           The standard input shall be a text file consisting of commands, as described in the EXTENDED  
 85841           DESCRIPTION section.

85842 **INPUT FILES**

85843           The input files shall be text files.

85844 **ENVIRONMENT VARIABLES**85845           The following environment variables shall affect the execution of *ed*:

85846           HOME        Determine the pathname of the user's home directory.

85847           LANG        Provide a default value for the internationalization variables that are unset or null.  
 85848                           (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 85849                           variables used to determine the values of locale categories.)

85850           LC\_ALL       If set to a non-empty string value, override the values of all the other  
 85851                           internationalization variables.

85852           LC\_COLLATE

85853                           Determine the locale for the behavior of ranges, equivalence classes, and multi-  
 85854                           character collating elements within regular expressions.

85855           LC\_CTYPE    Determine the locale for the interpretation of sequences of bytes of text data as  
 85856                           characters (for example, single-byte as opposed to multi-byte characters in  
 85857                           arguments and input files) and the behavior of character classes within regular  
 85858                           expressions.



|       |                             |                                                                                                                                                                         |
|-------|-----------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 85859 | <i>LC_MESSAGES</i>          |                                                                                                                                                                         |
| 85860 |                             | Determine the locale that should be used to affect the format and contents of                                                                                           |
| 85861 |                             | diagnostic messages written to standard error and informative messages written to                                                                                       |
| 85862 |                             | standard output.                                                                                                                                                        |
| 85863 | XSI                         | <i>NLSPATH</i> Determine the location of message catalogs for the processing of <i>LC_MESSAGES</i> .                                                                    |
| 85864 | <b>ASYNCHRONOUS EVENTS</b>  |                                                                                                                                                                         |
| 85865 |                             | The <i>ed</i> utility shall take the standard action for all signals (see the ASYNCHRONOUS EVENTS                                                                       |
| 85866 |                             | section in <a href="#">Section 1.4</a> , on page 2310) with the following exceptions:                                                                                   |
| 85867 | SIGINT                      | The <i>ed</i> utility shall interrupt its current activity, write the string "?\n" to standard                                                                          |
| 85868 |                             | output, and return to command mode (see the EXTENDED DESCRIPTION                                                                                                        |
| 85869 |                             | section).                                                                                                                                                               |
| 85870 | SIGHUP                      | If the buffer is not empty and has changed since the last write, the <i>ed</i> utility shall                                                                            |
| 85871 |                             | attempt to write a copy of the buffer in a file. First, the file named <b>ed.hup</b> in the                                                                             |
| 85872 |                             | current directory shall be used; if that fails, the file named <b>ed.hup</b> in the directory                                                                           |
| 85873 |                             | named by the <i>HOME</i> environment variable shall be used. In any case, the <i>ed</i> utility                                                                         |
| 85874 |                             | shall exit without writing the file to the currently remembered pathname and                                                                                            |
| 85875 |                             | without returning to command mode.                                                                                                                                      |
| 85876 | SIGQUIT                     | The <i>ed</i> utility shall ignore this event.                                                                                                                          |
| 85877 | <b>STDOUT</b>               |                                                                                                                                                                         |
| 85878 |                             | Various editing commands and the prompting feature (see <b>-p</b> ) write to standard output, as                                                                        |
| 85879 |                             | described in the EXTENDED DESCRIPTION section.                                                                                                                          |
| 85880 | <b>STDERR</b>               |                                                                                                                                                                         |
| 85881 |                             | The standard error shall be used only for diagnostic messages.                                                                                                          |
| 85882 | <b>OUTPUT FILES</b>         |                                                                                                                                                                         |
| 85883 |                             | The output files shall be text files whose formats are dependent on the editing commands given.                                                                         |
| 85884 | <b>EXTENDED DESCRIPTION</b> |                                                                                                                                                                         |
| 85885 |                             | The <i>ed</i> utility shall operate on a copy of the file it is editing; changes made to the copy shall have                                                            |
| 85886 |                             | no effect on the file until a <b>w</b> (write) command is given. The copy of the text is called the <i>buffer</i> .                                                     |
| 85887 |                             | Commands to <i>ed</i> have a simple and regular structure: zero, one, or two <i>addresses</i> followed by a                                                             |
| 85888 |                             | single-character <i>command</i> , possibly followed by parameters to that command. These addresses                                                                      |
| 85889 |                             | specify one or more lines in the buffer. Every command that requires addresses has default                                                                              |
| 85890 |                             | addresses, so that the addresses very often can be omitted. If the <b>-p</b> option is specified, the                                                                   |
| 85891 |                             | prompt string shall be written to standard output before each command is read.                                                                                          |
| 85892 |                             | In general, only one command can appear on a line. Certain commands allow text to be input.                                                                             |
| 85893 |                             | This text is placed in the appropriate place in the buffer. While <i>ed</i> is accepting text, it is said to be                                                         |
| 85894 |                             | in <i>input mode</i> . In this mode, no commands shall be recognized; all input is merely collected.                                                                    |
| 85895 |                             | Input mode is terminated by entering a line consisting of two characters: a <period> ('.') followed by a <newline>. This line is not considered part of the input text. |
| 85896 |                             |                                                                                                                                                                         |

## Regular Expressions in ed

The *ed* utility shall support basic regular expressions, as described in XBD [Section 9.3](#) (on page 183). Since regular expressions in *ed* are always matched against single lines (excluding the terminating <newline> characters), never against any larger section of text, there is no way for a regular expression to match a <newline>.

A null RE shall be equivalent to the last RE encountered.

Regular expressions are used in addresses to specify lines, and in some commands (for example, the **s** substitute command) to specify portions of a line to be substituted.

## Addresses in ed

Addressing in *ed* relates to the current line. Generally, the current line is the last line affected by a command. The current line number is the address of the current line. If the edit buffer is not empty, the initial value for the current line shall be the last line in the edit buffer; otherwise, zero.

Addresses shall be constructed as follows:

1. The <period> character ( ' . ' ) shall address the current line.
2. The <dollar-sign> character ( ' \$ ' ) shall address the last line of the edit buffer.
3. The positive decimal number *n* shall address the *n*th line of the edit buffer.
4. The <apostrophe>-*x* character pair ( " ' *x* " ) shall address the line marked with the mark name character *x*, which shall be a lowercase letter from the portable character set. It shall be an error if the character has not been set to mark a line or if the line that was marked is not currently present in the edit buffer.
5. A BRE enclosed by <slash> characters ( ' / ' ) shall address the first line found by searching forwards from the line following the current line toward the end of the edit buffer and stopping at the first line for which the line excluding the terminating <newline> matches the BRE. The BRE consisting of a null BRE delimited by a pair of <slash> characters shall address the next line for which the line excluding the terminating <newline> matches the last BRE encountered. In addition, the second <slash> can be omitted at the end of a command line. Within the BRE, a <backslash>-<slash> pair ( " \ / " ) shall represent a literal <slash> instead of the BRE delimiter. If necessary, the search shall wrap around to the beginning of the buffer and continue up to and including the current line, so that the entire buffer is searched.
6. A BRE enclosed by <question-mark> characters ( ' ? ' ) shall address the first line found by searching backwards from the line preceding the current line toward the beginning of the edit buffer and stopping at the first line for which the line excluding the terminating <newline> matches the BRE. The BRE consisting of a null BRE delimited by a pair of <question-mark> characters ( " ? ? " ) shall address the previous line for which the line excluding the terminating <newline> matches the last BRE encountered. In addition, the second <question-mark> can be omitted at the end of a command line. Within the BRE, a <backslash>-<question-mark> pair ( " \ ? " ) shall represent a literal <question-mark> instead of the BRE delimiter. If necessary, the search shall wrap around to the end of the buffer and continue up to and including the current line, so that the entire buffer is searched.
7. A <plus-sign> ( ' + ' ) or <hyphen> character ( ' - ' ) followed by a decimal number shall address the current line plus or minus the number. A <plus-sign> or <hyphen> character not followed by a decimal number shall address the current line plus or minus 1.

Addresses can be followed by zero or more address offsets, optionally <blank>-separated. Address offsets are constructed as follows:

- A <plus-sign> or <hyphen> character followed by a decimal number shall add or subtract, respectively, the indicated number of lines to or from the address. A <plus-sign> or <hyphen> character not followed by a decimal number shall add or subtract 1 to or from the address.
- A decimal number shall add the indicated number of lines to the address.

It shall not be an error for an intermediate address value to be less than zero or greater than the last line in the edit buffer. It shall be an error for the final address value to be less than zero or greater than the last line in the edit buffer. It shall be an error if a search for a BRE fails to find a matching line.

Commands accept zero, one, or two addresses. If more than the required number of addresses are provided to a command that requires zero addresses, it shall be an error. Otherwise, if more than the required number of addresses are provided to a command, the addresses specified first shall be evaluated and then discarded until the maximum number of valid addresses remain, for the specified command.

Addresses shall be separated from each other by a <comma> (',' ) or <semicolon> character (';'). In the case of a <semicolon> separator, the current line ('.') shall be set to the first address, and only then will the second address be calculated. This feature can be used to determine the starting line for forwards and backwards searches; see rules 5. and 6.

Addresses can be omitted on either side of the <comma> or <semicolon> separator, in which case the resulting address pairs shall be as follows:

| Specified | Resulting   |
|-----------|-------------|
| ,         | 1 , \$      |
| , addr    | 1 , addr    |
| addr ,    | addr , addr |
| ;         | . ; \$      |
| ; addr    | . ; addr    |
| addr ;    | addr ; addr |

Any <blank> characters included between addresses, address separators, or address offsets shall be ignored.

### Commands in ed

In the following list of *ed* commands, the default addresses are shown in parentheses. The number of addresses shown in the default shall be the number expected by the command. The parentheses are not part of the address; they show that the given addresses are the default.

It is generally invalid for more than one command to appear on a line. However, any command (except **e**, **E**, **f**, **q**, **Q**, **r**, **w**, and **!**) can be suffixed by the letter **l**, **n**, or **p**; in which case, except for the **l**, **n**, and **p** commands, the command shall be executed and then the new current line shall be written as described below under the **l**, **n**, and **p** commands. When an **l**, **n**, or **p** suffix is used with an **l**, **n**, or **p** command, the command shall write to standard output as described below, but it is unspecified whether the suffix writes the current line again in the requested format or whether the suffix has no effect. For example, the **pl** command (base **p** command with an **l** suffix) shall either write just the current line or write it twice—once as specified for **p** and once as specified for **l**. Also, the **g**, **G**, **v**, and **V** commands shall take a command as a parameter.

Each address component can be preceded by zero or more <blank> characters. The command

letter can be preceded by zero or more <blank> characters. If a suffix letter (**l**, **n**, or **p**) is given, the application shall ensure that it immediately follows the command.

The **e**, **E**, **f**, **r**, and **w** commands shall take an optional *file* parameter, separated from the command letter by one or more <blank> characters.

If changes have been made in the buffer since the last **w** command that wrote the entire buffer, *ed* shall warn the user if an attempt is made to destroy the editor buffer via the **e** or **q** commands. The *ed* utility shall write the string:

"?\n"

(followed by an explanatory message if *help mode* has been enabled via the **H** command) to standard output and shall continue in command mode with the current line number unchanged. If the **e** or **q** command is repeated with no intervening command, it shall take effect.

If a terminal disconnect (see XBD [Chapter 11](#) (on page 199), Modem Disconnect and Closing a Device Terminal), is detected:

- If accompanied by a SIGHUP signal, the *ed* utility shall operate as described in the ASYNCHRONOUS EVENTS section for a SIGHUP signal.
- If not accompanied by a SIGHUP signal, the *ed* utility shall act as if an end-of-file had been detected on standard input.

If an end-of-file is detected on standard input:

- If the *ed* utility is in input mode, *ed* shall terminate input mode and return to command mode. It is unspecified if any partially entered lines (that is, input text without a terminating <newline>) are discarded from the input text.
- If the *ed* utility is in command mode, it shall act as if a **q** command had been entered.

If the closing delimiter of an RE or of a replacement string (for example, ' / ' ) in a **g**, **G**, **s**, **v**, or **V** command would be the last character before a <newline>, that delimiter can be omitted, in which case the addressed line shall be written. For example, the following pairs of commands are equivalent:

```
s/s1/s2 s/s1/s2/p
g/s1 g/s1/p
?s1 ?s1?
```

If an invalid command is entered, *ed* shall write the string:

"?\n"

(followed by an explanatory message if *help mode* has been enabled via the **H** command) to standard output and shall continue in command mode with the current line number unchanged.

## Append Command

*Synopsis:*     ( . ) a  
              <text>  
              .

The **a** command shall read the given text and append it after the addressed line; the current line number shall become the address of the last inserted line or, if there were none, the addressed line. Address 0 shall be valid for this command; it shall cause the appended text to be placed at the beginning of the buffer.

**Change Command**

*Synopsis:*     ( . , . ) c  
                   <text>  
                   .

The **c** command shall delete the addressed lines, then accept input text that replaces these lines; the current line shall be set to the address of the last line input; or, if there were none, at the line after the last line deleted; if the lines deleted were originally at the end of the buffer, the current line number shall be set to the address of the new last line; if no lines remain in the buffer, the current line number shall be set to zero. Address 0 shall be valid for this command; it shall be interpreted as if address 1 were specified.

**Delete Command**

*Synopsis:*     ( . , . ) d

The **d** command shall delete the addressed lines from the buffer. The address of the line after the last line deleted shall become the current line number; if the lines deleted were originally at the end of the buffer, the current line number shall be set to the address of the new last line; if no lines remain in the buffer, the current line number shall be set to zero.

**Edit Command**

*Synopsis:*     e [*file*]

The **e** command shall delete the entire contents of the buffer and then read in the file named by the pathname *file*. The current line number shall be set to the address of the last line of the buffer. If no pathname is given, the currently remembered pathname, if any, shall be used (see the **f** command). The number of bytes read shall be written to standard output, unless the **-s** option was specified, in the following format:

"%d\n", <number of bytes read>

The name *file* shall be remembered for possible use as a default pathname in subsequent **e**, **E**, **r**, and **w** commands. If *file* is replaced by **'!'**, the rest of the line shall be taken to be a shell command line whose output is to be read. Such a shell command line shall not be remembered as the current *file*. All marks shall be discarded upon the completion of a successful **e** command. If the buffer has changed since the last time the entire buffer was written, the user shall be warned, as described previously.

**Edit Without Checking Command**

*Synopsis:*     E [*file*]

The **E** command shall possess all properties and restrictions of the **e** command except that the editor shall not check to see whether any changes have been made to the buffer since the last **w** command.

## Filename Command

*Synopsis:*     **f** [*file*]

If *file* is given, the **f** command shall change the currently remembered pathname to *file*; whether the name is changed or not, it shall then write the (possibly new) currently remembered pathname to the standard output in the following format:

"%s\n", <pathname>

The current line number shall be unchanged.

## Global Command

*Synopsis:*     (1, \$) **g**/RE/command list

In the **g** command, the first step shall be to mark every line for which the line excluding the terminating <newline> matches the given RE. Then, going sequentially from the beginning of the file to the end of the file, the given *command list* shall be executed for each marked line, with the current line number set to the address of that line. Any line modified by the *command list* shall be unmarked. When the **g** command completes, the current line number shall have the value assigned by the last command in the *command list*. If there were no matching lines, the current line number shall not be changed. A single command or the first of a list of commands shall appear on the same line as the global command. All lines of a multi-line list except the last line shall be ended with a <backslash> preceding the terminating <newline>; the **a**, **i**, and **c** commands and associated input are permitted. The ' .' terminating input mode can be omitted if it would be the last line of the *command list*. An empty *command list* shall be equivalent to the **p** command. The use of the **g**, **G**, **v**, **V**, and **!** commands in the *command list* produces undefined results. Any character other than <space> or <newline> can be used instead of a <slash> to delimit the RE. Within the RE, the RE delimiter itself can be used as a literal character if it is preceded by a <backslash>.

## Interactive Global Command

*Synopsis:*     (1, \$) **G**/RE/

In the **G** command, the first step shall be to mark every line for which the line excluding the terminating <newline> matches the given RE. Then, for every such line, that line shall be written, the current line number shall be set to the address of that line, and any one command (other than one of the **a**, **c**, **i**, **g**, **G**, **v**, and **V** commands) shall be read and executed. A <newline> shall act as a null command (causing no action to be taken on the current line); an ' & ' shall cause the re-execution of the most recent non-null command executed within the current invocation of **G**. Note that the commands input as part of the execution of the **G** command can address and affect any lines in the buffer. Any line modified by the command shall be unmarked. The final value of the current line number shall be the value set by the last command successfully executed. (Note that the last command successfully executed shall be the **G** command itself if a command fails or the null command is specified.) If there were no matching lines, the current line number shall not be changed. The **G** command can be terminated by a SIGINT signal. Any character other than <space> or <newline> can be used instead of a <slash> to delimit the RE and the replacement. Within the RE, the RE delimiter itself can be used as a literal character if it is preceded by a <backslash>.

**Help Command**

*Synopsis:*     h

The **h** command shall write a short message to standard output that explains the reason for the most recent ' ? ' notification. The current line number shall be unchanged.

**Help-Mode Command**

*Synopsis:*     H

The **H** command shall cause *ed* to enter a mode in which help messages (see the **h** command) shall be written to standard output for all subsequent ' ? ' notifications. The **H** command alternately shall turn this mode on and off; it is initially off. If the help-mode is being turned on, the **H** command also explains the previous ' ? ' notification, if there was one. The current line number shall be unchanged.

**Insert Command**

*Synopsis:*     ( . ) i  
                  <text>  
                  .

The **i** command shall insert the given text before the addressed line; the current line is set to the last inserted line or, if there was none, to the addressed line. This command differs from the **a** command only in the placement of the input text. Address 0 shall be valid for this command; it shall be interpreted as if address 1 were specified.

**Join Command**

*Synopsis:*     ( . , . +1 ) j

The **j** command shall join contiguous lines by removing the appropriate <newline> characters. If exactly one address is given, this command shall do nothing. If lines are joined, the current line number shall be set to the address of the joined line; otherwise, the current line number shall be unchanged.

**Mark Command**

*Synopsis:*     ( . ) k x

The **k** command shall mark the addressed line with name *x*, which the application shall ensure is a lowercase letter from the portable character set. The address "' x" shall then refer to this line; the current line number shall be unchanged.

**List Command**

*Synopsis:*     ( . , . ) l

The **l** command shall write to standard output the addressed lines in a visually unambiguous form. The characters listed in XBD Table 5-1 (on page 121) ('\\', '\a', '\b', '\f', '\r', '\t', '\v') shall be written as the corresponding escape sequence; the '\n' in that table is not applicable. Non-printable characters not in the table shall be written as one three-digit octal number (with a preceding <backslash> character) for each byte in the character (most significant byte first).

Long lines shall be folded, with the point of folding indicated by <newline> preceded by a <backslash>; the length at which folding occurs is unspecified, but should be appropriate for the

output device. The end of each line shall be marked with a '\$', and '\$' characters within the text shall be written with a preceding <backslash>. An **l** command can be appended to any other command other than **e**, **E**, **f**, **q**, **Q**, **r**, **w**, or **!**. The current line number shall be set to the address of the last line written.

### Move Command

*Synopsis:* ( , . ) *m* *address*

The **m** command shall reposition the addressed lines after the line addressed by *address*. Address 0 shall be valid for *address* and cause the addressed lines to be moved to the beginning of the buffer. It shall be an error if *address* falls within the range of moved lines. The current line number shall be set to the address of the last line moved.

### Number Command

*Synopsis:* ( , . ) *n*

The **n** command shall write to standard output the addressed lines, preceding each line by its line number and a <tab>; the current line number shall be set to the address of the last line written. The **n** command can be appended to any command other than **e**, **E**, **f**, **q**, **Q**, **r**, **w**, or **!**.

### Print Command

*Synopsis:* ( , . ) *p*

The **p** command shall write to standard output the addressed lines; the current line number shall be set to the address of the last line written. The **p** command can be appended to any command other than **e**, **E**, **f**, **q**, **Q**, **r**, **w**, or **!**.

### Prompt Command

*Synopsis:* *P*

The **P** command shall cause *ed* to prompt with an <asterisk> ('\*') (or *string*, if **-p** is specified) for all subsequent commands. The **P** command alternatively shall turn this mode on and off; it shall be initially on if the **-p** option is specified; otherwise, off. The current line number shall be unchanged.

### Quit Command

*Synopsis:* *q*

The **q** command shall cause *ed* to exit. If the buffer has changed since the last time the entire buffer was written, the user shall be warned, as described previously.

### Quit Without Checking Command

*Synopsis:* *Q*

The **Q** command shall cause *ed* to exit without checking whether changes have been made in the buffer since the last **w** command.



**Read Command**

*Synopsis:*        (\$)**r** [*file*]

The **r** command shall read in the file named by the pathname *file* and append it after the addressed line. If no *file* argument is given, the currently remembered pathname, if any, shall be used (see the **e** and **f** commands). The currently remembered pathname shall not be changed unless there is no remembered pathname. Address 0 shall be valid for **r** and shall cause the file to be read at the beginning of the buffer. If the read is successful, and **-s** was not specified, the number of bytes read shall be written to standard output in the following format:

"%d\n", <number of bytes read>

The current line number shall be set to the address of the last line read in. If *file* is replaced by '!', the rest of the line shall be taken to be a shell command line whose output is to be read. Such a shell command line shall not be remembered as the current pathname.

**Substitute Command**

*Synopsis:*        (.,.)**s**/*RE/replacement/flags*

The **s** command shall search each addressed line for an occurrence of the specified RE and replace either the first or all (non-overlapped) matched strings with the *replacement*; see the following description of the **g** suffix. It is an error if the substitution fails on every addressed line. Any character other than <space> or <newline> can be used instead of a <slash> to delimit the RE and the replacement. Within the RE, the RE delimiter itself can be used as a literal character if it is preceded by a <backslash>. The current line shall be set to the address of the last line on which a substitution occurred.

An <ampersand> ('&') appearing in the *replacement* shall be replaced by the string matching the RE on the current line. The special meaning of '&' in this context can be suppressed by preceding it by <backslash>. As a more general feature, the characters '\n', where *n* is a digit, shall be replaced by the text matched by the corresponding back-reference expression. If the corresponding back-reference expression does not match, then the characters '\n' shall be replaced by the empty string. When the character '%' is the only character in the *replacement*, the *replacement* used in the most recent substitute command shall be used as the *replacement* in the current substitute command; if there was no previous substitute command, the use of '%' in this manner shall be an error. The '%' shall lose its special meaning when it is in a replacement string of more than one character or is preceded by a <backslash>. For each <backslash> encountered in scanning *replacement* from beginning to end, the following character shall lose its special meaning (if any). It is unspecified what special meaning is given to any character other than <backslash>, '&', '%', or digits.

A line can be split by substituting a <newline> into it. The application shall ensure it escapes the <newline> in the *replacement* by preceding it by <backslash>. Such substitution cannot be done as part of a **g** or **v** *command list*. The current line number shall be set to the address of the last line on which a substitution is performed. If no substitution is performed, the current line number shall be unchanged. If a line is split, a substitution shall be considered to have been performed on each of the new lines for the purpose of determining the new current line number. A substitution shall be considered to have been performed even if the replacement string is identical to the string that it replaces.

The application shall ensure that the value of *flags* is zero or more of:

*count*        Substitute for the *count*th occurrence only of the RE found on each addressed line.

- 86221 **g** Globally substitute for all non-overlapping instances of the RE rather than just the first  
 86222 one. If both **g** and *count* are specified, the results are unspecified.
- 86223 **l** Write to standard output the final line in which a substitution was made. The line shall  
 86224 be written in the format specified for the **l** command.
- 86225 **n** Write to standard output the final line in which a substitution was made. The line shall  
 86226 be written in the format specified for the **n** command.
- 86227 **p** Write to standard output the final line in which a substitution was made. The line shall  
 86228 be written in the format specified for the **p** command.

### 86229 Copy Command

86230 *Synopsis:* `(. , .) t address`

86231 The **t** command shall be equivalent to the **m** command, except that a copy of the addressed lines  
 86232 shall be placed after address *address* (which can be 0); the current line number shall be set to the  
 86233 address of the last line added.

### 86234 Undo Command

86235 *Synopsis:* `u`

86236 The **u** command shall nullify the effect of the most recent command that modified anything in  
 86237 the buffer, namely the most recent **a**, **c**, **d**, **g**, **i**, **j**, **m**, **r**, **s**, **t**, **u**, **v**, **G**, or **V** command. All changes  
 86238 made to the buffer by a **g**, **G**, **v**, or **V** global command shall be undone as a single change; if no  
 86239 changes were made by the global command (such as with **g**/RE/**p**), the **u** command shall have  
 86240 no effect. The current line number shall be set to the value it had immediately before the  
 86241 command being undone started.

### 86242 Global Non-Matched Command

86243 *Synopsis:* `(1, $) v/RE/command list`

86244 This command shall be equivalent to the global command **g** except that the lines that are marked  
 86245 during the first step shall be those for which the line excluding the terminating <newline> does  
 86246 not match the RE.

### 86247 Interactive Global Not-Matched Command

86248 *Synopsis:* `(1, $) V/RE/`

86249 This command shall be equivalent to the interactive global command **G** except that the lines that  
 86250 are marked during the first step shall be those for which the line excluding the terminating  
 86251 <newline> does not match the RE.

### 86252 Write Command

86253 *Synopsis:* `(1, $) w [file]`

86254 The **w** command shall write the addressed lines into the file named by the pathname *file*. The  
 86255 command shall create the file, if it does not exist, or shall replace the contents of the existing file.  
 86256 The currently remembered pathname shall not be changed unless there is no remembered  
 86257 pathname. If no pathname is given, the currently remembered pathname, if any, shall be used  
 86258 (see the **e** and **f** commands); the current line number shall be unchanged. If the command is  
 86259 successful, the number of bytes written shall be written to standard output, unless the **-s** option  
 86260 was specified, in the following format:

86261 "%d\n", <number of bytes written>

86262 If *file* begins with '!', the rest of the line shall be taken to be a shell command line whose  
 86263 standard input shall be the addressed lines. Such a shell command line shall not be remembered  
 86264 as the current pathname. This usage of the write command with '!' shall not be considered as a  
 86265 "last **w** command that wrote the entire buffer", as described previously; thus, this alone shall  
 86266 not prevent the warning to the user if an attempt is made to destroy the editor buffer via the **e** or  
 86267 **q** commands.

## 86268 **Line Number Command**

86269 *Synopsis:* (\$) =

86270 The line number of the addressed line shall be written to standard output in the following  
 86271 format:

86272 "%d\n", <line number>

86273 The current line number shall be unchanged by this command.

## 86274 **Shell Escape Command**

86275 *Synopsis:* !*command*

86276 The remainder of the line after the '!' shall be sent to the command interpreter to be  
 86277 interpreted as a shell command line. Within the text of that shell command line, the unescaped  
 86278 character '%' shall be replaced with the remembered pathname; if a '!' appears as the first  
 86279 character of the command, it shall be replaced with the text of the previous shell command  
 86280 executed via '!'. Thus, "!!" shall repeat the previous !*command*. If any replacements of '%' or  
 86281 '!' are performed, the modified line shall be written to the standard output before *command* is  
 86282 executed. The ! command shall write:

86283 "!\n"

86284 to standard output upon completion, unless the -s option is specified. The current line number  
 86285 shall be unchanged.

## 86286 **Null Command**

86287 *Synopsis:* (.+1)

86288 An address alone on a line shall cause the addressed line to be written. A <newline> alone shall  
 86289 be equivalent to "+1p". The current line number shall be set to the address of the written line.

## 86290 **EXIT STATUS**

86291 The following exit values shall be returned:

86292 0 Successful completion without any file or command errors.

86293 >0 An error occurred.

## 86294 **CONSEQUENCES OF ERRORS**

86295 When an error in the input script is encountered, or when an error is detected that is a  
 86296 consequence of the data (not) present in the file or due to an external condition such as a read or  
 86297 write error:

- 86298 • If the standard input is a terminal device file, all input shall be flushed, and a new  
 86299 command read.

- If the standard input is a regular file, *ed* shall terminate with a non-zero exit status.

## 86301 APPLICATION USAGE

86302 Because of the extremely terse nature of the default error messages, the prudent script writer  
86303 begins the *ed* input commands with an **H** command, so that if any errors do occur at least some  
86304 clue as to the cause is made available.

86305 In earlier versions of this standard, an obsolescent `–` option was described. This is no longer  
86306 specified. Applications should use the `–s` option. Using `–` as a *file* operand now produces  
86307 unspecified results. This allows implementations to continue to support the former required  
86308 behavior.

## 86309 EXAMPLES

86310 None.

## 86311 RATIONALE

86312 The initial description of this utility was adapted from the SVID. It contains some features not  
86313 found in Version 7 or BSD-derived systems. Some of the differences between the POSIX and  
86314 BSD *ed* utilities include, but need not be limited to:

- 86315 • The BSD `–` option does not suppress the `'!'` prompt after a `!` command.
- 86316 • BSD does not support the special meanings of the `'%'` and `'!'` characters within a `!`  
86317 command.
- 86318 • BSD does not support the *addresses* `' ; '` and `' , '`.
- 86319 • BSD allows the command/suffix pairs **pp**, **ll**, and so on, which are unspecified in this  
86320 volume of POSIX.1-2008.
- 86321 • BSD does not support the `'!'` character part of the **e**, **r**, or **w** commands.
- 86322 • A failed **g** command in BSD sets the line number to the last line searched if there are no  
86323 matches.
- 86324 • BSD does not default the *command list* to the **p** command.
- 86325 • BSD does not support the **G**, **h**, **H**, **n**, or **V** commands.
- 86326 • On BSD, if there is no inserted text, the insert command changes the current line to the  
86327 referenced line `–1`; that is, the line before the specified line.
- 86328 • On BSD, the *join* command with only a single address changes the current line to that  
86329 address.
- 86330 • BSD does not support the **P** command; moreover, in BSD it is synonymous with the **p**  
86331 command.
- 86332 • BSD does not support the *undo* of the commands **j**, **m**, **r**, **s**, or **t**.
- 86333 • The Version 7 *ed* command **W**, and the BSD *ed* commands **W**, **wq**, and **z** are not present in  
86334 this volume of POSIX.1-2008.

86335 The `–s` option was added to allow the functionality of the removed `–` option in a manner  
86336 compatible with the Utility Syntax Guidelines.

86337 In early proposals there was a limit, {ED\_FILE\_MAX}, that described the historical limitations of  
86338 some *ed* utilities in their handling of large files; some of these have had problems with files  
86339 larger than 100 000 bytes. It was this limitation that prompted much of the desire to include a  
86340 *split* command in this volume of POSIX.1-2008. Since this limit was removed, this volume of  
86341 POSIX.1-2008 requires that implementations document the file size limits imposed by *ed* in the

conformance document. The limit {ED\_LINE\_MAX} was also removed; therefore, the global limit {LINE\_MAX} is used for input and output lines.

The manner in which the **l** command writes non-printable characters was changed to avoid the historical backspace-overstrike method. On video display terminals, the overstrike is ambiguous because most terminals simply replace overstruck characters, making the **l** format not useful for its intended purpose of unambiguously understanding the content of the line. The historical <backslash>-escapes were also ambiguous. (The string "a\0011" could represent a line containing those six characters or a line containing the three characters 'a', a byte with a binary value of 1, and a 1.) In the format required here, a <backslash> appearing in the line is written as "\\ " so that the output is truly unambiguous. The method of marking the ends of lines was adopted from the *ex* editor and is required for any line ending in <space> characters; the '\$' is placed on all lines so that a real '\$' at the end of a line cannot be misinterpreted.

Earlier versions of this standard allowed for implementations with bytes other than eight bits, but this has been modified in this version.

The description of how a NUL is written was removed. The NUL character cannot be in text files, and this volume of POSIX.1-2008 should not dictate behavior in the case of undefined, erroneous input.

Unlike some of the other editing utilities, the filenames accepted by the **E**, **e**, **R**, and **r** commands are not patterns.

Early proposals stated that the **-p** option worked only when standard input was associated with a terminal device. This has been changed to conform to historical implementations, thereby allowing applications to interpose themselves between a user and the *ed* utility.

The form of the substitute command that uses the **n** suffix was limited in some historical documentation (where this was described incorrectly as "backreferencing"). This limit has been omitted because there is no reason why an editor processing lines of {LINE\_MAX} length should have this restriction. The command **s/x/X/2047** should be able to substitute the 2047th occurrence of 'x' on a line.

The use of printing commands with printing suffixes (such as **pn**, **lp**, and so on) was made unspecified because BSD-based systems allow this, whereas System V does not.

Some BSD-based systems exit immediately upon receipt of end-of-file if all of the lines in the file have been deleted. Since this volume of POSIX.1-2008 refers to the **q** command in this instance, such behavior is not allowed.

Some historical implementations returned exit status zero even if command errors had occurred; this is not allowed by this volume of POSIX.1-2008.

Some historical implementations contained a bug that allowed a single <period> to be entered in input mode as <backslash> <period> <newline>. This is not allowed by *ed* because there is no description of escaping any of the characters in input mode; <backslash> characters are entered into the buffer exactly as typed. The typical method of entering a single <period> has been to precede it with another character and then use the substitute command to delete that character.

It is difficult under some modes of some versions of historical operating system terminal drivers to distinguish between an end-of-file condition and terminal disconnect. POSIX.1-2008 does not require implementations to distinguish between the two situations, which permits historical implementations of the *ed* utility on historical platforms to conform. Implementations are encouraged to distinguish between the two, if possible, and take appropriate action on terminal disconnect.

Historically, *ed* accepted a zero address for the **a** and **r** commands in order to insert text at the

start of the edit buffer. When the buffer was empty the command `.=` returned zero. POSIX.1-2008 requires conformance to historical practice.

For consistency with the `a` and `r` commands and better user functionality, the `i` and `c` commands must also accept an address of 0, in which case `0i` is treated as `1i` and likewise for the `c` command.

All of the following are valid addresses:

```
+++ Three lines after the current line.
/pattern/- One line before the next occurrence of pattern.
-2 Two lines before the current line.
3 ---- 2 Line one (note the intermediate negative address).
1 2 3 Line six.
```

Any number of addresses can be provided to commands taking addresses; for example, `"1,2,3,4,5p"` prints lines 4 and 5, because two is the greatest valid number of addresses accepted by the `print` command. This, in combination with the `<semicolon>` delimiter, permits users to create commands based on ordered patterns in the file. For example, the command `"3;/foo/;+2p"` will display the first line after line 3 that contains the pattern `foo`, plus the next two lines. Note that the address `"3;"` must still be evaluated before being discarded, because the search origin for the `"/foo/"` command depends on this.

Historically, `ed` disallowed address chains, as discussed above, consisting solely of `<comma>` or `<semicolon>` separators; for example, `",,,,"` or `";;;"` were considered an error. For consistency of address specification, this restriction is removed. The following table lists some of the address forms now possible:

| Address | Addr1 | Addr2 | Status     | Comment               |
|---------|-------|-------|------------|-----------------------|
| 7,      | 7     | 7     | Historical | Valid, but erroneous. |
| 7,5,    | 5     | 5     | Historical |                       |
| 7,5,9   | 5     | 9     | Historical |                       |
| 7,9     | 7     | 9     | Historical |                       |
| 7,+     | 7     | 8     | Historical |                       |
| ,       | 1     | \$    | Historical |                       |
| ,7      | 1     | 7     | Extension  |                       |
| ,,      | \$    | \$    | Extension  |                       |
| ,;      | \$    | \$    | Extension  |                       |
| 7;      | 7     | 7     | Historical |                       |
| 7;5;    | 5     | 5     | Historical |                       |
| 7;5;9   | 5     | 9     | Historical |                       |
| 7;5,9   | 5     | 9     | Historical |                       |
| 7;\$;4  | \$    | 4     | Historical |                       |
| 7;9     | 7     | 9     | Historical |                       |
| 7;+     | 7     | 8     | Historical |                       |
| ;       | .     | \$    | Historical |                       |
| ;7      | .     | 7     | Extension  |                       |
| ;;      | \$    | \$    | Extension  |                       |
| ;;      | \$    | \$    | Extension  |                       |

Historically, `ed` accepted the `'^'` character as an address, in which case it was identical to the `<hyphen>` character. POSIX.1-2008 does not require or prohibit this behavior.

**86433 FUTURE DIRECTIONS**

86434 None.

**86435 SEE ALSO**

86436 [Section 1.4](#) (on page 2310), *ex*, *sed*, *sh*, *vi*

86437 XBD [Table 5-1](#) (on page 121), [Chapter 8](#) (on page 173), [Section 9.3](#) (on page 183), [Chapter 11](#) (on  
86438 page 199), [Section 12.2](#) (on page 215)

**86439 CHANGE HISTORY**

86440 First released in Issue 2.

**86441 Issue 5**

86442 In the OPTIONS section, the meaning of `-s` and `-` is clarified.

86443 A second FUTURE DIRECTION is added.

**86444 Issue 6**

86445 The obsolescent single-minus form is removed.

86446 A second APPLICATION USAGE note is added.

86447 The Open Group Corrigendum U025/2 is applied, correcting the description of the Edit section.

86448 The *ed* utility is updated to align with the IEEE P1003.2b draft standard. This includes addition  
86449 of the treatment of the SIGQUIT signal, changes to *ed* addressing, and changes to processing  
86450 when end-of-file is detected and when terminal disconnect is detected.

86451 The normative text is reworded to avoid use of the term “must” for application requirements.

86452 IEEE Std 1003.1-2001/Cor 1-2002, item XCU/TC1/D6/22 is applied, adding the text: “Any line  
86453 modified by the *command list* shall be unmarked.” to the **G** command. This change corresponds  
86454 to a similar change made to the **g** command in the first version of this standard.

86455 IEEE Std 1003.1-2001/Cor 2-2004, item XCU/TC2/D6/7 is applied, removing text describing  
86456 behavior on systems with bytes consisting of more than eight bits.

**86457 Issue 7**

86458 Austin Group Interpretation 1003.1-2001 #027 is applied, clarifying the behavior if an operand is  
86459 ‘-’.

86460 Austin Group Interpretation 1003.1-2001 #036 is applied, clarifying the behavior for BREs.

86461 SD5-XCU-ERN-94 is applied, updating text in the EXTENDED DESCRIPTION where a terminal  
86462 disconnect is detected (in Commands in *ed*).

86463 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

86464 SD5-XCU-ERN-135 is applied, removing some RATIONALE text that is no longer applicable.

86465 **NAME**86466 `env` — set the environment for command invocation86467 **SYNOPSIS**86468 `env [-i] [name=value]... [utility [argument...]]`86469 **DESCRIPTION**86470 The *env* utility shall obtain the current environment, modify it according to its arguments, then  
86471 invoke the utility named by the *utility* operand with the modified environment.86472 Optional arguments shall be passed to *utility*.86473 If no *utility* operand is specified, the resulting environment shall be written to the standard  
86474 output, with one *name=value* pair per line.86475 If the first argument is `'-'`, the results are unspecified.86476 **OPTIONS**86477 The *env* utility shall conform to XBD [Section 12.2](#) (on page 215), except for the unspecified usage  
86478 of `'-'`.

86479 The following options shall be supported:

86480 **-i** Invoke *utility* with exactly the environment specified by the arguments; the  
86481 inherited environment shall be ignored completely.86482 **OPERANDS**

86483 The following operands shall be supported:

86484 *name=value* Arguments of the form *name=value* shall modify the execution environment, and  
86485 shall be placed into the inherited environment before the *utility* is invoked.86486 *utility* The name of the utility to be invoked. If the *utility* operand names any of the  
86487 special built-in utilities in [Section 2.14](#) (on page 2356), the results are undefined.86488 *argument* A string to pass as an argument for the invoked utility.86489 **STDIN**

86490 Not used.

86491 **INPUT FILES**

86492 None.

86493 **ENVIRONMENT VARIABLES**86494 The following environment variables shall affect the execution of *env*:86495 *LANG* Provide a default value for the internationalization variables that are unset or null.  
86496 (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
86497 variables used to determine the values of locale categories.)86498 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
86499 internationalization variables.86500 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
86501 characters (for example, single-byte as opposed to multi-byte characters in  
86502 arguments).86503 *LC\_MESSAGES*86504 Determine the locale that should be used to affect the format and contents of  
86505 diagnostic messages written to standard error.



|       |     |                               |                                                                                                                                                     |
|-------|-----|-------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------|
| 86506 | XSI | <b>NLSPATH</b>                | Determine the location of message catalogs for the processing of <i>LC_MESSAGES</i> .                                                               |
| 86507 |     | <b>PATH</b>                   | Determine the location of the <i>utility</i> , as described in XBD <a href="#">Chapter 8</a> (on page 173).                                         |
| 86508 |     |                               | If <i>PATH</i> is specified as a <i>name=value</i> operand to <i>env</i> , the <i>value</i> given shall be used in                                  |
| 86509 |     |                               | the search for <i>utility</i> .                                                                                                                     |
| 86510 |     | <b>ASYNCHRONOUS EVENTS</b>    |                                                                                                                                                     |
| 86511 |     |                               | Default.                                                                                                                                            |
| 86512 |     | <b>STDOUT</b>                 |                                                                                                                                                     |
| 86513 |     |                               | If no <i>utility</i> operand is specified, each <i>name=value</i> pair in the resulting environment shall be                                        |
| 86514 |     |                               | written in the form:                                                                                                                                |
| 86515 |     |                               | "%s=%s\n", < <i>name</i> >, < <i>value</i> >                                                                                                        |
| 86516 |     |                               | If the <i>utility</i> operand is specified, the <i>env</i> utility shall not write to standard output.                                              |
| 86517 |     | <b>STDERR</b>                 |                                                                                                                                                     |
| 86518 |     |                               | The standard error shall be used only for diagnostic messages.                                                                                      |
| 86519 |     | <b>OUTPUT FILES</b>           |                                                                                                                                                     |
| 86520 |     |                               | None.                                                                                                                                               |
| 86521 |     | <b>EXTENDED DESCRIPTION</b>   |                                                                                                                                                     |
| 86522 |     |                               | None.                                                                                                                                               |
| 86523 |     | <b>EXIT STATUS</b>            |                                                                                                                                                     |
| 86524 |     |                               | If <i>utility</i> is invoked, the exit status of <i>env</i> shall be the exit status of <i>utility</i> ; otherwise, the <i>env</i>                  |
| 86525 |     |                               | utility shall exit with one of the following values:                                                                                                |
| 86526 |     | 0                             | The <i>env</i> utility completed successfully.                                                                                                      |
| 86527 |     | 1–125                         | An error occurred in the <i>env</i> utility.                                                                                                        |
| 86528 |     | 126                           | The utility specified by <i>utility</i> was found but could not be invoked.                                                                         |
| 86529 |     | 127                           | The utility specified by <i>utility</i> could not be found.                                                                                         |
| 86530 |     | <b>CONSEQUENCES OF ERRORS</b> |                                                                                                                                                     |
| 86531 |     |                               | Default.                                                                                                                                            |
| 86532 |     | <b>APPLICATION USAGE</b>      |                                                                                                                                                     |
| 86533 |     |                               | The <i>command</i> , <i>env</i> , <i>nice</i> , <i>nohup</i> , <i>time</i> , and <i>xargs</i> utilities have been specified to use exit code 127 if |
| 86534 |     |                               | an error occurs so that applications can distinguish “failure to find a utility” from “invoked                                                      |
| 86535 |     |                               | utility exited with an error indication”. The value 127 was chosen because it is not commonly                                                       |
| 86536 |     |                               | used for other meanings; most utilities use small values for “normal error conditions” and the                                                      |
| 86537 |     |                               | values above 128 can be confused with termination due to receipt of a signal. The value 126 was                                                     |
| 86538 |     |                               | chosen in a similar manner to indicate that the utility could be found, but not invoked. Some                                                       |
| 86539 |     |                               | scripts produce meaningful error messages differentiating the 126 and 127 cases. The distinction                                                    |
| 86540 |     |                               | between exit codes 126 and 127 is based on KornShell practice that uses 127 when all attempts to                                                    |
| 86541 |     |                               | <i>exec</i> the utility fail with [ENOENT], and uses 126 when any attempt to <i>exec</i> the utility fails for                                      |
| 86542 |     |                               | any other reason.                                                                                                                                   |
| 86543 |     |                               | Historical implementations of the <i>env</i> utility use the <i>execvp()</i> or <i>execlp()</i> functions defined in the                            |
| 86544 |     |                               | System Interfaces volume of POSIX.1-2008 to invoke the specified utility; this provides better                                                      |
| 86545 |     |                               | performance and keeps users from having to escape characters with special meaning to the shell.                                                     |
| 86546 |     |                               | Therefore, shell functions, special built-ins, and built-ins that are only provided by the shell are                                                |
| 86547 |     |                               | not found.                                                                                                                                          |

**EXAMPLES**

The following command:

```
env -i PATH=/mybin:"$PATH" $(getconf V7_ENV) mygrep xyz myfile
```

invokes the command *mygrep* with a new *PATH* value as the only entry in its environment other than any variables required by the implementation for conformance. In this case, *PATH* is used to locate *mygrep*, which is expected to reside in */mybin*.

**RATIONALE**

As with all other utilities that invoke other utilities, this volume of POSIX.1-2008 only specifies what *env* does with standard input, standard output, standard error, input files, and output files. If a utility is executed, it is not constrained by the specification of input and output by *env*.

The *-i* option was added to allow the functionality of the removed *-* option in a manner compatible with the Utility Syntax Guidelines. It is possible to create a non-conforming environment using the *-i* option, as it may remove environment variables required by the implementation for conformance. The following will preserve these environment variables as well as preserve the *PATH* for conforming utilities:

```
IFS='
The preceding value should be <space><tab><newline>.
Set IFS to its default value.

set -f
disable pathname expansion

\unalias -a
Unset all possible aliases.
Note that unalias is escaped to prevent an alias
being used for unalias.
This step is not strictly necessary, since aliases are not inherited,
and the ENV environment variable is only used by interactive shells,
the only way any aliases can exist in a script is if it defines them
itself.

unset -f env getconf
Ensure env and getconf are not user functions.

env -i $(getconf V7_ENV) PATH="$(getconf PATH)" command
```

Some have suggested that *env* is redundant since the same effect is achieved by:

```
name=value ... utility [argument ...]
```

The example is equivalent to *env* when an environment variable is being added to the environment of the command, but not when the environment is being set to the given value. The *env* utility also writes out the current environment if invoked without arguments. There is sufficient functionality beyond what the example provides to justify inclusion of *env*.

**FUTURE DIRECTIONS**

None.

**SEE ALSO**

[Section 2.14](#) (on page 2356), [Section 2.5](#) (on page 2323)

[XBD Chapter 8](#) (on page 173), [Section 12.2](#) (on page 215)

**86591 CHANGE HISTORY**

86592 First released in Issue 2.

**86593 Issue 7**

86594 Austin Group Interpretation 1003.1-2001 #027 is applied, clarifying the behavior if the first  
86595 argument is `'-'`.

86596 Austin Group Interpretation 1003.1-2001 #047 is applied, providing RATIONALE on how to use  
86597 the `env` utility to preserve a conforming environment.

86598 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

86599 The EXAMPLES section is revised to change the use of `env -i`.

## 86600 NAME

86601 ex — text editor

## 86602 SYNOPSIS

86603 UP `ex [-rR] [-s|-v] [-c command] [-t tagstring] [-w size] [file...]`

## 86604 DESCRIPTION

86605 The *ex* utility is a line-oriented text editor. There are two other modes of the editor—open and  
 86606 visual—in which screen-oriented editing is available. This is described more fully by the *ex* **open**  
 86607 and **visual** commands and in *vi*.

86608 If an operand is *'-'*, the results are unspecified.

86609 This section uses the term *edit buffer* to describe the current working text. No specific  
 86610 implementation is implied by this term. All editing changes are performed on the edit buffer,  
 86611 and no changes to it shall affect any file until an editor command writes the file.

86612 Certain terminals do not have all the capabilities necessary to support the complete *ex* definition,  
 86613 such as the full-screen editing commands (*visual mode* or *open mode*). When these commands  
 86614 cannot be supported on such terminals, this condition shall not produce an error message such  
 86615 as “not an editor command” or report a syntax error. The implementation may either accept the  
 86616 commands and produce results on the screen that are the result of an unsuccessful attempt to  
 86617 meet the requirements of this volume of POSIX.1-2008 or report an error describing the terminal-  
 86618 related deficiency.

## 86619 OPTIONS

86620 The *ex* utility shall conform to XBD [Section 12.2](#) (on page 215), except for the unspecified usage  
 86621 of *'-'*, and that *'+'* may be recognized as an option delimiter as well as *'-'*.

86622 The following options shall be supported:

86623 **-c *command*** Specify an initial command to be executed in the first edit buffer loaded from an  
 86624 existing file (see the EXTENDED DESCRIPTION section). Implementations may  
 86625 support more than a single **-c** option. In such implementations, the specified  
 86626 commands shall be executed in the order specified on the command line.

86627 **-r** Recover the named files (see the EXTENDED DESCRIPTION section). Recovery  
 86628 information for a file shall be saved during an editor or system crash (for example,  
 86629 when the editor is terminated by a signal which the editor can catch), or after the  
 86630 use of an *ex* **preserve** command.

86631 A *crash* in this context is an unexpected failure of the system or utility that requires  
 86632 restarting the failed system or utility. A system crash implies that any utilities  
 86633 running at the time also crash. In the case of an editor or system crash, the number  
 86634 of changes to the edit buffer (since the most recent **preserve** command) that will be  
 86635 recovered is unspecified.

86636 If no *file* operands are given and the **-t** option is not specified, all other options, the  
 86637 *EXINIT* variable, and any **.exrc** files shall be ignored; a list of all recoverable files  
 86638 available to the invoking user shall be written, and the editor shall exit normally  
 86639 without further action.

86640 **-R** Set **readonly** edit option.

86641 **-s** Prepare *ex* for batch use by taking the following actions:

- 86642 • Suppress writing prompts and informational (but not diagnostic) messages.
- 86643 • Ignore the value of *TERM* and any implementation default terminal type and
- 86644 assume the terminal is a type incapable of supporting open or visual modes;
- 86645 see the **visual** command and the description of *vi*.
- 86646 • Suppress the use of the *EXINIT* environment variable and the reading of any
- 86647 **.exrc** file; see the EXTENDED DESCRIPTION section.
- 86648 • Suppress autoindentation, ignoring the value of the **autoindent** edit option.
- 86649 **-t tagstring** Edit the file containing the specified *tagstring*; see *ctags*. The tags feature
- 86650 represented by **-t tagstring** and the **tag** command is optional. It shall be provided
- 86651 on any system that also provides a conforming implementation of *ctags*; otherwise,
- 86652 the use of **-t** produces undefined results. On any system, it shall be an error to
- 86653 specify more than a single **-t** option.
- 86654 **-v** Begin in visual mode (see *vi*).
- 86655 **-w size** Set the value of the *window* editor option to *size*.

## 86656 OPERANDS

86657 The following operand shall be supported:

86658 *file* A pathname of a file to be edited.

## 86659 STDIN

86660 The standard input consists of a series of commands and input text, as described in the

86661 EXTENDED DESCRIPTION section. The implementation may limit each line of standard input

86662 to a length of {LINE\_MAX}.

86663 If the standard input is not a terminal device, it shall be as if the **-s** option had been specified.

86664 If a read from the standard input returns an error, or if the editor detects an end-of-file condition

86665 from the standard input, it shall be equivalent to a SIGHUP asynchronous event.

## 86666 INPUT FILES

86667 Input files shall be text files or files that would be text files except for an incomplete last line that

86668 is not longer than {LINE\_MAX}-1 bytes in length and contains no NUL characters. By default,

86669 any incomplete last line shall be treated as if it had a trailing <newline>. The editing of other

86670 forms of files may optionally be allowed by *ex* implementations.

86671 The **.exrc** files and source files shall be text files consisting of *ex* commands; see the EXTENDED

86672 DESCRIPTION section.

86673 By default, the editor shall read lines from the files to be edited without interpreting any of those

86674 lines as any form of editor command.

## 86675 ENVIRONMENT VARIABLES

86676 The following environment variables shall affect the execution of *ex*:

86677 **COLUMNS** Override the system-selected horizontal screen size. See XBD Chapter 8 (on page

86678 173) for valid values and results when it is unset or null.

86679 **EXINIT** Determine a list of *ex* commands that are executed on editor start-up. See the

86680 EXTENDED DESCRIPTION section for more details of the initialization phase.

86681 **HOME** Determine a pathname of a directory that shall be searched for an editor start-up

86682 file named **.exrc**; see the EXTENDED DESCRIPTION section.

|       |                                                                                                              |                                                                                                                                                                                                                                                                                                                                                                                                   |
|-------|--------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 86683 | <i>LANG</i>                                                                                                  | Provide a default value for the internationalization variables that are unset or null. (See XBD <a href="#">Section 8.2</a> (on page 174) for the precedence of internationalization variables used to determine the values of locale categories.)                                                                                                                                                |
| 86684 |                                                                                                              |                                                                                                                                                                                                                                                                                                                                                                                                   |
| 86685 |                                                                                                              |                                                                                                                                                                                                                                                                                                                                                                                                   |
| 86686 | <i>LC_ALL</i>                                                                                                | If set to a non-empty string value, override the values of all the other internationalization variables.                                                                                                                                                                                                                                                                                          |
| 86687 |                                                                                                              |                                                                                                                                                                                                                                                                                                                                                                                                   |
| 86688 | <i>LC_COLLATE</i>                                                                                            |                                                                                                                                                                                                                                                                                                                                                                                                   |
| 86689 |                                                                                                              | Determine the locale for the behavior of ranges, equivalence classes, and multi-character collating elements within regular expressions.                                                                                                                                                                                                                                                          |
| 86690 |                                                                                                              |                                                                                                                                                                                                                                                                                                                                                                                                   |
| 86691 | <i>LC_CTYPE</i>                                                                                              | Determine the locale for the interpretation of sequences of bytes of text data as characters (for example, single-byte as opposed to multi-byte characters in arguments and input files), the behavior of character classes within regular expressions, the classification of characters as uppercase or lowercase letters, the case conversion of letters, and the detection of word boundaries. |
| 86692 |                                                                                                              |                                                                                                                                                                                                                                                                                                                                                                                                   |
| 86693 |                                                                                                              |                                                                                                                                                                                                                                                                                                                                                                                                   |
| 86694 |                                                                                                              |                                                                                                                                                                                                                                                                                                                                                                                                   |
| 86695 |                                                                                                              |                                                                                                                                                                                                                                                                                                                                                                                                   |
| 86696 | <i>LC_MESSAGES</i>                                                                                           |                                                                                                                                                                                                                                                                                                                                                                                                   |
| 86697 |                                                                                                              | Determine the locale that should be used to affect the format and contents of diagnostic messages written to standard error.                                                                                                                                                                                                                                                                      |
| 86698 |                                                                                                              |                                                                                                                                                                                                                                                                                                                                                                                                   |
| 86699 | <i>LINES</i>                                                                                                 | Override the system-selected vertical screen size, used as the number of lines in a screenful and the vertical screen size in visual mode. See XBD <a href="#">Chapter 8</a> (on page 173) for valid values and results when it is unset or null.                                                                                                                                                 |
| 86700 |                                                                                                              |                                                                                                                                                                                                                                                                                                                                                                                                   |
| 86701 |                                                                                                              |                                                                                                                                                                                                                                                                                                                                                                                                   |
| 86702 | XSI <i>NLSPATH</i>                                                                                           | Determine the location of message catalogs for the processing of <i>LC_MESSAGES</i> .                                                                                                                                                                                                                                                                                                             |
| 86703 | <i>PATH</i>                                                                                                  | Determine the search path for the shell command specified in the <i>ex</i> editor commands <b>!</b> , <b>shell</b> , <b>read</b> , and <b>write</b> , and the open and visual mode command <b>!</b> ; see the description of command search and execution in <a href="#">Section 2.9.1.1</a> (on page 2339).                                                                                      |
| 86704 |                                                                                                              |                                                                                                                                                                                                                                                                                                                                                                                                   |
| 86705 |                                                                                                              |                                                                                                                                                                                                                                                                                                                                                                                                   |
| 86706 | <i>SHELL</i>                                                                                                 | Determine the preferred command line interpreter for use as the default value of the <b>shell</b> edit option.                                                                                                                                                                                                                                                                                    |
| 86707 |                                                                                                              |                                                                                                                                                                                                                                                                                                                                                                                                   |
| 86708 | <i>TERM</i>                                                                                                  | Determine the name of the terminal type. If this variable is unset or null, an unspecified default terminal type shall be used.                                                                                                                                                                                                                                                                   |
| 86709 |                                                                                                              |                                                                                                                                                                                                                                                                                                                                                                                                   |
| 86710 | <b>ASYNCHRONOUS EVENTS</b>                                                                                   |                                                                                                                                                                                                                                                                                                                                                                                                   |
| 86711 | The following term is used in this and following sections to specify command and asynchronous event actions: |                                                                                                                                                                                                                                                                                                                                                                                                   |
| 86712 |                                                                                                              |                                                                                                                                                                                                                                                                                                                                                                                                   |
| 86713 | <i>complete write</i>                                                                                        |                                                                                                                                                                                                                                                                                                                                                                                                   |
| 86714 |                                                                                                              | A complete write is a write of the entire contents of the edit buffer to a file of a type other than a terminal device, or the saving of the edit buffer caused by the user executing the <i>ex</i> <b>preserve</b> command. Writing the contents of the edit buffer to a temporary file that will be removed when the editor exits shall not be considered a complete write.                     |
| 86715 |                                                                                                              |                                                                                                                                                                                                                                                                                                                                                                                                   |
| 86716 |                                                                                                              |                                                                                                                                                                                                                                                                                                                                                                                                   |
| 86717 |                                                                                                              |                                                                                                                                                                                                                                                                                                                                                                                                   |
| 86718 |                                                                                                              |                                                                                                                                                                                                                                                                                                                                                                                                   |
| 86719 | The following actions shall be taken upon receipt of signals:                                                |                                                                                                                                                                                                                                                                                                                                                                                                   |
| 86720 | <i>SIGINT</i>                                                                                                | If the standard input is not a terminal device, <i>ex</i> shall not write the file or return to command or text input mode, and shall exit with a non-zero exit status.                                                                                                                                                                                                                           |
| 86721 |                                                                                                              |                                                                                                                                                                                                                                                                                                                                                                                                   |
| 86722 |                                                                                                              | Otherwise, if executing an open or visual text input mode command, <i>ex</i> in receipt of <i>SIGINT</i> shall behave identically to its receipt of the <ESC> character.                                                                                                                                                                                                                          |
| 86723 |                                                                                                              |                                                                                                                                                                                                                                                                                                                                                                                                   |
| 86724 |                                                                                                              | Otherwise:                                                                                                                                                                                                                                                                                                                                                                                        |

- 86725 1. If executing an *ex* text input mode command, all input lines that have been  
86726 completely entered shall be resolved into the edit buffer, and any partially  
86727 entered line shall be discarded.
- 86728 2. If there is a currently executing command, it shall be aborted and a message  
86729 displayed. Unless otherwise specified by the *ex* or *vi* command descriptions,  
86730 it is unspecified whether any lines modified by the executing command  
86731 appear modified, or as they were before being modified by the executing  
86732 command, in the buffer.
- 86733 If the currently executing command was a motion command, its associated  
86734 command shall be discarded.
- 86735 3. If in open or visual command mode, the terminal shall be alerted.
- 86736 4. The editor shall then return to command mode.
- 86737 SIGCONT The screen shall be refreshed if in open or visual mode.
- 86738 SIGHUP If the edit buffer has been modified since the last complete write, *ex* shall attempt  
86739 to save the edit buffer so that it can be recovered later using the **-r** option or the *ex*  
86740 **recover** command. The editor shall not write the file or return to command or text  
86741 input mode, and shall terminate with a non-zero exit status.
- 86742 SIGTERM Refer to SIGHUP.
- 86743 The action taken for all other signals is unspecified.
- 86744 **STDOUT**
- 86745 The standard output shall be used only for writing prompts to the user, for informational  
86746 messages, and for writing lines from the file.
- 86747 **STDERR**
- 86748 The standard error shall be used only for diagnostic messages.
- 86749 **OUTPUT FILES**
- 86750 The output from *ex* shall be text files.
- 86751 **EXTENDED DESCRIPTION**
- 86752 Only the *ex* mode of the editor is described in this section. See *vi* for additional editing  
86753 capabilities available in *ex*.
- 86754 When an error occurs, *ex* shall write a message. If the terminal supports a standout mode (such  
86755 as inverse video), the message shall be written in standout mode. If the terminal does not  
86756 support a standout mode, and the edit option **errorbells** is set, an alert action shall precede the  
86757 error message.
- 86758 By default, *ex* shall start in command mode, which shall be indicated by a **:** prompt; see the  
86759 **prompt** command. Text input mode can be entered by the **append**, **insert**, or **change** commands;  
86760 it can be exited (and command mode re-entered) by typing a **<period>** (**'.'**) alone at the  
86761 beginning of a line.

## Initialization in ex and vi

The following symbols are used in this and following sections to specify locations in the edit buffer:

### *alternate and current pathnames*

Two pathnames, named *current* and *alternate*, are maintained by the editor. Any *ex* commands that take filenames as arguments shall set them as follows:

1. If a *file* argument is specified to the *ex* **edit**, **ex**, or **recover** commands, or if an *ex* **tag** command replaces the contents of the edit buffer.
  - a. If the command replaces the contents of the edit buffer, the current pathname shall be set to the *file* argument or the file indicated by the tag, and the alternate pathname shall be set to the previous value of the current pathname.
  - b. Otherwise, the alternate pathname shall be set to the *file* argument.
2. If a *file* argument is specified to the *ex* **next** command:
  - a. If the command replaces the contents of the edit buffer, the current pathname shall be set to the first *file* argument, and the alternate pathname shall be set to the previous value of the current pathname.
3. If a *file* argument is specified to the *ex* **file** command, the current pathname shall be set to the *file* argument, and the alternate pathname shall be set to the previous value of the current pathname.
4. If a *file* argument is specified to the *ex* **read** and **write** commands (that is, when reading or writing a file, and not to the program named by the **shell** edit option), or a *file* argument is specified to the *ex* **xit** command:
  - a. If the current pathname has no value, the current pathname shall be set to the *file* argument.
  - b. Otherwise, the alternate pathname shall be set to the *file* argument.

If the alternate pathname is set to the previous value of the current pathname when the current pathname had no previous value, then the alternate pathname shall have no value as a result.

### *current line*

The line of the edit buffer referenced by the cursor. Each command description specifies the current line after the command has been executed, as the *current line value*. When the edit buffer contains no lines, the current line shall be zero; see [Addressing in ex](#) (on page 2668).

### *current column*

The current display line column occupied by the cursor. (The columns shall be numbered beginning at 1.) Each command description specifies the current column after the command has been executed, as the *current column value*. This column is an *ideal* column that is remembered over the lifetime of the editor. The actual display line column upon which the cursor rests may be different from the current column; see the cursor positioning discussion in [Command Descriptions in vi](#) (on page 3337).

### *set to non-<blank>*

A description for a current column value, meaning that the current column shall be set to the last display line column on which is displayed any part of the first non-<blank> of the line. If the line has no non-<blank> non-<newline> characters, the current column shall be set to the last display line column on which is displayed any part of the last non-<newline>



86806 character in the line. If the line is empty, the current column shall be set to column position  
86807 1.

86808 The length of lines in the edit buffer may be limited to {LINE\_MAX} bytes. In open and visual  
86809 mode, the length of lines in the edit buffer may be limited to the number of characters that will  
86810 fit in the display. If either limit is exceeded during editing, an error message shall be written. If  
86811 either limit is exceeded by a line read in from a file, an error message shall be written and the  
86812 edit session may be terminated.

86813 If the editor stops running due to any reason other than a user command, and the edit buffer has  
86814 been modified since the last complete write, it shall be equivalent to a SIGHUP asynchronous  
86815 event. If the system crashes, it shall be equivalent to a SIGHUP asynchronous event.

86816 During initialization (before the first file is copied into the edit buffer or any user commands  
86817 from the terminal are processed) the following shall occur:

- 86818 1. If the environment variable *EXINIT* is set, the editor shall execute the *ex* commands  
86819 contained in that variable.
- 86820 2. If the *EXINIT* variable is not set, and all of the following are true:
  - 86821 a. The *HOME* environment variable is not null and not empty.
  - 86822 b. The file **.exrc** in the directory referred to by the *HOME* environment variable:
    - 86823 i. Exists
    - 86824 ii. Is owned by the same user ID as the real user ID of the process or the  
86825 process has appropriate privileges
    - 86826 iii. Is not writable by anyone other than the owner

86827 the editor shall execute the *ex* commands contained in that file.

- 86828 3. If and only if all of the following are true:
  - 86829 a. The current directory is not referred to by the *HOME* environment variable.
  - 86830 b. A command in the *EXINIT* environment variable or a command in the **.exrc** file in  
86831 the directory referred to by the *HOME* environment variable sets the editor option  
86832 **exrc**.
  - 86833 c. The **.exrc** file in the current directory:
    - 86834 i. Exists
    - 86835 ii. Is owned by the same user ID as the real user ID of the process, or by one of  
86836 a set of implementation-defined user IDs
    - 86837 iii. Is not writable by anyone other than the owner

86838 the editor shall attempt to execute the *ex* commands contained in that file.

86839 Lines in any **.exrc** file that are blank lines shall be ignored. If any **.exrc** file exists, but is not read  
86840 for ownership or permission reasons, it shall be an error.

86841 After the *EXINIT* variable and any **.exrc** files are processed, the first file specified by the user  
86842 shall be edited, as follows:

- 86843 1. If the user specified the **-t** option, the effect shall be as if the *ex tag* command was entered  
86844 with the specified argument, with the exception that if tag processing does not result in a  
86845 file to edit, the effect shall be as described in step 3. below.

2. Otherwise, if the user specified any command line *file* arguments, the effect shall be as if the *ex* **edit** command was entered with the first of those arguments as its *file* argument.
3. Otherwise, the effect shall be as if the *ex* **edit** command was entered with a nonexistent filename as its *file* argument. It is unspecified whether this action shall set the current pathname. In an implementation where this action does not set the current pathname, any editor command using the current pathname shall fail until an editor command sets the current pathname.

If the **-r** option was specified, the first time a file in the initial argument list or a file specified by the **-t** option is edited, if recovery information has previously been saved about it, that information shall be recovered and the editor shall behave as if the contents of the edit buffer have already been modified. If there are multiple instances of the file to be recovered, the one most recently saved shall be recovered, and an informational message that there are previous versions of the file that can be recovered shall be written. If no recovery information about a file is available, an informational message to this effect shall be written, and the edit shall proceed as usual.

If the **-c** option was specified, the first time a file that already exists (including a file that might not exist but for which recovery information is available, when the **-r** option is specified) replaces or initializes the contents of the edit buffer, the current line shall be set to the last line of the edit buffer, the current column shall be set to non-`<blank>`, and the *ex* commands specified with the **-c** option shall be executed. In this case, the current line and current column shall not be set as described for the command associated with the replacement or initialization of the edit buffer contents. However, if the **-t** option or a **tag** command is associated with this action, the **-c** option commands shall be executed and then the movement to the tag shall be performed.

The current argument list shall initially be set to the filenames specified by the user on the command line. If no filenames are specified by the user, the current argument list shall be empty. If the **-t** option was specified, it is unspecified whether any filename resulting from tag processing shall be prepended to the current argument list. In the case where the filename is added as a prefix to the current argument list, the current argument list reference shall be set to that filename. In the case where the filename is not added as a prefix to the current argument list, the current argument list reference shall logically be located before the first of the filenames specified on the command line (for example, a subsequent *ex* **next** command shall edit the first filename from the command line). If the **-t** option was not specified, the current argument list reference shall be to the first of the filenames on the command line.

### Addressing in *ex*

Addressing in *ex* relates to the current line and the current column; the address of a line is its 1-based line number, the address of a column is its 1-based count from the beginning of the line. Generally, the current line is the last line affected by a command. The current line number is the address of the current line. In each command description, the effect of the command on the current line number and the current column is described.

Addresses are constructed as follows:

1. The character `'.'` (period) shall address the current line.
2. The character `'$'` shall address the last line of the edit buffer.
3. The positive decimal number *n* shall address the *n*th line of the edit buffer.
4. The address `"'x"` refers to the line marked with the mark name character `'x'`, which shall be a lowercase letter from the portable character set, the backquote character, or the single-quote character. It shall be an error if the line that was marked is not currently

present in the edit buffer or the mark has not been set. Lines can be marked with the *ex mark* or *k* commands, or the *vi m* command.

5. A regular expression enclosed by <slash> characters ('/') shall address the first line found by searching forwards from the line following the current line toward the end of the edit buffer and stopping at the first line for which the line excluding the terminating <newline> matches the regular expression. As stated in [Regular Expressions in ex](#) (on page 2699), an address consisting of a null regular expression delimited by <slash> characters ("/") shall address the next line for which the line excluding the terminating <newline> matches the last regular expression encountered. In addition, the second <slash> can be omitted at the end of a command line. If the **wrapscan** edit option is set, the search shall wrap around to the beginning of the edit buffer and continue up to and including the current line, so that the entire edit buffer is searched. Within the regular expression, the sequence "\/" shall represent a literal <slash> instead of the regular expression delimiter.
6. A regular expression enclosed in <question-mark> characters ('?') shall address the first line found by searching backwards from the line preceding the current line toward the beginning of the edit buffer and stopping at the first line for which the line excluding the terminating <newline> matches the regular expression. An address consisting of a null regular expression delimited by <question-mark> characters ("??") shall address the previous line for which the line excluding the terminating <newline> matches the last regular expression encountered. In addition, the second <question-mark> can be omitted at the end of a command line. If the **wrapscan** edit option is set, the search shall wrap around from the beginning of the edit buffer to the end of the edit buffer and continue up to and including the current line, so that the entire edit buffer is searched. Within the regular expression, the sequence "\?" shall represent a literal <question-mark> instead of the RE delimiter.
7. A <plus-sign> ('+') or a minus-sign ('-') followed by a decimal number shall address the current line plus or minus the number. A '+' or '-' not followed by a decimal number shall address the current line plus or minus 1.

Addresses can be followed by zero or more address offsets, optionally <blank>-separated. Address offsets are constructed as follows:

1. A '+' or '-' immediately followed by a decimal number shall add (subtract) the indicated number of lines to (from) the address. A '+' or '-' not followed by a decimal number shall add (subtract) 1 to (from) the address.
2. A decimal number shall add the indicated number of lines to the address.

It shall not be an error for an intermediate address value to be less than zero or greater than the last line in the edit buffer. It shall be an error for the final address value to be less than zero or greater than the last line in the edit buffer.

Commands take zero, one, or two addresses; see the descriptions of *1addr* and *2addr* in [Command Descriptions in ex](#) (on page 2675). If more than the required number of addresses are provided to a command that requires zero addresses, it shall be an error. Otherwise, if more than the required number of addresses are provided to a command, the addresses specified first shall be evaluated and then discarded until the maximum number of valid addresses remain.

Addresses shall be separated from each other by a <comma> (',') or a <semicolon> (';'). If no address is specified before or after a <comma> or <semicolon> separator, it shall be as if the address of the current line was specified before or after the separator. In the case of a <semicolon> separator, the current line ('.') shall be set to the first address, and only then will

the next address be calculated. This feature can be used to determine the starting line for forwards and backwards searches (see rules 5. and 6.).

A <percent-sign> ( ' % ' ) shall be equivalent to entering the two addresses "1, \$".

Any delimiting <blank> characters between addresses, address separators, or address offsets shall be discarded.

### Command Line Parsing in ex

The following symbol is used in this and following sections to describe parsing behavior:

*escape* If a character is referred to as "<backslash>-escaped" or "<control>-V-escaped", it shall mean that the character acquired or lost a special meaning by virtue of being preceded, respectively, by a <backslash> or <control>-V character. Unless otherwise specified, the escaping character shall be discarded at that time and shall not be further considered for any purpose.

Command-line parsing shall be done in the following steps. For each step, characters already evaluated shall be ignored; that is, the phrase "leading character" refers to the next character that has not yet been evaluated.

1. Leading <colon> characters shall be skipped.
2. Leading <blank> characters shall be skipped.
3. If the leading character is a double-quote character, the characters up to and including the next non-<backslash>-escaped <newline> shall be discarded, and any subsequent characters shall be parsed as a separate command.
4. Leading characters that can be interpreted as addresses shall be evaluated; see [Addressing in ex](#) (on page 2668).
5. Leading <blank> characters shall be skipped.
6. If the next character is a <vertical-line> character or a <newline>:
  - a. If the next character is a <newline>:
    - i. If *ex* is in open or visual mode, the current line shall be set to the last address specified, if any.
    - ii. Otherwise, if the last command was terminated by a <vertical-line> character, no action shall be taken; for example, the command "||<newline>" shall execute two implied commands, not three.
    - iii. Otherwise, step 6.b. shall apply.
  - b. Otherwise, the implied command shall be the **print** command. The last #, **p**, and **l** flags specified to any *ex* command shall be remembered and shall apply to this implied command. Executing the *ex* **number**, **print**, or **list** command shall set the remembered flags to #, nothing, and **l**, respectively, plus any other flags specified for that execution of the **number**, **print**, or **list** command.

If *ex* is not currently performing a **global** or **v** command, and no address or count is specified, the current line shall be incremented by 1 before the command is executed. If incrementing the current line would result in an address past the last line in the edit buffer, the command shall fail, and the increment shall not happen.

- c. The <newline> or <vertical-line> character shall be discarded and any subsequent characters shall be parsed as a separate command.
7. The command name shall be comprised of the next character (if the character is not alphabetic), or the next character and any subsequent alphabetic characters (if the character is alphabetic), with the following exceptions:
- a. Commands that consist of any prefix of the characters in the command name **delete**, followed immediately by any of the characters 'l', 'p', '+', '-', or '#' shall be interpreted as a **delete** command, followed by a <blank>, followed by the characters that were not part of the prefix of the **delete** command. The maximum number of characters shall be matched to the command name **delete**; for example, "del" shall not be treated as "de" followed by the flag l.
  - b. Commands that consist of the character 'k', followed by a character that can be used as the name of a mark, shall be equivalent to the mark command followed by a <blank>, followed by the character that followed the 'k'.
  - c. Commands that consist of the character 's', followed by characters that could be interpreted as valid options to the s command, shall be the equivalent of the s command, without any pattern or replacement values, followed by a <blank>, followed by the characters after the 's'.
8. The command name shall be matched against the possible command names, and a command name that contains a prefix matching the characters specified by the user shall be the executed command. In the case of commands where the characters specified by the user could be ambiguous, the executed command shall be as follows:

|    |        |    |       |    |       |
|----|--------|----|-------|----|-------|
| a  | append | n  | next  | t  | t     |
| c  | change | p  | print | u  | undo  |
| ch | change | pr | print | un | undo  |
| e  | edit   | r  | read  | v  | v     |
| m  | move   | re | read  | w  | write |
| ma | mark   | s  | s     |    |       |

- Implementation extensions with names causing similar ambiguities shall not be checked for a match until all possible matches for commands specified by POSIX.1-2008 have been checked.
9. If the command is a ! command, or if the command is a **read** command followed by zero or more <blank> characters and a !, or if the command is a **write** command followed by one or more <blank> characters and a !, the rest of the command shall include all characters up to a non-<backslash>-escaped <newline>. The <newline> shall be discarded and any subsequent characters shall be parsed as a separate *ex* command.
10. Otherwise, if the command is an **edit**, **ex**, or **next** command, or a **visual** command while in open or visual mode, the next part of the command shall be parsed as follows:
- a. Any '!' character immediately following the command shall be skipped and be part of the command.
  - b. Any leading <blank> characters shall be skipped and be part of the command.
  - c. If the next character is a '+', characters up to the first non-<backslash>-escaped <newline> or non-<backslash>-escaped <blank> shall be skipped and be part of the command.

- 87023 d. The rest of the command shall be determined by the steps specified in paragraph  
87024 12.
- 87025 11. Otherwise, if the command is a **global**, **open**, **s**, or **v** command, the next part of the  
87026 command shall be parsed as follows:
- 87027 a. Any leading <blank> characters shall be skipped and be part of the command.
- 87028 b. If the next character is not an alphanumeric, double-quote, <newline>,  
87029 <backslash>, or <vertical-line> character:
- 87030 i. The next character shall be used as a command delimiter.
- 87031 ii. If the command is a **global**, **open**, or **v** command, characters up to the first  
87032 non-<backslash>-escaped <newline>, or first non-<backslash>-escaped  
87033 delimiter character, shall be skipped and be part of the command.
- 87034 iii. If the command is an **s** command, characters up to the first  
87035 non-<backslash>-escaped <newline>, or second non-<backslash>-escaped  
87036 delimiter character, shall be skipped and be part of the command.
- 87037 c. If the command is a **global** or **v** command, characters up to the first  
87038 non-<backslash>-escaped <newline> shall be skipped and be part of the  
87039 command.
- 87040 d. Otherwise, the rest of the command shall be determined by the steps specified in  
87041 paragraph 12.
- 87042 12. Otherwise:
- 87043 a. If the command was a **map**, **unmap**, **abbreviate**, or **unabbreviate** command,  
87044 characters up to the first non-<control>-V-escaped <newline>, <vertical-line>, or  
87045 double-quote character shall be skipped and be part of the command.
- 87046 b. Otherwise, characters up to the first non-<backslash>-escaped <newline>,  
87047 <vertical-line>, or double-quote character shall be skipped and be part of the  
87048 command.
- 87049 c. If the command was an **append**, **change**, or **insert** command, and the step 12.b.  
87050 ended at a <vertical-line> character, any subsequent characters, up to the next  
87051 non-<backslash>-escaped <newline> shall be used as input text to the command.
- 87052 d. If the command was ended by a double-quote character, all subsequent characters,  
87053 up to the next non-<backslash>-escaped <newline>, shall be discarded.
- 87054 e. The terminating <newline> or <vertical-line> character shall be discarded and any  
87055 subsequent characters shall be parsed as a separate *ex* command.

87056 Command arguments shall be parsed as described by the Synopsis and Description of each  
87057 individual *ex* command. This parsing shall not be <blank>-sensitive, except for the **!** argument,  
87058 which must follow the command name without intervening <blank> characters, and where it  
87059 would otherwise be ambiguous. For example, *count* and *flag* arguments need not be  
87060 <blank>-separated because "d22p" is not ambiguous, but *file* arguments to the *ex* **next**  
87061 command must be separated by one or more <blank> characters. Any <blank> in command  
87062 arguments for the **abbreviate**, **unabbreviate**, **map**, and **unmap** commands can be  
87063 <control>-V-escaped, in which case the <blank> shall not be used as an argument delimiter. Any  
87064 <blank> in the command argument for any other command can be <backslash>-escaped, in  
87065 which case that <blank> shall not be used as an argument delimiter.

87066 Within command arguments for the **abbreviate**, **unabbreviate**, **map**, and **unmap** commands,

any character can be <control>-V-escaped. All such escaped characters shall be treated literally and shall have no special meaning. Within command arguments for all other *ex* commands that are not regular expressions or replacement strings, any character that would otherwise have a special meaning can be <backslash>-escaped. Escaped characters shall be treated literally, without special meaning as shell expansion characters or '!', '%', and '#' expansion characters. See [Regular Expressions in ex](#) (on page 2699) and [Replacement Strings in ex](#) (on page 2700) for descriptions of command arguments that are regular expressions or replacement strings.

Non-<backslash>-escaped '%' characters appearing in *file* arguments to any *ex* command shall be replaced by the current pathname; unescaped '#' characters shall be replaced by the alternate pathname. It shall be an error if '%' or '#' characters appear unescaped in an argument and their corresponding values are not set.

Non-<backslash>-escaped '!' characters in the arguments to either the *ex* ! command or the open and visual mode ! command, or in the arguments to the *ex* **read** command, where the first non-<blank> after the command name is a '!' character, or in the arguments to the *ex* **write** command where the command name is followed by one or more <blank> characters and the first non-<blank> after the command name is a '!' character, shall be replaced with the arguments to the last of those three commands as they appeared after all unescaped '%', '#', and '!' characters were replaced. It shall be an error if '!' characters appear unescaped in one of these commands and there has been no previous execution of one of these commands.

If an error occurs during the parsing or execution of an *ex* command:

- An informational message to this effect shall be written. Execution of the *ex* command shall stop, and the cursor (for example, the current line and column) shall not be further modified.
- If the *ex* command resulted from a map expansion, all characters from that map expansion shall be discarded, except as otherwise specified by the **map** command.
- Otherwise, if the *ex* command resulted from the processing of an *EXINIT* environment variable, a **.exrc** file, a **:source** command, a **-c** option, or a **+command** specified to an *ex* **edit**, **ex**, **next**, or **visual** command, no further commands from the source of the commands shall be executed.
- Otherwise, if the *ex* command resulted from the execution of a buffer or a **global** or **v** command, no further commands caused by the execution of the buffer or the **global** or **v** command shall be executed.
- Otherwise, if the *ex* command was not terminated by a <newline>, all characters up to and including the next non-<backslash>-escaped <newline> shall be discarded.

## Input Editing in ex

The following symbol is used in this and the following sections to specify command actions:

*word* In the POSIX locale, a word consists of a maximal sequence of letters, digits, and underscores, delimited at both ends by characters other than letters, digits, or underscores, or by the beginning or end of a line or the edit buffer.

When accepting input characters from the user, in either *ex* command mode or *ex* text input mode, *ex* shall enable canonical mode input processing, as defined in the System Interfaces volume of POSIX.1-2008.

87110 If in *ex* text input mode:

- 87111 1. If the **number** edit option is set, *ex* shall prompt for input using the line number that  
87112 would be assigned to the line if it is entered, in the format specified for the *ex* **number**  
87113 command.
- 87114 2. If the **autoindent** edit option is set, *ex* shall prompt for input using **autoindent** characters,  
87115 as described by the **autoindent** edit option. **autoindent** characters shall follow the line  
87116 number, if any.

87117 If in *ex* command mode:

- 87118 1. If the **prompt** edit option is set, input shall be prompted for using a single ' : ' character;  
87119 otherwise, there shall be no prompt.

87120 The input characters in the following sections shall have the following effects on the input line.

## 87121 Scroll

87122 *Synopsis:*      `eof`

87123 See the description of the *stty* `eof` character in *stty*.

87124 If in *ex* command mode:

87125 If the `eof` character is the first character entered on the line, the line shall be evaluated as if  
87126 it contained two characters: a <control>-D and a <newline>.

87127 Otherwise, the `eof` character shall have no special meaning.

87128 If in *ex* text input mode:

87129 If the cursor follows an **autoindent** character, the **autoindent** characters in the line shall be  
87130 modified so that a part of the next text input character will be displayed on the first  
87131 column in the line after the previous **shiftwidth** edit option column boundary, and the  
87132 user shall be prompted again for input for the same line.

87133 Otherwise, if the cursor follows a ' 0 ', which follows an **autoindent** character, and the ' 0 '  
87134 was the previous text input character, the ' 0 ' and all **autoindent** characters in the line  
87135 shall be discarded, and the user shall be prompted again for input for the same line.

87136 Otherwise, if the cursor follows a ' ^ ', which follows an **autoindent** character, and the ' ^ '  
87137 was the previous text input character, the ' ^ ' and all **autoindent** characters in the line  
87138 shall be discarded, and the user shall be prompted again for input for the same line. In  
87139 addition, the **autoindent** level for the next input line shall be derived from the same line  
87140 from which the **autoindent** level for the current input line was derived.

87141 Otherwise, if there are no **autoindent** or text input characters in the line, the `eof` character  
87142 shall be discarded.

87143 Otherwise, the `eof` character shall have no special meaning.



87144        **<newline>**

87145        *Synopsis:*        <newline>  
 87146                    <control>-J

87147        If in *ex* command mode:

87148                    Cause the command line to be parsed; <control>-J shall be mapped to the <newline> for  
 87149                    this purpose.

87150        If in *ex* text input mode:

87151                    Terminate the current line. If there are no characters other than **autoindent** characters on  
 87152                    the line, all characters on the line shall be discarded.

87153                    Prompt for text input on a new line after the current line. If the **autoindent** edit option is  
 87154                    set, an appropriate number of **autoindent** characters shall be added as a prefix to the line  
 87155                    as described by the *ex* **autoindent** edit option.

87156        **<backslash>**

87157        *Synopsis:*        <backslash>

87158        Allow the entry of a subsequent <newline> or <control>-J as a literal character, removing any  
 87159        special meaning that it may have to the editor during text input mode. The <backslash>  
 87160        character shall be retained and evaluated when the command line is parsed, or retained and  
 87161        included when the input text becomes part of the edit buffer.

87162        **<control>-V**

87163        *Synopsis:*        <control>-V

87164        Allow the entry of any subsequent character as a literal character, removing any special meaning  
 87165        that it may have to the editor during text input mode. The <control>-V character shall be  
 87166        discarded before the command line is parsed or the input text becomes part of the edit buffer.

87167        If the “literal next” functionality is performed by the underlying system, it is implementation-  
 87168        defined whether a character other than <control>-V performs this function.

87169        **<control>-W**

87170        *Synopsis:*        <control>-W

87171        Discard the <control>-W, and the word previous to it in the input line, including any <blank>  
 87172        characters following the word and preceding the <control>-W. If the “word erase” functionality  
 87173        is performed by the underlying system, it is implementation-defined whether a character other  
 87174        than <control>-W performs this function.

87175        **Command Descriptions in ex**

87176        The following symbols are used in this section to represent command modifiers. Some of these  
 87177        modifiers can be omitted, in which case the specified defaults shall be used.

87178        *laddr*        A single line address, given in any of the forms described in [Addressing in ex](#) (on  
 87179        page 2668); the default shall be the current line (‘.’), unless otherwise specified.

87180        If the line address is zero, it shall be an error, unless otherwise specified in the  
 87181        following command descriptions.

|       |               |                                                                                                                                                |
|-------|---------------|------------------------------------------------------------------------------------------------------------------------------------------------|
| 87182 |               | If the edit buffer is empty, and the address is specified with a command other than                                                            |
| 87183 |               | <b>=</b> , <b>append</b> , <b>insert</b> , <b>open</b> , <b>put</b> , <b>read</b> , or <b>visual</b> , or the address is not zero, it shall be |
| 87184 |               | an error.                                                                                                                                      |
| 87185 | <i>2addr</i>  | Two addresses specifying an inclusive range of lines. If no addresses are specified,                                                           |
| 87186 |               | the default for <i>2addr</i> shall be the current line only (" . , . "), unless otherwise                                                      |
| 87187 |               | specified in the following command descriptions. If one address is specified, <i>2addr</i>                                                     |
| 87188 |               | shall specify that line only, unless otherwise specified in the following command                                                              |
| 87189 |               | descriptions.                                                                                                                                  |
| 87190 |               | It shall be an error if the first address is greater than the second address.                                                                  |
| 87191 |               | If the edit buffer is empty, and the two addresses are specified with a command                                                                |
| 87192 |               | other than the <b>!</b> , <b>write</b> , <b>wq</b> , or <b>xit</b> commands, or either address is not zero, it shall                           |
| 87193 |               | be an error.                                                                                                                                   |
| 87194 | <i>count</i>  | A positive decimal number. If <i>count</i> is specified, it shall be equivalent to specifying                                                  |
| 87195 |               | an additional address to the command, unless otherwise specified by the following                                                              |
| 87196 |               | command descriptions. The additional address shall be equal to the last address                                                                |
| 87197 |               | specified to the command (either explicitly or by default) plus <i>count</i> –1.                                                               |
| 87198 |               | If this would result in an address greater than the last line of the edit buffer, it shall                                                     |
| 87199 |               | be corrected to equal the last line of the edit buffer.                                                                                        |
| 87200 | <i>flags</i>  | One or more of the characters <b>' + '</b> , <b>' - '</b> , <b>' # '</b> , <b>' p '</b> , or <b>' l '</b> (ell). The flag characters           |
| 87201 |               | can be <blank>-separated, and in any order or combination. The characters <b>' # '</b> ,                                                       |
| 87202 |               | <b>' p '</b> , and <b>' l '</b> shall cause lines to be written in the format specified by the <b>print</b>                                    |
| 87203 |               | command with the specified <i>flags</i> .                                                                                                      |
| 87204 |               | The lines to be written are as follows:                                                                                                        |
| 87205 |               | 1. All edit buffer lines written during the execution of the <i>ex</i> <b>&amp;</b> , <b>~</b> , <b>list</b> , <b>number</b> ,                 |
| 87206 |               | <b>open</b> , <b>print</b> , <b>s</b> , <b>visual</b> , and <b>z</b> commands shall be written as specified by <i>flags</i> .                  |
| 87207 |               | 2. After the completion of an <i>ex</i> command with a flag as an argument, the                                                                |
| 87208 |               | current line shall be written as specified by <i>flags</i> , unless the current line was                                                       |
| 87209 |               | the last line written by the command.                                                                                                          |
| 87210 |               | The characters <b>' + '</b> and <b>' - '</b> cause the value of the current line after the execution                                           |
| 87211 |               | of the <i>ex</i> command to be adjusted by the offset address as described in <a href="#">Addressing</a>                                       |
| 87212 |               | <a href="#">in ex</a> (on page 2668). This adjustment shall occur before the current line is written                                           |
| 87213 |               | as described in 2. above.                                                                                                                      |
| 87214 |               | The default for <i>flags</i> shall be none.                                                                                                    |
| 87215 | <i>buffer</i> | One of a number of named areas for holding text. The named buffers are specified                                                               |
| 87216 |               | by the alphanumeric characters of the POSIX locale. There shall also be one                                                                    |
| 87217 |               | "unnamed" buffer. When no buffer is specified for editor commands that use a                                                                   |
| 87218 |               | buffer, the unnamed buffer shall be used. Commands that store text into buffers                                                                |
| 87219 |               | shall store the text as it was before the command took effect, and shall store text                                                            |
| 87220 |               | occurring earlier in the file before text occurring later in the file, regardless of how                                                       |
| 87221 |               | the text region was specified. Commands that store text into buffers shall store the                                                           |
| 87222 |               | text into the unnamed buffer as well as any specified buffer.                                                                                  |
| 87223 |               | In <i>ex</i> commands, buffer names are specified as the name by itself. In open or visual                                                     |
| 87224 |               | mode commands the name is preceded by a double-quote ( <b>' "</b> ) character.                                                                 |
| 87225 |               | If the specified buffer name is an uppercase character, and the buffer contents are                                                            |
| 87226 |               | to be modified, the buffer shall be appended to rather than being overwritten. If                                                              |

the buffer is not being modified, specifying the buffer name in lowercase and uppercase shall have identical results.

There shall also be buffers named by the numbers 1 through 9. In open and visual mode, if a region of text including characters from more than a single line is being modified by the *vi* **c** or **d** commands, the motion character associated with the **c** or **d** commands specifies that the buffer text shall be in line mode, or the commands **%**, **'**, **/**, **?**, **(**, **)**, **N**, **n**, **{**, or **}** are used to define a region of text for the **c** or **d** commands, the contents of buffers 1 through 8 shall be moved into the buffer named by the next numerically greater value, the contents of buffer 9 shall be discarded, and the region of text shall be copied into buffer 1. This shall be in addition to copying the text into a user-specified buffer or unnamed buffer, or both. Numeric buffers can be specified as a source buffer for open and visual mode commands; however, specifying a numeric buffer as the write target of an open or visual mode command shall have unspecified results.

The text of each buffer shall have the characteristic of being in either line or character mode. Appending text to a non-empty buffer shall set the mode to match the characteristic of the text being appended. Appending text to a buffer shall cause the creation of at least one additional line in the buffer. All text stored into buffers by *ex* commands shall be in line mode. The *ex* commands that use buffers as the source of text specify individually how buffers of different modes are handled. Each open or visual mode command that uses buffers for any purpose specifies individually the mode of the text stored into the buffer and how buffers of different modes are handled.

*file* Command text used to derive a pathname. The default shall be the current pathname, as defined previously, in which case, if no current pathname has yet been established it shall be an error, except where specifically noted in the individual command descriptions that follow. If the command text contains any of the characters **~**, **{**, **[**, **\***, **?**, **\$**, **"**, backquote, single-quote, and **<backslash>**, it shall be subjected to the process of "shell expansions", as described below; if more than a single pathname results and the command expects only one, it shall be an error.

The process of shell expansions in the editor shall be done as follows. The *ex* utility shall pass two arguments to the program named by the shell edit option; the first shall be **-c**, and the second shall be the string **"echo"** and the command text as a single argument. The standard output and standard error of that command shall replace the command text.

**!** A character that can be appended to the command name to modify its operation, as detailed in the individual command descriptions. With the exception of the *ex* **read**, **write**, and **!** commands, the **'!'** character shall only act as a modifier if there are no **<blank>** characters between it and the command name.

*remembered search direction*

The *vi* commands **N** and **n** begin searching in a forwards or backwards direction in the edit buffer based on a remembered search direction, which is initially unset, and is set by the *ex* **global**, **v**, **s**, and **tag** commands, and the *vi* **/** and **?** commands.

## Abbreviate

*Synopsis:*     ab[*breviate*] [*lhs rhs*]

If *lhs* and *rhs* are not specified, write the current list of abbreviations and do nothing more.

Implementations may restrict the set of characters accepted in *lhs* or *rhs*, except that printable characters and <blank> characters shall not be restricted. Additional restrictions shall be implementation-defined.

In both *lhs* and *rhs*, any character may be escaped with a <control>-V, in which case the character shall not be used to delimit *lhs* from *rhs*, and the escaping <control>-V shall be discarded.

In open and visual text input mode, if a non-word or <ESC> character that is not escaped by a <control>-V character is entered after a word character, a check shall be made for a set of characters matching *lhs*, in the text input entered during this command. If it is found, the effect shall be as if *rhs* was entered instead of *lhs*.

The set of characters that are checked is defined as follows:

1. If there are no characters inserted before the word and non-word or <ESC> characters that triggered the check, the set of characters shall consist of the word character.
2. If the character inserted before the word and non-word or <ESC> characters that triggered the check is a word character, the set of characters shall consist of the characters inserted immediately before the triggering characters that are word characters, plus the triggering word character.
3. If the character inserted before the word and non-word or <ESC> characters that triggered the check is not a word character, the set of characters shall consist of the characters that were inserted before the triggering characters that are neither <blank> characters nor word characters, plus the triggering word character.

It is unspecified whether the *lhs* argument entered for the *ex* **abbreviate** and **unabbreviate** commands is replaced in this fashion. Regardless of whether or not the replacement occurs, the effect of the command shall be as if the replacement had not occurred.

*Current line:* Unchanged.

*Current column:* Unchanged.

## Append

*Synopsis:*     [*laddr*] a[*ppend*][!]

Enter *ex* text input mode; the input text shall be placed after the specified line. If line zero is specified, the text shall be placed at the beginning of the edit buffer.

This command shall be affected by the **number** and **autoindent** edit options; following the command name with '!' shall cause the **autoindent** edit option setting to be toggled for the duration of this command only.

*Current line:* Set to the last input line; if no lines were input, set to the specified line, or to the first line of the edit buffer if a line of zero was specified, or zero if the edit buffer is empty.

*Current column:* Set to non-<blank>.

**Arguments**

*Synopsis:*     ar[gs]

Write the current argument list, with the current argument-list entry, if any, between ' [' and ' ] ' characters.

*Current line:* Unchanged.

*Current column:* Unchanged.

**Change**

*Synopsis:*     [2addr] c[hange] [!] [count]

Enter *ex* text input mode; the input text shall replace the specified lines. The specified lines shall be copied into the unnamed buffer, which shall become a line mode buffer.

This command shall be affected by the **number** and **autoindent** edit options; following the command name with ' ! ' shall cause the **autoindent** edit option setting to be toggled for the duration of this command only.

*Current line:* Set to the last input line; if no lines were input, set to the line before the first address, or to the first line of the edit buffer if there are no lines preceding the first address, or to zero if the edit buffer is empty.

*Current column:* Set to non-<blank>.

**Change Directory**

*Synopsis:*     chd[ir] [!] [directory]  
              cd[!] [directory]

Change the current working directory to *directory*.

If no *directory* argument is specified, and the *HOME* environment variable is set to a non-null and non-empty value, *directory* shall default to the value named in the *HOME* environment variable. If the *HOME* environment variable is empty or is undefined, the default value of *directory* is implementation-defined.

If no ' ! ' is appended to the command name, and the edit buffer has been modified since the last complete write, and the current pathname does not begin with a ' / ', it shall be an error.

*Current line:* Unchanged.

*Current column:* Unchanged.

**Copy**

*Synopsis:*     [2addr] co[py] laddr [flags]  
              [2addr] t laddr [flags]

Copy the specified lines after the specified destination line; line zero specifies that the lines shall be placed at the beginning of the edit buffer.

*Current line:* Set to the last line copied.

*Current column:* Set to non-<blank>.

**Delete**

*Synopsis:*     `[2addr] d[ele]te [buffer] [count] [flags]`

Delete the specified lines into a buffer (defaulting to the unnamed buffer), which shall become a line-mode buffer.

Flags can immediately follow the command name; see [Command Line Parsing in ex](#) (on page 2670).

*Current line:* Set to the line following the deleted lines, or to the last line in the edit buffer if that line is past the end of the edit buffer, or to zero if the edit buffer is empty.

*Current column:* Set to non-<blank>.

**Edit**

*Synopsis:*     `e[dit] [!] [+command] [file]`  
               `ex [!] [+command] [file]`

If no '!' is appended to the command name, and the edit buffer has been modified since the last complete write, it shall be an error.

If *file* is specified, replace the current contents of the edit buffer with the current contents of *file*, and set the current pathname to *file*. If *file* is not specified, replace the current contents of the edit buffer with the current contents of the file named by the current pathname. If for any reason the current contents of the file cannot be accessed, the edit buffer shall be empty.

The *+command* option shall be <blank>-delimited; <blank> characters within the *+command* can be escaped by preceding them with a <backslash> character. The *+command* shall be interpreted as an *ex* command immediately after the contents of the edit buffer have been replaced and the current line and column have been set.

If the edit buffer is empty:

*Current line:* Set to 0.

*Current column:* Set to 1.

Otherwise, if executed while in *ex* command mode or if the *+command* argument is specified:

*Current line:* Set to the last line of the edit buffer.

*Current column:* Set to non-<blank>.

Otherwise, if *file* is omitted or results in the current pathname:

*Current line:* Set to the first line of the edit buffer.

*Current column:* Set to non-<blank>.

Otherwise, if *file* is the same as the last file edited, the line and column shall be set as follows; if the file was previously edited, the line and column may be set as follows:

*Current line:* Set to the last value held when that file was last edited. If this value is not a valid line in the new edit buffer, set to the first line of the edit buffer.

*Current column:* If the current line was set to the last value held when the file was last edited, set to the last value held when the file was last edited. Otherwise, or if the last value is not a valid column in the new edit buffer, set to non-<blank>.

Otherwise:

*Current line*: Set to the first line of the edit buffer.

*Current column*: Set to non-<blank>.

## File

*Synopsis:*     *f*[*file*][*file*]

If a *file* argument is specified, the alternate pathname shall be set to the current pathname, and the current pathname shall be set to *file*.

Write an informational message. If the file has a current pathname, it shall be included in this message; otherwise, the message shall indicate that there is no current pathname. If the edit buffer contains lines, the current line number and the number of lines in the edit buffer shall be included in this message; otherwise, the message shall indicate that the edit buffer is empty. If the edit buffer has been modified since the last complete write, this fact shall be included in this message. If the **readonly** edit option is set, this fact shall be included in this message. The message may contain other unspecified information.

*Current line*: Unchanged.

*Current column*: Unchanged.

## Global

*Synopsis:*     [*2addr*] *g*[*lobal*] /*pattern*/ [*commands*]  
                   [*2addr*] *v* /*pattern*/ [*commands*]

The optional **'!'** character after the **global** command shall be the same as executing the **v** command.

If *pattern* is empty (for example, *"/"*) or not specified, the last regular expression used in the editor command shall be used as the *pattern*. The *pattern* can be delimited by <slash> characters (shown in the Synopsis), as well as any non-alphanumeric or non-<blank> other than <backslash>, <vertical-line>, <newline>, or double-quote.

If no lines are specified, the lines shall default to the entire file.

The **global** and **v** commands are logically two-pass operations. First, mark the lines within the specified lines for which the line excluding the terminating <newline> matches (**global**) or does not match (**v** or **global!**) the specified pattern. Second, execute the *ex* commands given by *commands*, with the current line (**'.'**) set to each marked line. If an error occurs during this process, or the contents of the edit buffer are replaced (for example, by the *ex* **:edit** command) an error message shall be written and no more commands resulting from the execution of this command shall be processed.

Multiple *ex* commands can be specified by entering multiple commands on a single line using a <vertical-line> to delimit them, or one per line, by escaping each <newline> with a <backslash>.

If no commands are specified:

1. If in *ex* command mode, it shall be as if the **print** command were specified.
2. Otherwise, no command shall be executed.

For the **append**, **change**, and **insert** commands, the input text shall be included as part of the command, and the terminating <period> can be omitted if the command ends the list of commands. The **open** and **visual** commands can be specified as one of the commands, in which

case each marked line shall cause the editor to enter open or visual mode. If open or visual mode is exited using the *vi* **Q** command, the current line shall be set to the next marked line, and open or visual mode reentered, until the list of marked lines is exhausted.

The **global**, **v**, and **undo** commands cannot be used in *commands*. Marked lines may be deleted by commands executed for lines occurring earlier in the file than the marked lines. In this case, no commands shall be executed for the deleted lines.

If the remembered search direction is not set, the **global** and **v** commands shall set it to forward.

The **autoprint** and **autoindent** edit options shall be inhibited for the duration of the **g** or **v** command.

*Current line*: If no commands executed, set to the last marked line. Otherwise, as specified for the executed *ex* commands.

*Current column*: If no commands are executed, set to non-<blank>; otherwise, as specified for the individual *ex* commands.

## Insert

*Synopsis*:     [*laddr*] i[nsert][!]

Enter *ex* text input mode; the input text shall be placed before the specified line. If the line is zero or 1, the text shall be placed at the beginning of the edit buffer.

This command shall be affected by the **number** and **autoindent** edit options; following the command name with '!' shall cause the **autoindent** edit option setting to be toggled for the duration of this command only.

*Current line*: Set to the last input line; if no lines were input, set to the line before the specified line, or to the first line of the edit buffer if there are no lines preceding the specified line, or zero if the edit buffer is empty.

*Current column*: Set to non-<blank>.

## Join

*Synopsis*:     [*2addr*] j[oin][!][*count*][*flags*]

If *count* is specified:

If no address was specified, the **join** command shall behave as if *2addr* were the current line and the current line plus *count* (*.*, *.* + *count*).

If one address was specified, the **join** command shall behave as if *2addr* were the specified address and the specified address plus *count* (*addr*, *addr* + *count*).

If two addresses were specified, the **join** command shall behave as if an additional address, equal to the last address plus *count* - 1 (*addr1*, *addr2*, *addr2* + *count* - 1), was specified.

If this would result in a second address greater than the last line of the edit buffer, it shall be corrected to be equal to the last line of the edit buffer.

If no *count* is specified:



87461 If no address was specified, the **join** command shall behave as if *2addr* were the current  
87462 line and the next line (*.*, *.* +1).

87463 If one address was specified, the **join** command shall behave as if *2addr* were the specified  
87464 address and the next line (*addr,addr* +1).

87465 Join the text from the specified lines together into a single line, which shall replace the specified  
87466 lines.

87467 If a '!' character is appended to the command name, the **join** shall be without modification of  
87468 any line, independent of the current locale.

87469 Otherwise, in the POSIX locale, set the current line to the first of the specified lines, and then, for  
87470 each subsequent line, proceed as follows:

- 87471 1. Discard leading <space> characters from the line to be joined.
- 87472 2. If the line to be joined is now empty, delete it, and skip steps 3 through 5.
- 87473 3. If the current line ends in a <blank>, or the first character of the line to be joined is a ' ) ' character, join the lines without further modification.
- 87474 3. If the current line ends in a <blank>, or the first character of the line to be joined is a ' ) ' character, join the lines without further modification.
- 87475 4. If the last character of the current line is a ' . ' , join the lines with two <space> characters  
87476 between them.
- 87477 5. Otherwise, join the lines with a single <space> between them.

87478 *Current line*: Set to the first line specified.

87479 *Current column*: Set to non-<blank>.

## 87480 List

87481 *Synopsis:*     [*2addr*] l[ist] [count] [flags]

87482 This command shall be equivalent to the *ex* command:

87483 [*2addr*] p[rint] [count] l[ist] [flags]

87484 See [Print](#) (on page 2687).

## 87485 Map

87486 *Synopsis:*     map[!] [*lhs rhs*]

87487 If *lhs* and *rhs* are not specified:

- 87488 1. If '!' is specified, write the current list of text input mode maps.
- 87489 2. Otherwise, write the current list of command mode maps.
- 87490 3. Do nothing more.

87491 Implementations may restrict the set of characters accepted in *lhs* or *rhs*, except that printable  
87492 characters and <blank> characters shall not be restricted. Additional restrictions shall be  
87493 implementation-defined. In both *lhs* and *rhs*, any character can be escaped with a <control>-V, in  
87494 which case the character shall not be used to delimit *lhs* from *rhs*, and the escaping <control>-V  
87495 shall be discarded.

87496 If the character '!' is appended to the **map** command name, the mapping shall be effective  
87497 during open or visual text input mode rather than **open** or **visual** command mode. This allows  
87498 *lhs* to have two different **map** definitions at the same time: one for command mode and one for

87499 text input mode.

87500 For command mode mappings:

87501 When the *lhs* is entered as any part of a *vi* command in open or visual mode (but not as

87502 part of the arguments to the command), the action shall be as if the corresponding *rhs* had

87503 been entered.

87504 If any character in the command, other than the first, is escaped using a <control>-V

87505 character, that character shall not be part of a match to an *lhs*.

87506 It is unspecified whether implementations shall support **map** commands where the *lhs* is

87507 more than a single character in length, where the first character of the *lhs* is printable.

87508 If *lhs* contains more than one character and the first character is '#', followed by a

87509 sequence of digits corresponding to a numbered function key, then when this function key

87510 is typed it shall be mapped to *rhs*. Characters other than digits following a '#' character

87511 also represent the function key named by the characters in the *lhs* following the '#' and

87512 may be mapped to *rhs*. It is unspecified how function keys are named or what function

87513 keys are supported.

87514 For text input mode mappings:

87515 When the *lhs* is entered as any part of text entered in open or visual text input modes, the

87516 action shall be as if the corresponding *rhs* had been entered.

87517 If any character in the input text is escaped using a <control>-V character, that character

87518 shall not be part of a match to an *lhs*.

87519 It is unspecified whether the *lhs* text entered for subsequent **map** or **unmap** commands is

87520 replaced with the *rhs* text for the purposes of the screen display; regardless of whether or

87521 not the display appears as if the corresponding *rhs* text was entered, the effect of the

87522 command shall be as if the *lhs* text was entered.

87523 If only part of the *lhs* is entered, it is unspecified how long the editor will wait for additional,

87524 possibly matching characters before treating the already entered characters as not matching the

87525 *lhs*.

87526 The *rhs* characters shall themselves be subject to remapping, unless otherwise specified by the

87527 **remap** edit option, except that if the characters in *lhs* occur as prefix characters in *rhs*, those

87528 characters shall not be remapped.

87529 On block-mode terminals, the mapping need not occur immediately (for example, it may occur

87530 after the terminal transmits a group of characters to the system), but it shall achieve the same

87531 results as if it occurred immediately.

87532 *Current line*: Unchanged.

87533 *Current column*: Unchanged.

**Mark**

*Synopsis:*     [*laddr*] *ma*[*rk*] *character*  
                   [*laddr*] *k* *character*

Implementations shall support *character* values of a single lowercase letter of the POSIX locale and the backquote and single-quote characters; support of other characters is implementation-defined.

If executing the *vi m* command, set the specified mark to the current line and 1-based numbered character referenced by the current column, if any; otherwise, column position 1.

Otherwise, set the specified mark to the specified line and 1-based numbered first non-<blank> non-<newline> in the line, if any; otherwise, the last non-<newline> in the line, if any; otherwise, column position 1.

The mark shall remain associated with the line until the mark is reset or the line is deleted. If a deleted line is restored by a subsequent **undo** command, any marks previously associated with the line, which have not been reset, shall be restored as well. Any use of a mark not associated with a current line in the edit buffer shall be an error.

The marks ' and ' shall be set as described previously, immediately before the following events occur in the editor:

1. The use of ' \$ ' as an *ex* address
2. The use of a positive decimal number as an *ex* address
3. The use of a search command as an *ex* address
4. The use of a mark reference as an *ex* address
5. The use of the following open and visual mode commands: <control>-], %, (, ), [, ], {, }
6. The use of the following open and visual mode commands: ', **G**, **H**, **L**, **M**, **z** if the current line will change as a result of the command
7. The use of the open and visual mode commands: /, ?, **N**, ', **n** if the current line or column will change as a result of the command
8. The use of the *ex* mode commands: **z**, **undo**, **global**, **v**

For rules 1., 2., 3., and 4., the ' and ' marks shall not be set if the *ex* command is parsed as specified by rule 6.a. in [Command Line Parsing in ex](#) (on page 2670).

For rules 5., 6., and 7., the ' and ' marks shall not be set if the commands are used as motion commands in open and visual mode.

For rules 1., 2., 3., 4., 5., 6., 7., and 8., the ' and ' marks shall not be set if the command fails.

The ' and ' marks shall be set as described previously, each time the contents of the edit buffer are replaced (including the editing of the initial buffer), if in open or visual mode, or if in **ex** mode and the edit buffer is not empty, before any commands or movements (including commands or movements specified by the **-c** or **-t** options or the **+command** argument) are executed on the edit buffer. If in open or visual mode, the marks shall be set as if executing the *vi m* command; otherwise, as if executing the *ex mark* command.

When changing from **ex** mode to open or visual mode, if the ' and ' marks are not already set, the ' and ' marks shall be set as described previously.

*Current line:* Unchanged.

87575 *Current column:* Unchanged.

## 87576 **Move**

87577 *Synopsis:*     [*2addr*] m[ove] *laddr* [*flags*]

87578 Move the specified lines after the specified destination line. A destination of line zero specifies  
87579 that the lines shall be placed at the beginning of the edit buffer. It shall be an error if the  
87580 destination line is within the range of lines to be moved.

87581 *Current line:* Set to the last of the moved lines.

87582 *Current column:* Set to non-<blank>.

## 87583 **Next**

87584 *Synopsis:*     n[ext] [!] [+*command*] [*file* ...]

87585 If no '!' is appended to the command name, and the edit buffer has been modified since the  
87586 last complete write, it shall be an error, unless the file is successfully written as specified by the  
87587 **autowrite** option.

87588 If one or more files is specified:

- 87589     1. Set the argument list to the specified filenames.
- 87590     2. Set the current argument list reference to be the first entry in the argument list.
- 87591     3. Set the current pathname to the first filename specified.

87592 Otherwise:

- 87593     1. It shall be an error if there are no more filenames in the argument list after the filename  
87594       currently referenced.
- 87595     2. Set the current pathname and the current argument list reference to the filename after the  
87596       filename currently referenced in the argument list.

87597 Replace the contents of the edit buffer with the contents of the file named by the current  
87598 pathname. If for any reason the contents of the file cannot be accessed, the edit buffer shall be  
87599 empty.

87600 This command shall be affected by the **autowrite** and **writeany** edit options.

87601 The +*command* option shall be <blank>-delimited; <blank> characters can be escaped by  
87602 preceding them with a <backslash> character. The +*command* shall be interpreted as an *ex*  
87603 command immediately after the contents of the edit buffer have been replaced and the current  
87604 line and column have been set.

87605 *Current line:* Set as described for the **edit** command.

87606 *Current column:* Set as described for the **edit** command.

**Number**

*Synopsis:*     [2addr] nu[mber] [count] [flags]  
                   [2addr] #[count] [flags]

These commands shall be equivalent to the *ex* command:

[2addr] p[rint] [count] #[flags]

See [Print](#).

**Open**

*Synopsis:*     [laddr] o[pen] /pattern/ [flags]

This command need not be supported on block-mode terminals or terminals with insufficient capabilities. If standard input, standard output, or standard error are not terminal devices, the results are unspecified.

Enter open mode.

The trailing delimiter can be omitted from *pattern* at the end of the command line. If *pattern* is empty (for example, *"/"/*) or not specified, the last regular expression used in the editor shall be used as the pattern. The pattern can be delimited by <slash> characters (shown in the Synopsis), as well as any alphanumeric, or non-<blank> other than <backslash>, <vertical-line>, <newline>, or double-quote.

*Current line:* Set to the specified line.

*Current column:* Set to non-<blank>.

**Preserve**

*Synopsis:*     pre[serve]

Save the edit buffer in a form that can later be recovered by using the *-r* option or by using the *ex* **recover** command. After the file has been preserved, a mail message shall be sent to the user. This message shall be readable by invoking the *mailx* utility. The message shall contain the name of the file, the time of preservation, and an *ex* command that could be used to recover the file. Additional information may be included in the mail message.

*Current line:* Unchanged.

*Current column:* Unchanged.

**Print**

*Synopsis:*     [2addr] p[rint] [count] [flags]

Write the addressed lines. The behavior is unspecified if the number of columns on the display is less than the number of columns required to write any single character in the lines being written.

Non-printable characters, except for the <tab>, shall be written as implementation-defined multi-character sequences.

If the # flag is specified or the **number** edit option is set, each line shall be preceded by its line number in the following format:

"%6dΔΔ", <line number>

If the l flag is specified or the **list** edit option is set:

1. The characters listed in XBD [Table 5-1](#) (on page 121) shall be written as the corresponding escape sequence.
2. Non-printable characters not in XBD [Table 5-1](#) (on page 121) shall be written as one three-digit octal number (with a preceding <backslash>) for each byte in the character (most significant byte first).
3. The end of each line shall be marked with a ' \$ ', and literal ' \$ ' characters within the line shall be written with a preceding <backslash>.

Long lines shall be folded; the length at which folding occurs is unspecified, but should be appropriate for the output terminal, considering the number of columns of the terminal.

If a line is folded, and the **l** flag is not specified and the **list** edit option is not set, it is unspecified whether a multi-column character at the folding position is separated; it shall not be discarded.

*Current line*: Set to the last written line.

*Current column*: Unchanged if the current line is unchanged; otherwise, set to non-<blank>.

## Put

*Synopsis*:     [*laddr*] pu[t] [*buffer*]

Append text from the specified buffer (by default, the unnamed buffer) to the specified line; line zero specifies that the text shall be placed at the beginning of the edit buffer. Each portion of a line in the buffer shall become a new line in the edit buffer, regardless of the mode of the buffer.

*Current line*: Set to the last line entered into the edit buffer.

*Current column*: Set to non-<blank>.

## Quit

*Synopsis*:     q[uit] [!]

If no ' ! ' is appended to the command name:

1. If the edit buffer has been modified since the last complete write, it shall be an error.
2. If there are filenames in the argument list after the filename currently referenced, and the last command was not a **quit**, **wq**, **xit**, or **ZZ** (see [Exit](#), on page 3371) command, it shall be an error.

Otherwise, terminate the editing session.

## Read

*Synopsis*:     [*laddr*] r[ead] [!]*[file]*

If ' ! ' is not the first non-<blank> to follow the command name, a copy of the specified file shall be appended into the edit buffer after the specified line; line zero specifies that the copy shall be placed at the beginning of the edit buffer. The number of lines and bytes read shall be written. If no *file* is named, the current pathname shall be the default. If there is no current pathname, then *file* shall become the current pathname. If there is no current pathname or *file* operand, it shall be an error. Specifying a *file* that is not of type regular shall have unspecified results.

Otherwise, if *file* is preceded by ' ! ', the rest of the line after the ' ! ' shall have ' % ', ' # ', and ' ! ' characters expanded as described in [Command Line Parsing in ex](#) (on page 2670).

The *ex* utility shall then pass two arguments to the program named by the shell edit option; the

first shall be **-c** and the second shall be the expanded arguments to the **read** command as a single argument. The standard input of the program shall be set to the standard input of the *ex* program when it was invoked. The standard error and standard output of the program shall be appended into the edit buffer after the specified line.

Each line in the copied file or program output (as delimited by <newline> characters or the end of the file or output if it is not immediately preceded by a <newline>), shall be a separate line in the edit buffer. Any occurrences of <carriage-return> and <newline> pairs in the output shall be treated as single <newline> characters.

The special meaning of the **' ! '** following the **read** command can be overridden by escaping it with a <backslash> character.

*Current line*: If no lines are added to the edit buffer, unchanged. Otherwise, if in open or visual mode, set to the first line entered into the edit buffer. Otherwise, set to the last line entered into the edit buffer.

*Current column*: Set to non-<blank>.

## Recover

*Synopsis:*     **rec[over] [!] file**

If no **' ! '** is appended to the command name, and the edit buffer has been modified since the last complete write, it shall be an error.

If no *file* operand is specified, then the current pathname shall be used. If there is no current pathname or *file* operand, it shall be an error.

If no recovery information has previously been saved about *file*, the **recover** command shall behave identically to the **edit** command, and an informational message to this effect shall be written.

Otherwise, set the current pathname to *file*, and replace the current contents of the edit buffer with the recovered contents of *file*. If there are multiple instances of the file to be recovered, the one most recently saved shall be recovered, and an informational message that there are previous versions of the file that can be recovered shall be written. The editor shall behave as if the contents of the edit buffer have already been modified.

*Current file*: Set as described for the **edit** command.

*Current column*: Set as described for the **edit** command.

## Rewind

*Synopsis:*     **rew[ind] [!]**

If no **' ! '** is appended to the command name, and the edit buffer has been modified since the last complete write, it shall be an error, unless the file is successfully written as specified by the **autowrite** option.

If the argument list is empty, it shall be an error.

The current argument list reference and the current pathname shall be set to the first filename in the argument list.

Replace the contents of the edit buffer with the contents of the file named by the current pathname. If for any reason the contents of the file cannot be accessed, the edit buffer shall be empty.

This command shall be affected by the **autowrite** and **writeany** edit options.

*Current line*: Set as described for the **edit** command.

*Current column*: Set as described for the **edit** command.

## Set

*Synopsis*:     `se[t][option[=[value]] ...][nooption ...][option? ...][all]`

When no arguments are specified, write the value of the **term** edit option and those options whose values have been changed from the default settings; when the argument *all* is specified, write all of the option values.

Giving an option name followed by the character '?' shall cause the current value of that option to be written. The '?' can be separated from the option name by zero or more <blank> characters. The '?' shall be necessary only for Boolean valued options. Boolean options can be given values by the form **set option** to turn them on or **set nooption** to turn them off; string and numeric options can be assigned by the form **set option=value**. Any <blank> characters in strings can be included as is by preceding each <blank> with an escaping <backslash>. More than one option can be set or listed by a single set command by specifying multiple arguments, each separated from the next by one or more <blank> characters.

See [Edit Options in ex](#) (on page 2700) for details about specific options.

*Current line*: Unchanged.

*Current column*: Unchanged.

## Shell

*Synopsis*:     `sh[ell]`

Invoke the program named in the **shell** edit option with the single argument **-i** (interactive mode). Editing shall be resumed when the program exits.

*Current line*: Unchanged.

*Current column*: Unchanged.

## Source

*Synopsis*:     `so[urce] file`

Read and execute *ex* commands from *file*. Lines in the file that are blank lines shall be ignored.

*Current line*: As specified for the individual *ex* commands.

*Current column*: As specified for the individual *ex* commands.

## Substitute

*Synopsis*:     `[2addr] s[ubstitute][/pattern/repl][options][count][flags]]`  
               `[2addr] &[options][count][flags]]`  
               `[2addr] ~[options][count][flags]]`

Replace the first instance of the pattern *pattern* by the string *repl* on each specified line. (See [Regular Expressions in ex](#) (on page 2699) and [Replacement Strings in ex](#) (on page 2700).) Any non-alphabetic, non-<blank> delimiter other than <backslash>, '|', <newline>, or double-quote can be used instead of '/'. <backslash> characters can be used to escape delimiters, <backslash> characters, and other special characters.



87764 The trailing delimiter can be omitted from *pattern* or from *repl* at the end of the command line. If  
 87765 both *pattern* and *repl* are not specified or are empty (for example, `"/ /"`), the last `s` command  
 87766 shall be repeated. If only *pattern* is not specified or is empty, the last regular expression used in  
 87767 the editor shall be used as the pattern. If only *repl* is not specified or is empty, the pattern shall be  
 87768 replaced by nothing. If the entire replacement pattern is `' % '`, the last replacement pattern to an  
 87769 `s` command shall be used.

87770 Entering a <carriage-return> in *repl* (which requires an escaping <backslash> in *ex* mode and an  
 87771 escaping <control>-V in open or *vi* mode) shall split the line at that point, creating a new line in  
 87772 the edit buffer. The <carriage-return> shall be discarded.

87773 If *options* includes the letter `' g '` (**global**), all non-overlapping instances of the pattern in the line  
 87774 shall be replaced.

87775 If *options* includes the letter `' c '` (**confirm**), then before each substitution the line shall be written;  
 87776 the written line shall reflect all previous substitutions. On the following line, <space> characters  
 87777 shall be written beneath the characters from the line that are before the *pattern* to be replaced,  
 87778 and `' ^ '` characters written beneath the characters included in the *pattern* to be replaced. The *ex*  
 87779 utility shall then wait for a response from the user. An affirmative response shall cause the  
 87780 substitution to be done, while any other input shall not make the substitution. An affirmative  
 87781 response shall consist of a line with the affirmative response (as defined by the current locale) at  
 87782 the beginning of the line. This line shall be subject to editing in the same way as the *ex* command  
 87783 line.

87784 If interrupted (see the ASYNCHRONOUS EVENTS section), any modifications confirmed by the  
 87785 user shall be preserved in the edit buffer after the interrupt.

87786 If the remembered search direction is not set, the `s` command shall set it to forward.

87787 In the second Synopsis, the `&` command shall repeat the previous substitution, as if the `&`  
 87788 command were replaced by:

87789 `s/pattern/repl/`

87790 where *pattern* and *repl* are as specified in the previous `s`, `&`, or `~` command.

87791 In the third Synopsis, the `~` command shall repeat the previous substitution, as if the `' ^ '` were  
 87792 replaced by:

87793 `s/pattern/repl/`

87794 where *pattern* shall be the last regular expression specified to the editor, and *repl* shall be from  
 87795 the previous substitution (including `&` and `~`) command.

87796 These commands shall be affected by the `LC_MESSAGES` environment variable.

87797 *Current line*: Set to the last line in which a substitution occurred, or, unchanged if no substitution  
 87798 occurred.

87799 *Current column*: Set to non-<blank>.

## Suspend

*Synopsis:*     su[spend] [!]  
              st[op] [!]

Allow control to return to the invoking process; *ex* shall suspend itself as if it had received the SIGTSTP signal. The suspension shall occur only if job control is enabled in the invoking shell (see the description of *set -m*).

These commands shall be affected by the **autowrite** and **writeany** edit options.

The current **susp** character (see *stty*) shall be equivalent to the **suspend** command.

## Tag

*Synopsis:*     ta[g] [!] *tagstring*

The results are unspecified if the format of a tags file is not as specified by the *ctags* utility (see *ctags*) description.

The **tag** command shall search for *tagstring* in the tag files referred to by the **tag** edit option, in the order they are specified, until a reference to *tagstring* is found. Files shall be searched from beginning to end. If no reference is found, it shall be an error and an error message to this effect shall be written. If the reference is not found, or if an error occurs while processing a file referred to in the **tag** edit option, it shall be an error, and an error message shall be written at the first occurrence of such an error.

Otherwise, if the tags file contained a pattern, the pattern shall be treated as a regular expression used in the editor; for example, for the purposes of the **s** command.

If the *tagstring* is in a file with a different name than the current pathname, set the current pathname to the name of that file, and replace the contents of the edit buffer with the contents of that file. In this case, if no '!' is appended to the command name, and the edit buffer has been modified since the last complete write, it shall be an error, unless the file is successfully written as specified by the **autowrite** option.

This command shall be affected by the **autowrite**, **tag**, **taglength**, and **writeany** edit options.

*Current line:* If the tags file contained a line number, set to that line number. If the line number is larger than the last line in the edit buffer, an error message shall be written and the current line shall be set as specified for the **edit** command.

If the tags file contained a pattern, set to the first occurrence of the pattern in the file. If no matching pattern is found, an error message shall be written and the current line shall be set as specified for the **edit** command.

*Current column:* If the tags file contained a line-number reference and that line-number was not larger than the last line in the edit buffer, or if the tags file contained a pattern and that pattern was found, set to non-<blank>. Otherwise, set as specified for the **edit** command.

**Unabbreviate**

*Synopsis:*     una[bbrev] lhs

If *lhs* is not an entry in the current list of abbreviations (see [Abbreviate](#), on page 2678), it shall be an error. Otherwise, delete *lhs* from the list of abbreviations.

*Current line:* Unchanged.

*Current column:* Unchanged.

**Undo**

*Synopsis:*     u[ndo]

Reverse the changes made by the last command that modified the contents of the edit buffer, including **undo**. For this purpose, the **global**, **v**, **open**, and **visual** commands, and commands resulting from buffer executions and mapped character expansions, are considered single commands.

If no action that can be undone preceded the **undo** command, it shall be an error.

If the **undo** command restores lines that were marked, the mark shall also be restored unless it was reset subsequent to the deletion of the lines.

*Current line:*

1. If lines are added or changed in the file, set to the first line added or changed.
2. Set to the line before the first line deleted, if it exists.
3. Set to 1 if the edit buffer is not empty.
4. Set to zero.

*Current column:* Set to non-<blank>.

**Unmap**

*Synopsis:*     unm[ap][!] lhs

If '!' is appended to the command name, and if *lhs* is not an entry in the list of text input mode map definitions, it shall be an error. Otherwise, delete *lhs* from the list of text input mode map definitions.

If no '!' is appended to the command name, and if *lhs* is not an entry in the list of command mode map definitions, it shall be an error. Otherwise, delete *lhs* from the list of command mode map definitions.

*Current line:* Unchanged.

*Current column:* Unchanged.

**Version**

*Synopsis:*     `ve[rsion]`

Write a message containing version information for the editor. The format of the message is unspecified.

*Current line:* Unchanged.

*Current column:* Unchanged.

**Visual**

*Synopsis:*     `[laddr] vi[sual] [type] [count] [flags]`

If *ex* is currently in open or visual mode, the Synopsis and behavior of the visual command shall be the same as the **edit** command, as specified by **Edit** (on page 2680).

Otherwise, this command need not be supported on block-mode terminals or terminals with insufficient capabilities. If standard input, standard output, or standard error are not terminal devices, the results are unspecified.

If *count* is specified, the value of the **window** edit option shall be set to *count* (as described in **window**, on page 2707). If the `'^'` type character was also specified, the **window** edit option shall be set before being used by the type character.

Enter visual mode. If *type* is not specified, it shall be as if a *type* of `'+'` was specified. The *type* shall cause the following effects:

- + Place the beginning of the specified line at the top of the display.
- Place the end of the specified line at the bottom of the display.
- . Place the beginning of the specified line in the middle of the display.
- ^ If the specified line is less than or equal to the value of the **window** edit option, set the line to 1; otherwise, decrement the line by the value of the **window** edit option minus 1. Place the beginning of this line as close to the bottom of the displayed lines as possible, while still displaying the value of the **window** edit option number of lines.

*Current line:* Set to the specified line.

*Current column:* Set to non-<blank>.

**Write**

*Synopsis:*     `[2addr] w[rite] [!] [>>] [file]`  
                   `[2addr] w[rite] [!] [file]`  
                   `[2addr] wq[!] [>>] [file]`

If no lines are specified, the lines shall default to the entire file.

The command **wq** shall be equivalent to a **write** command followed by a **quit** command; **wq!** shall be equivalent to **write!** followed by **quit**. In both cases, if the **write** command fails, the **quit** shall not be attempted.

If the command name is not followed by one or more <blank> characters, or *file* is not preceded by a `'!'` character, the **write** shall be to a file.

1. If the `>>` argument is specified, and the file already exists, the lines shall be appended to the file instead of replacing its contents. If the `>>` argument is specified, and the file does not already exist, it is unspecified whether the write shall proceed as if the `>>` argument

- 87906 had not been specified or if the **write** shall fail.
- 87907 2. If the **readonly** edit option is set (see [readonly](#), on page 2704), the **write** shall fail.
- 87908 3. If *file* is specified, and is not the current pathname, and the file exists, the **write** shall fail.
- 87909 4. If *file* is not specified, the current pathname shall be used. If there is no current pathname,
- 87910 the **write** command shall fail.
- 87911 5. If the current pathname is used, and the current pathname has been changed by the **file**
- 87912 or **read** commands, and the file exists, the **write** shall fail. If the **write** is successful,
- 87913 subsequent **writes** shall not fail for this reason (unless the current pathname is changed
- 87914 again).
- 87915 6. If the whole edit buffer is not being written, and the file to be written exists, the **write**
- 87916 shall fail.

87917 For rules 1., 2., 3., and 5., the **write** can be forced by appending the character '!' to the

87918 command name.

87919 For rules 2., 3., and 5., the **write** can be forced by setting the **writeany** edit option.

87920 Additional, implementation-defined tests may cause the **write** to fail.

87921 If the edit buffer is empty, a file without any contents shall be written.

87922 An informational message shall be written noting the number of lines and bytes written.

87923 Otherwise, if the command is followed by one or more <blank> characters, and the file is

87924 preceded by '!', the rest of the line after the '!' shall have '%', '#', and '!' characters

87925 expanded as described in [Command Line Parsing in ex](#) (on page 2670).

87926 The *ex* utility shall then pass two arguments to the program named by the **shell** edit option; the

87927 first shall be **-c** and the second shall be the expanded arguments to the **write** command as a

87928 single argument. The specified lines shall be written to the standard input of the command. The

87929 standard error and standard output of the program, if any, shall be written as described for the

87930 **print** command. If the last character in that output is not a <newline>, a <newline> shall be

87931 written at the end of the output.

87932 The special meaning of the '!' following the **write** command can be overridden by escaping it

87933 with a <backslash> character.

87934 *Current line*: Unchanged.

87935 *Current column*: Unchanged.

## 87936 Write and Exit

87937 *Synopsis:*     [*2addr*] x[*it*][!][*file*]

87938 If the edit buffer has not been modified since the last complete **write**, **xit** shall be equivalent to

87939 the **quit** command, or if a '!' is appended to the command name, to **quit!**.

87940 Otherwise, **xit** shall be equivalent to the **wq** command, or if a '!' is appended to the command

87941 name, to **wq!**.

87942 *Current line*: Unchanged.

87943 *Current column*: Unchanged.

**Yank**

*Synopsis:*     `[2addr] ya[nk] [buffer] [count]`

Copy the specified lines to the specified buffer (by default, the unnamed buffer), which shall become a line-mode buffer.

*Current line:* Unchanged.

*Current column:* Unchanged.

**Adjust Window**

*Synopsis:*     `[laddr] z[!][type ...] [count] [flags]`

If no line is specified, the current line shall be the default; if *type* is omitted as well, the current line value shall first be incremented by 1. If incrementing the current line would cause it to be greater than the last line in the edit buffer, it shall be an error.

If there are <blank> characters between the *type* argument and the preceding *z* command name or optional '!' character, it shall be an error.

If *count* is specified, the value of the **window** edit option shall be set to *count* (as described in **window**, on page 2707). If *count* is omitted, it shall default to 2 times the value of the **scroll** edit option, or if ! was specified, the number of lines in the display minus 1.

If *type* is omitted, then *count* lines starting with the specified line shall be written. Otherwise, *count* lines starting with the line specified by the *type* argument shall be written.

The *type* argument shall change the lines to be written. The possible values of *type* are as follows:

- The specified line shall be decremented by the following value:

$((\text{number of ``-'' characters}) \times \text{count}) - 1$

If the calculation would result in a number less than 1, it shall be an error. Write lines from the edit buffer, starting at the new value of line, until *count* lines or the last line in the edit buffer has been written.

- + The specified line shall be incremented by the following value:

$((\text{number of ``+'' characters}) - 1) \times \text{count} + 1$

If the calculation would result in a number greater than the last line in the edit buffer, it shall be an error. Write lines from the edit buffer, starting at the new value of line, until *count* lines or the last line in the edit buffer has been written.

**=,.** If more than a single ' .' or '=' is specified, it shall be an error. The following steps shall be taken:

1. If *count* is zero, nothing shall be written.
2. Write as many of the *N* lines before the current line in the edit buffer as exist. If *count* or '!' was specified, *N* shall be:

$(\text{count} - 1) / 2$

Otherwise, *N* shall be:

$(\text{count} - 3) / 2$

If *N* is a number less than 3, no lines shall be written.

- 87982 3. If '=' was specified as the type character, write a line consisting of the smaller of the  
87983 number of columns in the display divided by two, or 40 '-' characters.
- 87984 4. Write the current line.
- 87985 5. Repeat step 3.
- 87986 6. Write as many of the *N* lines after the current line in the edit buffer as exist. *N* shall  
87987 be defined as in step 2. If *N* is a number less than 3, no lines shall be written. If *count*  
87988 is less than 3, no lines shall be written.

87989 ^ The specified line shall be decremented by the following value:

87990  $((\text{number of ``^`` characters}) + 1) \times \text{count} - 1$

87991 If the calculation would result in a number less than 1, it shall be an error. Write lines from  
87992 the edit buffer, starting at the new value of line, until *count* lines or the last line in the edit  
87993 buffer has been written.

87994 *Current line*: Set to the last line written, unless the type is =, in which case, set to the specified  
87995 line.

87996 *Current column*: Set to non-<blank>.

## 87997 Escape

87998 *Synopsis:*     ! *command*  
87999               [*addr*] ! *command*

88000 The contents of the line after the '!' shall have '%', '#', and '!' characters expanded as  
88001 described in [Command Line Parsing in ex](#) (on page 2670). If the expansion causes the text of the  
88002 line to change, it shall be redisplayed, preceded by a single '!' character.

88003 The *ex* utility shall execute the program named by the **shell** edit option. It shall pass two  
88004 arguments to the program; the first shall be -c, and the second shall be the expanded arguments  
88005 to the ! command as a single argument.

88006 If no lines are specified, the standard input, standard output, and standard error of the program  
88007 shall be set to the standard input, standard output, and standard error of the *ex* program when it  
88008 was invoked. In addition, a warning message shall be written if the edit buffer has been  
88009 modified since the last complete write, and the **warn** edit option is set.

88010 If lines are specified, they shall be passed to the program as standard input, and the standard  
88011 output and standard error of the program shall replace those lines in the edit buffer. Each line in  
88012 the program output (as delimited by <newline> characters or the end of the output if it is not  
88013 immediately preceded by a <newline>), shall be a separate line in the edit buffer. Any  
88014 occurrences of <carriage-return> and <newline> pairs in the output shall be treated as single  
88015 <newline> characters. The specified lines shall be copied into the unnamed buffer before they  
88016 are replaced, and the unnamed buffer shall become a line-mode buffer.

88017 If in *ex* mode, a single '!' character shall be written when the program completes.

88018 This command shall be affected by the **shell** and **warn** edit options. If no lines are specified, this  
88019 command shall be affected by the **autowrite** and **writeany** edit options. If lines are specified, this  
88020 command shall be affected by the **autoprint** edit option.

88021 *Current line:*

- 88022 1. If no lines are specified, unchanged.
- 88023 2. Otherwise, set to the last line read in, if any lines are read in.
- 88024 3. Otherwise, set to the line before the first line of the lines specified, if that line exists.
- 88025 4. Otherwise, set to the first line of the edit buffer if the edit buffer is not empty.
- 88026 5. Otherwise, set to zero.

88027 *Current column:* If no lines are specified, unchanged. Otherwise, set to non-<blank>.

## 88028 **Shift Left**

88029 *Synopsis:* `[2addr] <[< ...] [count] [flags]`

88030 Shift the specified lines to the start of the line; the number of column positions to be shifted shall  
 88031 be the number of command characters times the value of the **shiftwidth** edit option. Only  
 88032 leading <blank> characters shall be deleted or changed into other <blank> characters in shifting;  
 88033 other characters shall not be affected.

88034 Lines to be shifted shall be copied into the unnamed buffer, which shall become a line-mode  
 88035 buffer.

88036 This command shall be affected by the **autoprint** edit option.

88037 *Current line:* Set to the last line in the lines specified.

88038 *Current column:* Set to non-<blank>.

## 88039 **Shift Right**

88040 *Synopsis:* `[2addr] >[> ...] [count] [flags]`

88041 Shift the specified lines away from the start of the line; the number of column positions to be  
 88042 shifted shall be the number of command characters times the value of the **shiftwidth** edit  
 88043 option. The shift shall be accomplished by adding <blank> characters as a prefix to the line or  
 88044 changing leading <blank> characters into other <blank> characters. Empty lines shall not be  
 88045 changed.

88046 Lines to be shifted shall be copied into the unnamed buffer, which shall become a line-mode  
 88047 buffer.

88048 This command shall be affected by the **autoprint** edit option.

88049 *Current line:* Set to the last line in the lines specified.

88050 *Current column:* Set to non-<blank>.

## 88051 **<control>-D**

88052 *Synopsis:* `<control>-D`

88053 Write the next *n* lines, where *n* is the minimum of the values of the **scroll** edit option and the  
 88054 number of lines after the current line in the edit buffer. If the current line is the last line of the  
 88055 edit buffer it shall be an error.

88056 *Current line:* Set to the last line written.

88057 *Current column:* Set to non-<blank>.



**Write Line Number**

*Synopsis:*     [*laddr*] = [*flags*]

If *line* is not specified, it shall default to the last line in the edit buffer. Write the line number of the specified line.

*Current line:* Unchanged.

*Current column:* Unchanged.

**Execute**

*Synopsis:*     [*2addr*] @ *buffer*  
                 [*2addr*] \* *buffer*

If no *buffer* is specified or is specified as '@' or '\*', the last *buffer* executed shall be used. If no previous *buffer* has been executed, it shall be an error.

For each line specified by the addresses, set the current line ('.') to the specified line, and execute the contents of the named *buffer* (as they were at the time the @ command was executed) as *ex* commands. For each line of a line-mode *buffer*, and all but the last line of a character-mode *buffer*, the *ex* command parser shall behave as if the line was terminated by a <newline>.

If an error occurs during this process, or a line specified by the addresses does not exist when the current line would be set to it, or more than a single line was specified by the addresses, and the contents of the edit buffer are replaced (for example, by the *ex* :edit command) an error message shall be written, and no more commands resulting from the execution of this command shall be processed.

*Current line:* As specified for the individual *ex* commands.

*Current column:* As specified for the individual *ex* commands.

**Regular Expressions in ex**

The *ex* utility shall support regular expressions that are a superset of the basic regular expressions described in XBD [Section 9.3](#) (on page 183). A null regular expression ("/ /") shall be equivalent to the last regular expression encountered.

Regular expressions can be used in addresses to specify lines and, in some commands (for example, the **substitute** command), to specify portions of a line to be substituted.

The following constructs can be used to enhance the basic regular expressions:

\< Match the beginning of a *word*. (See the definition of *word* at the beginning of [Command Descriptions in ex](#) (on page 2675).)

\> Match the end of a *word*.

~ Match the replacement part of the last **substitute** command. The <tilde> ('~') character can be escaped in a regular expression to become a normal character with no special meaning. The <backslash> shall be discarded.

When the editor option **magic** is not set, the only characters with special meanings shall be '^' at the beginning of a pattern, '\$' at the end of a pattern, and <backslash>. The characters '.', '\*', '[', and '~' shall be treated as ordinary characters unless preceded by a <backslash>; when preceded by a <backslash> they shall regain their special meaning, or in the case of <backslash>, be handled as a single <backslash>. <backslash> characters used to escape other characters shall be discarded.

## Replacement Strings in ex

The character `'&'` (`'\&'` if the editor option **magic** is not set) in the replacement string shall stand for the text matched by the pattern to be replaced. The character `'~'` (`'\~'` if **magic** is not set) shall be replaced by the replacement part of the previous **substitute** command. The sequence `'\n'`, where *n* is an integer, shall be replaced by the text matched by the corresponding back-reference expression. If the corresponding back-reference expression does not match, then the characters `'\n'` shall be replaced by the empty string.

The strings `'\l'`, `'\u'`, `'\L'`, and `'\U'` can be used to modify the case of elements in the replacement string (using the `'\&'` or `"\"digit`) notation. The string `'\l'` (`'\u'`) shall cause the character that follows to be converted to lowercase (uppercase). The string `'\L'` (`'\U'`) shall cause all characters subsequent to it to be converted to lowercase (uppercase) as they are inserted by the substitution until the string `'\e'` or `'\E'`, or the end of the replacement string, is encountered.

Otherwise, any character following a `<backslash>` shall be treated as that literal character, and the escaping `<backslash>` shall be discarded.

An example of case conversion with the **s** command is as follows:

```
:p
The cat sat on the mat.
:s/\<.at\>/\u&/gp
The Cat Sat on the Mat.
:s/S\(.*\)M/S\U\1\eM/p
The Cat SAT ON THE Mat.
```

## Edit Options in ex

The *ex* utility has a number of options that modify its behavior. These options have default settings, which can be changed using the **set** command.

Options are Boolean unless otherwise specified.

### autoindent, ai

[Default *unset*]

If **autoindent** is set, each line in input mode shall be indented (using first as many `<tab>` characters as possible, as determined by the editor option **tabstop**, and then using `<space>` characters) to align with another line, as follows:

1. If in open or visual mode and the text input is part of a line-oriented command (see the EXTENDED DESCRIPTION in *vi*), align to the first column.
2. Otherwise, if in open or visual mode, indentation for each line shall be set as follows:
  - a. If a line was previously inserted as part of this command, it shall be set to the indentation of the last inserted line by default, or as otherwise specified for the `<control>-D` character in *Input Mode Commands* in *vi* (on page 3371).
  - b. Otherwise, it shall be set to the indentation of the previous current line, if any; otherwise, to the first column.
3. For the *ex* **a**, **i**, and **c** commands, indentation for each line shall be set as follows:

- 88139 a. If a line was previously inserted as part of this command, it shall be set to the
- 88140 indentation of the last inserted line by default, or as otherwise specified for the *eof*
- 88141 character in [Scroll](#) (on page 2674).
- 88142 b. Otherwise, if the command is the *ex a* command, it shall be set to the line
- 88143 appended after, if any; otherwise to the first column.
- 88144 c. Otherwise, if the command is the *ex i* command, it shall be set to the line inserted
- 88145 before, if any; otherwise to the first column.
- 88146 d. Otherwise, if the command is the *ex c* command, it shall be set to the indentation of
- 88147 the line replaced.

#### 88148 **autoprint, ap**

88149 [Default *set*]

88150 If **autoprint** is set, the current line shall be written after each *ex* command that modifies the

88151 contents of the current edit buffer, and after each **tag** command for which the tag search pattern

88152 was found or tag line number was valid, unless:

- 88153 1. The command was executed while in open or visual mode.
- 88154 2. The command was executed as part of a **global** or **v** command or **@** buffer execution.
- 88155 3. The command was the form of the **read** command that reads a file into the edit buffer.
- 88156 4. The command was the **append**, **change**, or **insert** command.
- 88157 5. The command was not terminated by a <newline>.
- 88158 6. The current line shall be written by a flag specified to the command; for example, **delete #**
- 88159 shall write the current line as specified for the flag modifier to the **delete** command, and
- 88160 not as specified by the **autoprint** edit option.

#### 88161 **autowrite, aw**

88162 [Default *unset*]

88163 If **autowrite** is set, and the edit buffer has been modified since it was last completely written to

88164 any file, the contents of the edit buffer shall be written as if the *ex write* command had been

88165 specified without arguments, before each command affected by the **autowrite** edit option is

88166 executed. Appending the character '!' to the command name of any of the *ex* commands

88167 except '!' shall prevent the write. If the write fails, it shall be an error and the command shall

88168 not be executed.

#### 88169 **beautify, bf**

88170 XSI [Default *unset*]

88171 If **beautify** is set, all non-printable characters, other than <tab>, <newline>, and <form-feed>

88172 characters, shall be discarded from text read in from files.

88173        **directory, dir**  
88174        [Default *implementation-defined*]  
88175        The value of this option specifies the directory in which the editor buffer is to be placed. If this  
88176        directory is not writable by the user, the editor shall quit.

88177        **edcompatible, ed**  
88178        [Default *unset*]  
88179        Causes the presence of **g** and **c** suffixes on substitute commands to be remembered, and toggled  
88180        by repeating the suffixes.

88181        **errorbells, eb**  
88182        [Default *unset*]  
88183        If the editor is in *ex* mode, and the terminal does not support a standout mode (such as inverse  
88184        video), and **errorbells** is set, error messages shall be preceded by alerting the terminal.

88185        **exrc**  
88186        [Default *unset*]  
88187        If **exrc** is set, *ex* shall access any **.exrc** file in the current directory, as described in [Initialization in](#)  
88188        [ex and vi](#) (on page 2666). If **exrc** is not set, *ex* shall ignore any **.exrc** file in the current directory  
88189        during initialization, unless the current directory is that named by the *HOME* environment  
88190        variable.

88191        **ignorecase, ic**  
88192        [Default *unset*]  
88193        If **ignorecase** is set, characters that have uppercase and lowercase representations shall have  
88194        those representations considered as equivalent for purposes of regular expression comparison.  
88195        The **ignorecase** edit option shall affect all remembered regular expressions; for example,  
88196        unsetting the **ignorecase** edit option shall cause a subsequent *vi n* command to search for the  
88197        last basic regular expression in a case-sensitive fashion.

88198        **list**  
88199        [Default *unset*]  
88200        If **list** is set, edit buffer lines written while in *ex* command mode shall be written as specified for  
88201        the **print** command with the **l** flag specified. In open or visual mode, each edit buffer line shall  
88202        be displayed as specified for the *ex print* command with the **l** flag specified. In open or visual  
88203        text input mode, when the cursor does not rest on any character in the line, it shall rest on the  
88204        ' **\$** ' marking the end of the line.

88205 **magic**88206 [Default *set*]

88207 If **magic** is set, modify the interpretation of characters in regular expressions and substitution  
 88208 replacement strings (see [Regular Expressions in ex](#) (on page 2699) and [Replacement Strings in](#)  
 88209 [ex](#), on page 2700).

88210 **mesg**88211 [Default *set*]

88212 If **mesg** is set, the permission for others to use the **write** or **talk** commands to write to the  
 88213 terminal shall be turned on while in open or visual mode. The shell-level command *mesg n* shall  
 88214 take precedence over any setting of the *ex mesg* option; that is, if **mesg y** was issued before the  
 88215 editor started (or in a shell escape), such as:

88216 `:!mesg y`

88217 the **mesg** option in *ex* shall suppress incoming messages, but the **mesg** option shall not enable  
 88218 incoming messages if **mesg n** was issued.

88219 **number, nu**88220 [Default *unset*]

88221 If **number** is set, edit buffer lines written while in *ex* command mode shall be written with line  
 88222 numbers, in the format specified by the **print** command with the **#** flag specified. In *ex* text input  
 88223 mode, each line shall be preceded by the line number it will have in the file.

88224 In open or visual mode, each edit buffer line shall be displayed with a preceding line number, in  
 88225 the format specified by the *ex print* command with the **#** flag specified. This line number shall  
 88226 not be considered part of the line for the purposes of evaluating the current column; that is,  
 88227 column position 1 shall be the first column position after the format specified by the **print**  
 88228 command.

88229 **paragraphs, para**88230 [Default in the POSIX locale `IPLPPPQPP LIpplpipbp`]

88231 The **paragraphs** edit option shall define additional paragraph boundaries for the open and  
 88232 visual mode commands. The **paragraphs** edit option can be set to a character string consisting of  
 88233 zero or more character pairs. It shall be an error to set it to an odd number of characters.

88234 **prompt**88235 [Default *set*]

88236 If **prompt** is set, *ex* command mode input shall be prompted for with a <colon> (':'); when  
 88237 unset, no prompt shall be written.

**readonly**[Default *see text*]

If the **readonly** edit option is set, read-only mode shall be enabled (see **Write**, on page 2694). The **readonly** edit option shall be initialized to set if either of the following conditions are true:

- The command-line option **-R** was specified.
- Performing actions equivalent to the *access()* function called with the following arguments indicates that the file lacks write permission:
  1. The current pathname is used as the *path* argument.
  2. The constant **W\_OK** is used as the *amode* argument.

The **readonly** edit option may be initialized to set for other, implementation-defined reasons. The **readonly** edit option shall not be initialized to unset based on any special privileges of the user or process. The **readonly** edit option shall be reinitialized each time that the contents of the edit buffer are replaced (for example, by an **edit** or **next** command) unless the user has explicitly set it, in which case it shall remain set until the user explicitly unsets it. Once unset, it shall again be reinitialized each time that the contents of the edit buffer are replaced.

**redraw**[Default *unset*]

The editor simulates an intelligent terminal on a dumb terminal. (Since this is likely to require a large amount of output to the terminal, it is useful only at high transmission speeds.)

**remap**[Default *set*]

If **remap** is set, map translation shall allow for maps defined in terms of other maps; translation shall continue until a final product is obtained. If unset, only a one-step translation shall be done.

**report**

[Default 5]

The value of this **report** edit option specifies what number of lines being added, copied, deleted, or modified in the edit buffer will cause an informational message to be written to the user. The following conditions shall cause an informational message. The message shall contain the number of lines added, copied, deleted, or modified, but is otherwise unspecified.

- An *ex* or *vi* editor command, other than **open**, **undo**, or **visual**, that modifies at least the value of the **report** edit option number of lines, and which is not part of an *ex* **global** or **v** command, or *ex* or *vi* buffer execution, shall cause an informational message to be written.
- An *ex* **yank** or *vi* **y** or **Y** command, that copies at least the value of the **report** edit option plus 1 number of lines, and which is not part of an *ex* **global** or **v** command, or *ex* or *vi* buffer execution, shall cause an informational message to be written.
- An *ex* **global**, **v**, **open**, **undo**, or **visual** command or *ex* or *vi* buffer execution, that adds or deletes a total of at least the value of the **report** edit option number of lines, and which is not part of an *ex* **global** or **v** command, or *ex* or *vi* buffer execution, shall cause an informational message to be written. (For example, if 3 lines were added and 8 lines deleted during an *ex* **visual** command, 5 would be the number compared against the

88279 **report** edit option after the command completed.)

88280 **scroll, scr**

88281 [Default (number of lines in the display -1)/2]

88282 The value of the **scroll** edit option shall determine the number of lines scrolled by the *ex*  
88283 <control>-D and **z** commands. For the *vi* <control>-D and <control>-U commands, it shall be the  
88284 initial number of lines to scroll when no previous <control>-D or <control>-U command has  
88285 been executed.

88286 **sections**

88287 [Default in the POSIX locale `NHSHH HUnhsh`]

88288 The **sections** edit option shall define additional section boundaries for the open and visual mode  
88289 commands. The **sections** edit option can be set to a character string consisting of zero or more  
88290 character pairs; it shall be an error to set it to an odd number of characters.

88291 **shell, sh**

88292 [Default from the environment variable *SHELL*]

88293 The value of this option shall be a string. The default shall be taken from the *SHELL*  
88294 environment variable. If the *SHELL* environment variable is null or empty, the *sh* (see *sh*) utility  
88295 shall be the default.

88296 **shiftwidth, sw**

88297 [Default 8]

88298 The value of this option shall give the width in columns of an indentation level used during  
88299 autoindentation and by the shift commands (< and >).

88300 **showmatch, sm**

88301 [Default *unset*]

88302 The functionality described for the **showmatch** edit option need not be supported on block-  
88303 mode terminals or terminals with insufficient capabilities.

88304 If **showmatch** is set, in open or visual mode, when a ' ) ' or ' } ' is typed, if the matching ' ( ' or  
88305 ' { ' is currently visible on the display, the matching ' ( ' or ' { ' shall be flagged moving the  
88306 cursor to its location for an unspecified amount of time.

88307 **showmode**

88308 [Default *unset*]

88309 If **showmode** is set, in open or visual mode, the current mode that the editor is in shall be  
88310 displayed on the last line of the display. Command mode and text input mode shall be  
88311 differentiated; other unspecified modes and implementation-defined information may be  
88312 displayed.

**slowopen**

[Default *unset*]

If **slowopen** is set during open and visual text input modes, the editor shall not update portions of the display other than those display line columns that display the characters entered by the user (see [Input Mode Commands in vi](#), on page 3371).

**tabstop, ts**

[Default 8]

The value of this edit option shall specify the column boundary used by a <tab> in the display (see [autoprint, ap](#) (on page 2701) and [Input Mode Commands in vi](#), on page 3371).

**taglength, tl**

[Default zero]

The value of this edit option shall specify the maximum number of characters that are considered significant in the user-specified tag name and in the tag name from the tags file. If the value is zero, all characters in both tag names shall be significant.

**tags**

[Default *see text*]

The value of this edit option shall be a string of <blank>-delimited pathnames of files used by the **tag** command. The default value is unspecified.

**term**

[Default from the environment variable *TERM*]

The value of this edit option shall be a string. The default shall be taken from the *TERM* variable in the environment. If the *TERM* environment variable is empty or null, the default is unspecified. The editor shall use the value of this edit option to determine the type of the display device.

The results are unspecified if the user changes the value of the term edit option after editor initialization.

**terse**

[Default *unset*]

If **terse** is set, error messages may be less verbose. However, except for this caveat, error messages are unspecified. Furthermore, not all error messages need change for different settings of this option.



**warn**[Default *set*]

If **warn** is set, and the contents of the edit buffer have been modified since they were last completely written, the editor shall write a warning message before certain ! commands (see [Escape](#), on page 2697).

**window**[Default *see text*]

A value used in open and visual mode, by the <control>-B and <control>-F commands, and, in visual mode, to specify the number of lines displayed when the screen is repainted.

If the **-w** command-line option is not specified, the default value shall be set to the value of the *LINES* environment variable. If the *LINES* environment variable is empty or null, the default shall be the number of lines in the display minus 1.

Setting the **window** edit option to zero or to a value greater than the number of lines in the display minus 1 (either explicitly or based on the **-w** option or the *LINES* environment variable) shall cause the **window** edit option to be set to the number of lines in the display minus 1.

The baud rate of the terminal line may change the default in an implementation-defined manner.

**wrapmargin, wm**

[Default 0]

If the value of this edit option is zero, it shall have no effect.

If not in the POSIX locale, the effect of this edit option is implementation-defined.

Otherwise, it shall specify a number of columns from the ending margin of the terminal.

During open and visual text input modes, for each character for which any part of the character is displayed in a column that is less than **wrapmargin** columns from the ending margin of the display line, the editor shall behave as follows:

1. If the character triggering this event is a <blank>, it, and all immediately preceding <blank> characters on the current line entered during the execution of the current text input command, shall be discarded, and the editor shall behave as if the user had entered a single <newline> instead. In addition, if the next user-entered character is a <space>, it shall be discarded as well.
2. Otherwise, if there are one or more <blank> characters on the current line immediately preceding the last group of inserted non-<blank> characters which was entered during the execution of the current text input command, the <blank> characters shall be replaced as if the user had entered a single <newline> instead.

If the **autoindent** edit option is set, and the events described in 1. or 2. are performed, any <blank> characters at or after the cursor in the current line shall be discarded.

The ending margin shall be determined by the system or overridden by the user, as described for *COLUMNS* in the ENVIRONMENT VARIABLES section and XBD [Chapter 8](#) (on page 173).

88381 **wrapsan, ws**

88382 [Default *set*]

88383 If **wrapsan** is set, searches (the *ex* / or ? addresses, or open and visual mode /, ?, N, and n  
88384 commands) shall wrap around the beginning or end of the edit buffer; when unset, searches  
88385 shall stop at the beginning or end of the edit buffer.

88386 **writeany, wa**

88387 [Default *unset*]

88388 If **writeany** is set, some of the checks performed when executing the *ex* **write** commands shall be  
88389 inhibited, as described in editor option **autowrite**.

## 88390 EXIT STATUS

88391 The following exit values shall be returned:

88392 0 Successful completion.

88393 >0 An error occurred.

## 88394 CONSEQUENCES OF ERRORS

88395 When any error is encountered and the standard input is not a terminal device file, *ex* shall not  
88396 write the file or return to command or text input mode, and shall terminate with a non-zero exit  
88397 status.

88398 Otherwise, when an unrecoverable error is encountered, it shall be equivalent to a SIGHUP  
88399 asynchronous event.

88400 Otherwise, when an error is encountered, the editor shall behave as specified in [Command Line](#)  
88401 [Parsing in ex](#) (on page 2670).

## 88402 APPLICATION USAGE

88403 If a SIGSEGV signal is received while *ex* is saving a file, the file might not be successfully saved.

88404 The **next** command can accept more than one file, so usage such as:

88405 `next `ls [abc]*``

88406 is valid; it would not be valid for the **edit** or **read** commands, for example, because they expect  
88407 only one file and unspecified results occur.

## 88408 EXAMPLES

88409 None.

## 88410 RATIONALE

88411 The *ex/vi* specification is based on the historical practice found in the 4 BSD and System V  
88412 implementations of *ex* and *vi*.

88413 A *restricted editor* (both the historical *red* utility and modifications to *ex*) were considered and  
88414 rejected for inclusion. Neither option provided the level of security that users might expect.

88415 It is recognized that *ex* visual mode and related features would be difficult, if not impossible, to  
88416 implement satisfactorily on a block-mode terminal, or a terminal without any form of cursor  
88417 addressing; thus, it is not a mandatory requirement that such features should work on all  
88418 terminals. It is the intention, however, that an *ex* implementation should provide the full set of  
88419 capabilities on all terminals capable of supporting them.

## Options

The `-c` replacement for `+command` was inspired by the `-e` option of `sed`. Historically, all such commands (see `edit` and `next` as well) were executed from the last line of the edit buffer. This meant, for example, that `"/pattern"` would fail unless the `wrapsan` option was set. POSIX.1-2008 requires conformance to historical practice. The `+command` option is no longer specified by POSIX.1-2008 but may be present in some implementations. Historically, some implementations restricted the `ex` commands that could be listed as part of the command line arguments. For consistency, POSIX.1-2008 does not permit these restrictions.

In historical implementations of the editor, the `-R` option (and the `readonly` edit option) only prevented overwriting of files; appending to files was still permitted, mapping loosely into the `cs` `noclobber` variable. Some implementations, however, have not followed this semantic, and `readonly` does not permit appending either. POSIX.1-2008 follows the latter practice, believing that it is a more obvious and intuitive meaning of `readonly`.

The `-s` option suppresses all interactive user feedback and is useful for editing scripts in batch jobs. The list of specific effects is historical practice. The terminal type "incapable of supporting open and visual modes" has historically been named "dumb".

The `-t` option was required because the `ctags` utility appears in POSIX.1-2008 and the option is available in all historical implementations of `ex`.

Historically, the `ex` and `vi` utilities accepted a `-x` option, which did encryption based on the algorithm found in the historical `crypt` utility. The `-x` option for encryption, and the associated `crypt` utility, were omitted because the algorithm used was not specifiable and the export control laws of some nations make it difficult to export cryptographic technology. In addition, it did not historically provide the level of security that users might expect.

## Standard Input

An end-of-file condition is not equivalent to an end-of-file character. A common end-of-file character, `<control>-D`, is historically an `ex` command.

There was no maximum line length in historical implementations of `ex`. Specifically, as it was parsed in chunks, the addresses had a different maximum length than the filenames. Further, the maximum line buffer size was declared as `BUFSIZ`, which was different lengths on different systems. This version selected the value of `{LINE_MAX}` to impose a reasonable restriction on portable usage of `ex` and to aid test suite writers in their development of realistic tests that exercise this limit.

## Input Files

It was an explicit decision by the standard developers that a `<newline>` be added to any file lacking one. It was believed that this feature of `ex` and `vi` was relied on by users in order to make text files lacking a trailing `<newline>` more portable. It is recognized that this will require a user-specified option or extension for implementations that permit `ex` and `vi` to edit files of type other than text if such files are not otherwise identified by the system. It was agreed that the ability to edit files of arbitrary type can be useful, but it was not considered necessary to mandate that an `ex` or `vi` implementation be required to handle files other than text files.

The paragraph in the INPUT FILES section, "By default, ...", is intended to close a long-standing security problem in `ex` and `vi`; that of the "modeline" or "modelines" edit option. This feature allows any line in the first or last five lines of the file containing the strings `"ex:"` or `"vi:"` (and, apparently, `"ei:"` or `"vx:"`) to be a line containing editor commands, and `ex` interprets all the text up to the next `':'` or `<newline>` as a command. Consider the

consequences, for example, of an unsuspecting user using *ex* or *vi* as the editor when replying to a mail message in which a line such as:

```
ex:! rm -rf :
```

appeared in the signature lines. The standard developers believed strongly that an editor should not by default interpret any lines of a file. Vendors are strongly urged to delete this feature from their implementations of *ex* and *vi*.

### Asynchronous Events

The intention of the phrase “complete write” is that the entire edit buffer be written to stable storage. The note regarding temporary files is intended for implementations that use temporary files to back edit buffers unnamed by the user.

Historically, SIGQUIT was ignored by *ex*, but was the equivalent of the **Q** command in visual mode; that is, it exited visual mode and entered *ex* mode. POSIX.1-2008 permits, but does not require, this behavior. Historically, SIGINT was often used by *vi* users to terminate text input mode (<control>-C is often easier to enter than <ESC>). Some implementations of *vi* alerted the terminal on this event, and some did not. POSIX.1-2008 requires that SIGINT behave identically to <ESC>, and that the terminal not be alerted.

Historically, suspending the *ex* editor during text input mode was similar to SIGINT, as completed lines were retained, but any partial line discarded, and the editor returned to command mode. POSIX.1-2008 is silent on this issue; implementations are encouraged to follow historical practice, where possible.

Historically, the *vi* editor did not treat SIGTSTP as an asynchronous event, and it was therefore impossible to suspend the editor in visual text input mode. There are two major reasons for this. The first is that SIGTSTP is a broadcast signal on UNIX systems, and the chain of events where the shell *execs* an application that then *execs vi* usually caused confusion for the terminal state if SIGTSTP was delivered to the process group in the default manner. The second was that most implementations of the UNIX *curses* package did not handle SIGTSTP safely, and the receipt of SIGTSTP at the wrong time would cause them to crash. POSIX.1-2008 is silent on this issue; implementations are encouraged to treat suspension as an asynchronous event if possible.

Historically, modifications to the edit buffer made before SIGINT interrupted an operation were retained; that is, anywhere from zero to all of the lines to be modified might have been modified by the time the SIGINT arrived. These changes were not discarded by the arrival of SIGINT. POSIX.1-2008 permits this behavior, noting that the **undo** command is required to be able to undo these partially completed commands.

The action taken for signals other than SIGINT, SIGCONT, SIGHUP, and SIGTERM is unspecified because some implementations attempt to save the edit buffer in a useful state when other signals are received.

### Standard Error

For *ex/vi*, diagnostic messages are those messages reported as a result of a failed attempt to invoke *ex* or *vi*, such as invalid options or insufficient resources, or an abnormal termination condition. Diagnostic messages should not be confused with the error messages generated by inappropriate or illegal user commands.

## Initialization in *ex* and *vi*

If an *ex* command (other than **cd**, **chdir**, or **source**) has a filename argument, one or both of the alternate and current pathnames will be set. Informally, they are set as follows:

1. If the *ex* command is one that replaces the contents of the edit buffer, and it succeeds, the current pathname will be set to the filename argument (the first filename argument in the case of the **next** command) and the alternate pathname will be set to the previous current pathname, if there was one.
2. In the case of the file read/write forms of the **read** and **write** commands, if there is no current pathname, the current pathname will be set to the filename argument.
3. Otherwise, the alternate pathname will be set to the filename argument.

For example, **:edit foo** and **:recover foo**, when successful, set the current pathname, and, if there was a previous current pathname, the alternate pathname. The commands **:write**, **!command**, and **:edit** set neither the current or alternate pathnames. If the **:edit foo** command were to fail for some reason, the alternate pathname would be set. The **read** and **write** commands set the alternate pathname to their *file* argument, unless the current pathname is not set, in which case they set the current pathname to their *file* arguments. The alternate pathname was not historically set by the **:source** command. POSIX.1-2008 requires conformance to historical practice. Implementations adding commands that take filenames as arguments are encouraged to set the alternate pathname as described here.

Historically, *ex* and *vi* read the **.exrc** file in the *\$HOME* directory twice, if the editor was executed in the *\$HOME* directory. POSIX.1-2008 prohibits this behavior.

Historically, the 4 BSD *ex* and *vi* read the *\$HOME* and local **.exrc** files if they were owned by the real ID of the user, or the **sourceany** option was set, regardless of other considerations. This was a security problem because it is possible to put normal UNIX system commands inside a **.exrc** file. POSIX.1-2008 does not specify the **sourceany** option, and historical implementations are encouraged to delete it.

The **.exrc** files must be owned by the real ID of the user, and not writable by anyone other than the owner. The appropriate privileges exception is intended to permit users to acquire special privileges, but continue to use the **.exrc** files in their home directories.

System V Release 3.2 and later *vi* implementations added the option **[no]exrc**. The behavior is that local **.exrc** files are read-only if the **exrc** option is set. The default for the **exrc** option was off, so by default, local **.exrc** files were not read. The problem this was intended to solve was that System V permitted users to give away files, so there is no possible ownership or writeability test to ensure that the file is safe. This is still a security problem on systems where users can give away files, but there is nothing additional that POSIX.1-2008 can do. The implementation-defined exception is intended to permit groups to have local **.exrc** files that are shared by users, by creating pseudo-users to own the shared files.

POSIX.1-2008 does not mention system-wide *ex* and *vi* start-up files. While they exist in several implementations of *ex* and *vi*, they are not present in any implementations considered historical practice by POSIX.1-2008. Implementations that have such files should use them only if they are owned by the real user ID or an appropriate user (for example, root on UNIX systems) and if they are not writable by any user other than their owner. System-wide start-up files should be read before the *EXINIT* variable, *\$HOME/.exrc*, or local **.exrc** files are evaluated.

Historically, any *ex* command could be entered in the *EXINIT* variable or the **.exrc** file, although ones requiring that the edit buffer already contain lines of text generally caused historical implementations of the editor to drop **core**. POSIX.1-2008 requires that any *ex* command be

permitted in the *EXINIT* variable and *.exrc* files, for simplicity of specification and consistency, although many of them will obviously fail under many circumstances.

The initialization of the contents of the edit buffer uses the phrase “the effect shall be” with regard to various *ex* commands. The intent of this phrase is that edit buffer contents loaded during the initialization phase not be lost; that is, loading the edit buffer should fail if the *.exrc* file read in the contents of a file and did not subsequently write the edit buffer. An additional intent of this phrase is to specify that the initial current line and column is set as specified for the individual *ex* commands.

Historically, the *-t* option behaved as if the tag search were a *+command*; that is, it was executed from the last line of the file specified by the tag. This resulted in the search failing if the pattern was a forward search pattern and the **wrapsan** edit option was not set. POSIX.1-2008 does not permit this behavior, requiring that the search for the tag pattern be performed on the entire file, and, if not found, that the current line be set to a more reasonable location in the file.

Historically, the empty edit buffer presented for editing when a file was not specified by the user was unnamed. This is permitted by POSIX.1-2008; however, implementations are encouraged to provide users a temporary filename for this buffer because it permits them the use of *ex* commands that use the current pathname during temporary edit sessions.

Historically, the file specified using the *-t* option was not part of the current argument list. This practice is permitted by POSIX.1-2008; however, implementations are encouraged to include its name in the current argument list for consistency.

Historically, the *-c* command was generally not executed until a file that already exists was edited. POSIX.1-2008 requires conformance to this historical practice. Commands that could cause the *-c* command to be executed include the *ex* commands **edit**, **next**, **recover**, **rewind**, and **tag**, and the *vi* commands *<control>-^* and *<control>-]*. Historically, reading a file into an edit buffer did not cause the *-c* command to be executed (even though it might set the current pathname) with the exception that it did cause the *-c* command to be executed if: the editor was in *ex* mode, the edit buffer had no current pathname, the edit buffer was empty, and no read commands had yet been attempted. For consistency and simplicity of specification, POSIX.1-2008 does not permit this behavior.

Historically, the *-r* option was the same as a normal edit session if there was no recovery information available for the file. This allowed users to enter:

```
vi -r *.c
```

and recover whatever files were recoverable. In some implementations, recovery was attempted only on the first file named, and the file was not entered into the argument list; in others, recovery was attempted for each file named. In addition, some historical implementations ignored *-r* if *-t* was specified or did not support command line *file* arguments with the *-t* option. For consistency and simplicity of specification, POSIX.1-2008 disallows these special cases, and requires that recovery be attempted the first time each file is edited.

Historically, *vi* initialized the ‘ and ’ marks, but *ex* did not. This meant that if the first command in *ex* mode was **visual** or if an *ex* command was executed first (for example, *vi +10 file*), *vi* was entered without the marks being initialized. Because the standard developers believed the marks to be generally useful, and for consistency and simplicity of specification, POSIX.1-2008 requires that they always be initialized if in open or visual mode, or if in *ex* mode and the edit buffer is not empty. Not initializing it in *ex* mode if the edit buffer is empty is historical practice; however, it has always been possible to set (and use) marks in empty edit buffers in open and visual mode edit sessions.

## Addressing

Historically, *ex* and *vi* accepted the additional addressing forms '`\/'` and '`\?'`'. They were equivalent to '`//'`' and '`??`', respectively. They are not required by POSIX.1-2008, mostly because nobody can remember whether they ever did anything different historically.

Historically, *ex* and *vi* permitted an address of zero for several commands, and permitted the % address in empty files for others. For consistency, POSIX.1-2008 requires support for the former in the few commands where it makes sense, and disallows it otherwise. In addition, because POSIX.1-2008 requires that % be logically equivalent to '`1,$`', it is also supported where it makes sense and disallowed otherwise.

Historically, the % address could not be followed by further addresses. For consistency and simplicity of specification, POSIX.1-2008 requires that additional addresses be supported.

All of the following are valid *addresses*:

```
+++ Three lines after the current line.
/re/- One line before the next occurrence of re.
-2 Two lines before the current line.
3 ---- 2 Line one (note intermediate negative address).
1 2 3 Line six.
```

Any number of addresses can be provided to commands taking addresses; for example, '`1,2,3,4,5p`' prints lines 4 and 5, because two is the greatest valid number of addresses accepted by the **print** command. This, in combination with the <semicolon> delimiter, permits users to create commands based on ordered patterns in the file. For example, the command **3;/foo/;+2print** will display the first line after line 3 that contains the pattern *foo*, plus the next two lines. Note that the address **3;** must be evaluated before being discarded because the search origin for the **/foo/** command depends on this.

Historically, values could be added to addresses by including them after one or more <blank> characters; for example, **3 - 5p** wrote the seventh line of the file, and **/foo/ 5** was the same as **/foo/+5**. However, only absolute values could be added; for example, **5 /foo/** was an error. POSIX.1-2008 requires conformance to historical practice. Address offsets are separately specified from addresses because they could historically be provided to visual mode search commands.

Historically, any missing addresses defaulted to the current line. This was true for leading and trailing <comma>-delimited addresses, and for trailing <semicolon>-delimited addresses. For consistency, POSIX.1-2008 requires it for leading <semicolon> addresses as well.

Historically, *ex* and *vi* accepted the '`^`' character as both an address and as a flag offset for commands. In both cases it was identical to the '`-`' character. POSIX.1-2008 does not require or prohibit this behavior.

Historically, the enhancements to basic regular expressions could be used in addressing; for example, '`~'`', '`\<`', and '`\>`'. POSIX.1-2008 requires conformance to historical practice; that is, that regular expression usage be consistent, and that regular expression enhancements be supported wherever regular expressions are used.

## Command Line Parsing in ex

Historical *ex* command parsing was even more complex than that described here. POSIX.1-2008 requires the subset of the command parsing that the standard developers believed was documented and that users could reasonably be expected to use in a portable fashion, and that was historically consistent between implementations. (The discarded functionality is obscure, at best.) Historical implementations will require changes in order to comply with POSIX.1-2008; however, users are not expected to notice any of these changes. Most of the complexity in *ex* parsing is to handle three special termination cases:

1. The **!**, **global**, **v**, and the filter versions of the **read** and **write** commands are delimited by `<newline>` characters (they can contain `<vertical-line>` characters that are usually shell pipes).
2. The **ex**, **edit**, **next**, and **visual** in open and visual mode commands all take *ex* commands, optionally containing `<vertical-line>` characters, as their first arguments.
3. The **s** command takes a regular expression as its first argument, and uses the delimiting characters to delimit the command.

Historically, `<vertical-line>` characters in the `+command` argument of the **ex**, **edit**, **next**, **vi**, and **visual** commands, and in the *pattern* and *replacement* parts of the **s** command, did not delimit the command, and in the filter cases for **read** and **write**, and the **!**, **global**, and **v** commands, they did not delimit the command at all. For example, the following commands are all valid:

```
:edit +25 | s/abc/ABC/ file.c
:s/ | /PIPE/
:read !spell % | columnate
:global/pattern/p | l
:s/a/b/ | s/c/d | set
```

Historically, empty or `<blank>` filled lines in **.exrc** files and **sourced** files (as well as *EXINIT* variables and *ex* command scripts) were treated as default commands; that is, **print** commands. POSIX.1-2008 specifically requires that they be ignored when encountered in **.exrc** and **sourced** files to eliminate a common source of new user error.

Historically, *ex* commands with multiple adjacent (or `<blank>`-separated) vertical lines were handled oddly when executed from *ex* mode. For example, the command `l | l <carriage-return>`, when the cursor was on line 1, displayed lines 2, 3, and 5 of the file. In addition, the command `l` would only display the line after the next line, instead of the next two lines. The former worked more logically when executed from *vi* mode, and displayed lines 2, 3, and 4. POSIX.1-2008 requires the *vi* behavior; that is, a single default command and line number increment for each command separator, and trailing `<newline>` characters after `<vertical-line>` separators are discarded.

Historically, *ex* permitted a single extra `<colon>` as a leading command character; for example, **:g/pattern/:p** was a valid command. POSIX.1-2008 generalizes this to require that any number of leading `<colon>` characters be stripped.

Historically, any prefix of the **delete** command could be followed without intervening `<blank>` characters by a flag character because in the command **d p**, *p* is interpreted as the buffer *p*. POSIX.1-2008 requires conformance to historical practice.

Historically, the **k** command could be followed by the mark name without intervening `<blank>` characters. POSIX.1-2008 requires conformance to historical practice.

Historically, the **s** command could be immediately followed by flag and option characters; for example, **s/e/E/|s|sgc3p** was a valid command. However, flag characters could not stand alone;



for example, the commands **sp** and **s l** would fail, while the command **sgp** and **s gl** would succeed. (Obviously, the '#' flag character was used as a delimiter character if it followed the command.) Another issue was that option characters had to precede flag characters even when the command was fully specified; for example, the command **s/e/E/pg** would fail, while the command **s/e/E/gp** would succeed. POSIX.1-2008 requires conformance to historical practice.

Historically, the first command name that had a prefix matching the input from the user was the executed command; for example, **ve**, **ver**, and **vers** all executed the **version** command. Commands were in a specific order, however, so that **a** matched **append**, not **abbreviate**. POSIX.1-2008 requires conformance to historical practice. The restriction on command search order for implementations with extensions is to avoid the addition of commands such that the historical prefixes would fail to work portably.

Historical implementations of *ex* and *vi* did not correctly handle multiple *ex* commands, separated by <vertical-line> characters, that entered or exited visual mode or the editor. Because implementations of *vi* exist that do not exhibit this failure mode, POSIX.1-2008 does not permit it.

The requirement that alphabetic command names consist of all following alphabetic characters up to the next non-alphabetic character means that alphabetic command names must be separated from their arguments by one or more non-alphabetic characters, normally a <blank> or '!' character, except as specified for the exceptions, the **delete**, **k**, and **s** commands.

Historically, the repeated execution of the *ex* default **print** commands (<control>-D, *eof*, <newline>, <carriage-return>) erased any prompting character and displayed the next lines without scrolling the terminal; that is, immediately below any previously displayed lines. This provided a cleaner presentation of the lines in the file for the user. POSIX.1-2008 does not require this behavior because it may be impossible in some situations; however, implementations are strongly encouraged to provide this semantic if possible.

Historically, it was possible to change files in the middle of a command, and have the rest of the command executed in the new file; for example:

```
:edit +25 file.c | s/abc/ABC/ | 1
```

was a valid command, and the substitution was attempted in the newly edited file. POSIX.1-2008 requires conformance to historical practice. The following commands are examples that exercise the *ex* parser:

```
echo 'foo | bar' > file1; echo 'foo/bar' > file2;
vi
:edit +1 | s/|/PIPE/ | w file1 | e file2 | 1 | s/\\//SLASH/ | wq
```

Historically, there was no protection in editor implementations to avoid *ex* **global**, **v**, **@**, or **\*** commands changing edit buffers during execution of their associated commands. Because this would almost invariably result in catastrophic failure of the editor, and implementations exist that do exhibit these problems, POSIX.1-2008 requires that changing the edit buffer during a **global** or **v** command, or during a **@** or **\*** command for which there will be more than a single execution, be an error. Implementations supporting multiple edit buffers simultaneously are strongly encouraged to apply the same semantics to switching between buffers as well.

The *ex* command quoting required by POSIX.1-2008 is a superset of the quoting in historical implementations of the editor. For example, it was not historically possible to escape a <blank> in a filename; for example, **:edit foo\\\ bar** would report that too many filenames had been entered for the edit command, and there was no method of escaping a <blank> in the first argument of an **edit**, **ex**, **next**, or **visual** command at all. POSIX.1-2008 extends historical practice, requiring that quoting behavior be made consistent across all *ex* commands, except for

the **map**, **unmap**, **abbreviate**, and **unabbreviate** commands, which historically used <control>-V instead of <backslash> characters for quoting. For those four commands, POSIX.1-2008 requires conformance to historical practice.

Backslash quoting in *ex* is non-intuitive. <backslash>-escapes are ignored unless they escape a special character; for example, when performing *file* argument expansion, the string "\\%" is equivalent to '\%', not "\<current pathname>". This can be confusing for users because <backslash> is usually one of the characters that causes shell expansion to be performed, and therefore shell quoting rules must be taken into consideration. Generally, quoting characters are only considered if they escape a special character, and a quoting character must be provided for each layer of parsing for which the character is special. As another example, only a single <backslash> is necessary for the '\1' sequence in substitute replacement patterns, because the character '1' is not special to any parsing layer above it.

<control>-V quoting in *ex* is slightly different from backslash quoting. In the four commands where <control>-V quoting applies (**abbreviate**, **unabbreviate**, **map**, and **unmap**), any character may be escaped by a <control>-V whether it would have a special meaning or not. POSIX.1-2008 requires conformance to historical practice.

Historical implementations of the editor did not require delimiters within character classes to be escaped; for example, the command :s/[/] on the string "xxx/yyy" would delete the '/' from the string. POSIX.1-2008 disallows this historical practice for consistency and because it places a large burden on implementations by requiring that knowledge of regular expressions be built into the editor parser.

Historically, quoting <newline> characters in *ex* commands was handled inconsistently. In most cases, the <newline> character always terminated the command, regardless of any preceding escape character, because <backslash> characters did not escape <newline> characters for most *ex* commands. However, some *ex* commands (for example, **s**, **map**, and **abbreviation**) permitted <newline> characters to be escaped (although in the case of **map** and **abbreviation**, <control>-V characters escaped them instead of <backslash> characters). This was true in not only the command line, but also **.exrc** and **sourced** files. For example, the command:

```
map = foo<control-V><newline>bar
```

would succeed, although it was sometimes difficult to get the <control>-V and the inserted <newline> passed to the *ex* parser. For consistency and simplicity of specification, POSIX.1-2008 requires that it be possible to escape <newline> characters in *ex* commands at all times, using <backslash> characters for most *ex* commands, and using <control>-V characters for the **map** and **abbreviation** commands. For example, the command **print<newline>list** is required to be parsed as the single command **print<newline>list**. While this differs from historical practice, POSIX.1-2008 developers believed it unlikely that any script or user depended on the historical behavior.

Historically, an error in a command specified using the **-c** option did not cause the rest of the **-c** commands to be discarded. POSIX.1-2008 disallows this for consistency with mapped keys, the **@**, **global**, **source**, and **v** commands, the *EXINIT* environment variable, and the **.exrc** files.

## Input Editing in ex

One of the common uses of the historical *ex* editor is over slow network connections. Editors that run in canonical mode can require far less traffic to and from, and far less processing on, the host machine, as well as more easily supporting block-mode terminals. For these reasons, POSIX.1-2008 requires that *ex* be implemented using canonical mode input processing, as was done historically.

POSIX.1-2008 does not require the historical 4 BSD input editing characters “word erase” or “literal next”. For this reason, it is unspecified how they are handled by *ex*, although they must have the required effect. Implementations that resolve them after the line has been ended using a <newline> or <control>-M character, and implementations that rely on the underlying system terminal support for this processing, are both conforming. Implementations are strongly urged to use the underlying system functionality, if at all possible, for compatibility with other system text input interfaces.

Historically, when the *eof* character was used to decrement the **autoindent** level, the cursor moved to display the new end of the **autoindent** characters, but did not move the cursor to a new line, nor did it erase the <control>-D character from the line. POSIX.1-2008 does not specify that the cursor remain on the same line or that the rest of the line is erased; however, implementations are strongly encouraged to provide the best possible user interface; that is, the cursor should remain on the same line, and any <control>-D character on the line should be erased.

POSIX.1-2008 does not require the historical 4 BSD input editing character “reprint”, traditionally <control>-R, which redisplayed the current input from the user. For this reason, and because the functionality cannot be implemented after the line has been terminated by the user, POSIX.1-2008 makes no requirements about this functionality. Implementations are strongly urged to make this historical functionality available, if possible.

Historically, <control>-Q did not perform a literal next function in *ex*, as it did in *vi*. POSIX.1-2008 requires conformance to historical practice to avoid breaking historical *ex* scripts and **.exrc** files.

## eof

Whether the *eof* character immediately modifies the **autoindent** characters in the prompt is left unspecified so that implementations can conform in the presence of systems that do not support this functionality. Implementations are encouraged to modify the line and redisplay it immediately, if possible.

The specification of the handling of the *eof* character differs from historical practice only in that *eof* characters are not discarded if they follow normal characters in the text input. Historically, they were always discarded.

## Command Descriptions in ex

Historically, several commands (for example, **global**, **v**, **visual**, **s**, **write**, **wq**, **yank**, **!**, **<**, **>**, **&**, and **~**) were executable in empty files (that is, the default address(es) were 0), or permitted explicit addresses of 0 (for example, 0 was a valid address, or 0,0 was a valid range). Addresses of 0, or command execution in an empty file, make sense only for commands that add new text to the edit buffer or write commands (because users may wish to write empty files). POSIX.1-2008 requires this behavior for such commands and disallows it otherwise, for consistency and simplicity of specification.

A count to an *ex* command has been historically corrected to be no greater than the last line in a

file; for example, in a five-line file, the command **1,6print** would fail, but the command **1print300** would succeed. POSIX.1-2008 requires conformance to historical practice.

Historically, the use of flags in *ex* commands could be obscure. General historical practice was as described by POSIX.1-2008, but there were some special cases. For instance, the **list**, **number**, and **print** commands ignored trailing address offsets; for example, **3p +++#** would display line 3, and 3 would be the current line after the execution of the command. The **open** and **visual** commands ignored both the trailing offsets and the trailing flags. Also, flags specified to the **open** and **visual** commands interacted badly with the **list** edit option, and setting and then unsetting it during the open/visual session would cause *vi* to stop displaying lines in the specified format. For consistency and simplicity of specification, POSIX.1-2008 does not permit any of these exceptions to the general rule.

POSIX.1-2008 uses the word *copy* in several places when discussing buffers. This is not intended to imply implementation.

Historically, *ex* users could not specify numeric buffers because of the ambiguity this would cause; for example, in the command **3 delete 2**, it is unclear whether 2 is a buffer name or a *count*. POSIX.1-2008 requires conformance to historical practice by default, but does not preclude extensions.

Historically, the contents of the unnamed buffer were frequently discarded after commands that did not explicitly affect it; for example, when using the **edit** command to switch files. For consistency and simplicity of specification, POSIX.1-2008 does not permit this behavior.

The *ex* utility did not historically have access to the numeric buffers, and, furthermore, deleting lines in *ex* did not modify their contents. For example, if, after doing a delete in *vi*, the user switched to *ex*, did another delete, and then switched back to *vi*, the contents of the numeric buffers would not have changed. POSIX.1-2008 requires conformance to historical practice. Numeric buffers are described in the *ex* utility in order to confine the description of buffers to a single location in POSIX.1-2008.

The metacharacters that trigger shell expansion in *file* arguments match historical practice, as does the method for doing shell expansion. Implementations wishing to provide users with the flexibility to alter the set of metacharacters are encouraged to provide a **shellmeta** string edit option.

Historically, *ex* commands executed from *vi* refreshed the screen when it did not strictly need to do so; for example, **!date > /dev/null** does not require a screen refresh because the output of the UNIX *date* command requires only a single line of the screen. POSIX.1-2008 requires that the screen be refreshed if it has been overwritten, but makes no requirements as to how an implementation should make that determination. Implementations may prompt and refresh the screen regardless.

## Abbreviate

Historical practice was that characters that were entered as part of an abbreviation replacement were subject to **map** expansions, the **showmatch** edit option, further abbreviation expansions, and so on; that is, they were logically pushed onto the terminal input queue, and were not a simple replacement. POSIX.1-2008 requires conformance to historical practice. Historical practice was that whenever a non-word character (that had not been escaped by a <control>-V) was entered after a word character, *vi* would check for abbreviations. The check was based on the type of the character entered before the word character of the word/non-word pair that triggered the check. The word character of the word/non-word pair that triggered the check and all characters entered before the trigger pair that were of that type were included in the check, with the exception of <blank> characters, which always delimited the abbreviation.

This means that, for the abbreviation to work, the *lhs* must end with a word character, there can be no transitions from word to non-word characters (or *vice versa*) other than between the last and next-to-last characters in the *lhs*, and there can be no <blank> characters in the *lhs*. In addition, because of the historical quoting rules, it was impossible to enter a literal <control>-V in the *lhs*. POSIX.1-2008 requires conformance to historical practice. Historical implementations did not inform users when abbreviations that could never be used were entered; implementations are strongly encouraged to do so.

For example, the following abbreviations will work:

```
:ab (p REPLACE
:ab p REPLACE
:ab ((p REPLACE
```

The following abbreviations will not work:

```
:ab (REPLACE
:ab (pp REPLACE
```

Historical practice is that words on the *vi* colon command line were subject to abbreviation expansion, including the arguments to the **abbrev** (and more interestingly) the **unabbrev** command. Because there are implementations that do not do abbreviation expansion for the first argument to those commands, this is permitted, but not required, by POSIX.1-2008. However, the following sequence:

```
:ab foo bar
:ab foo baz
```

resulted in the addition of an abbreviation of "baz" for the string "bar" in historical *ex/vi*, and the sequence:

```
:ab foo1 bar
:ab foo2 bar
:unabbreviate foo2
```

deleted the abbreviation "foo1", not "foo2". These behaviors are not permitted by POSIX.1-2008 because they clearly violate the expectations of the user.

It was historical practice that <control>-V, not <backslash>, characters be interpreted as escaping subsequent characters in the **abbreviate** command. POSIX.1-2008 requires conformance to historical practice; however, it should be noted that an abbreviation containing a <blank> will never work.

## Append

Historically, any text following a <vertical-line> command separator after an **append**, **change**, or **insert** command became part of the insert text. For example, in the command:

```
:g/pattern/append|stuff1
```

a line containing the text "stuff1" would be appended to each line matching pattern. It was also historically valid to enter:

```
:append|stuff1
stuff2
.
```

and the text on the *ex* command line would be appended along with the text inserted after it. There was an historical bug, however, that the user had to enter two terminating lines (the ' . '

lines) to terminate text input mode in this case. POSIX.1-2008 requires conformance to historical practice, but disallows the historical need for multiple terminating lines.

### Change

See the RATIONALE for the **append** command. Historical practice for cursor positioning after the change command when no text is input, is as described in POSIX.1-2008. However, one System V implementation is known to have been modified such that the cursor is positioned on the first address specified, and not on the line before the first address. POSIX.1-2008 disallows this modification for consistency.

Historically, the **change** command did not support buffer arguments, although some implementations allow the specification of an optional buffer. This behavior is neither required nor disallowed by POSIX.1-2008.

### Change Directory

A common extension in *ex* implementations is to use the elements of a **cdpath** edit option as prefix directories for *path* arguments to **chdir** that are relative pathnames and that do not have `'.'` or `".."` as their first component. Elements in the **cdpath** edit option are `<colon>`-separated. The initial value of the **cdpath** edit option is the value of the shell *CDPATH* environment variable. This feature was not included in POSIX.1-2008 because it does not exist in any of the implementations considered historical practice.

### Copy

Historical implementations of *ex* permitted copies to lines inside of the specified range; for example, **:2,5copy3** was a valid command. POSIX.1-2008 requires conformance to historical practice.

### Delete

POSIX.1-2008 requires support for the historical parsing of a **delete** command followed by flags, without any intervening `<blank>` characters. For example:

**1dp** Deletes the first line and prints the line that was second.

**1delep** As for **1dp**.

**1d** Deletes the first line, saving it in buffer *p*.

**1d p1l** (Pee-one-ell.) Deletes the first line, saving it in buffer *p*, and listing the line that was second.

### Edit

Historically, any *ex* command could be entered as a *+command* argument to the **edit** command, although some (for example, **insert** and **append**) were known to confuse historical implementations. For consistency and simplicity of specification, POSIX.1-2008 requires that any command be supported as an argument to the **edit** command.

Historically, the command argument was executed with the current line set to the last line of the file, regardless of whether the **edit** command was executed from visual mode or not. POSIX.1-2008 requires conformance to historical practice.

Historically, the *+command* specified to the **edit** and **next** commands was delimited by the first `<blank>`, and there was no way to quote them. For consistency, POSIX.1-2008 requires that the usual *ex* backslash quoting be provided.

Historically, specifying the *+command* argument to the edit command required a filename to be specified as well; for example, **:edit +100** would always fail. For consistency and simplicity of specification, POSIX.1-2008 does not permit this usage to fail for that reason.

Historically, only the cursor position of the last file edited was remembered by the editor. POSIX.1-2008 requires that this be supported; however, implementations are permitted to remember and restore the cursor position for any file previously edited.

## File

Historical versions of the *ex* editor **file** command displayed a current line and number of lines in the edit buffer of 0 when the file was empty, while the *vi* <control>-G command displayed a current line and number of lines in the edit buffer of 1 in the same situation. POSIX.1-2008 does not permit this discrepancy, instead requiring that a message be displayed indicating that the file is empty.

## Global

The two-pass operation of the **global** and **v** commands is not intended to imply implementation, only the required result of the operation.

The current line and column are set as specified for the individual *ex* commands. This requirement is cumulative; that is, the current line and column must track across all the commands executed by the **global** or **v** commands.

## Insert

See the RATIONALE for the **append** command.

Historically, **insert** could not be used with an address of zero; that is, not when the edit buffer was empty. POSIX.1-2008 requires that this command behave consistently with the **append** command.

## Join

The action of the **join** command in relation to the special characters is only defined for the POSIX locale because the correct amount of white space after a period varies; in Japanese none is required, in French only a single space, and so on.

## List

The historical output of the **list** command was potentially ambiguous. The standard developers believed correcting this to be more important than adhering to historical practice, and POSIX.1-2008 requires unambiguous output.

## Map

Historically, command mode maps only applied to command names; for example, if the character 'x' was mapped to 'y', the command **fx** searched for the 'x' character, not the 'y' character. POSIX.1-2008 requires this behavior. Historically, entering <control>-V as the first character of a *vi* command was an error. Several implementations have extended the semantics of *vi* such that <control>-V means that the subsequent command character is not mapped. This is permitted, but not required, by POSIX.1-2008. Regardless, using <control>-V to escape the second or later character in a sequence of characters that might match a **map** command, or any character in text input mode, is historical practice, and stops the entered keys from matching a map. POSIX.1-2008 requires conformance to historical practice.

Historical implementations permitted digits to be used as a **map** command *lhs*, but then ignored the map. POSIX.1-2008 requires that the mapped digits not be ignored.

The historical implementation of the **map** command did not permit **map** commands that were more than a single character in length if the first character was printable. This behavior is permitted, but not required, by POSIX.1-2008.

Historically, mapped characters were remapped unless the **remap** edit option was not set, or the prefix of the mapped characters matched the mapping characters; for example, in the **map**:

```
:map ab abcd
```

the characters "ab" were used as is and were not remapped, but the characters "cd" were mapped if appropriate. This can cause infinite loops in the *vi* mapping mechanisms. POSIX.1-2008 requires conformance to historical practice, and that such loops be interruptible.

Text input maps had the same problems with expanding the *lhs* for the **ex map!** and **unmap!** command as did the **ex abbreviate** and **unabbreviate** commands. See the RATIONALE for the **ex abbreviate** command. POSIX.1-2008 requires similar modification of some historical practice for the **map** and **unmap** commands, as described for the **abbreviate** and **unabbreviate** commands.

Historically, **maps** that were subsets of other **maps** behaved differently depending on the order in which they were defined. For example:

```
:map! ab short
:map! abc long
```

would always translate the characters "ab" to "short", regardless of how fast the characters "abc" were entered. If the entry order was reversed:

```
:map! abc long
:map! ab short
```

the characters "ab" would cause the editor to pause, waiting for the completing 'c' character, and the characters might never be mapped to "short". For consistency and simplicity of specification, POSIX.1-2008 requires that the shortest match be used at all times.

The length of time the editor spends waiting for the characters to complete the *lhs* is unspecified because the timing capabilities of systems are often inexact and variable, and it may depend on other factors such as the speed of the connection. The time should be long enough for the user to be able to complete the sequence, but not long enough for the user to have to wait. Some implementations of *vi* have added a **keytime** option, which permits users to set the number of 0,1 seconds the editor waits for the completing characters. Because mapped terminal function and cursor keys tend to start with an <ESC> character, and <ESC> is the key ending *vi* text input mode, **maps** starting with <ESC> characters are generally exempted from this timeout period, or, at least timed out differently.

## Mark

Historically, users were able to set the "previous context" marks explicitly. In addition, the **ex** commands **"** and **"** and the *vi* commands **"**, **"**, **"**, and **"** all referred to the same mark. In addition, the previous context marks were not set if the command, with which the address setting the mark was associated, failed. POSIX.1-2008 requires conformance to historical practice. Historically, if marked lines were deleted, the mark was also deleted, but would reappear if the change was undone. POSIX.1-2008 requires conformance to historical practice.

The description of the special events that set the ' and ' marks matches historical practice. For example, historically the command **/a/** did not set the ' and ' marks, but the command



89032        **/a/,/b/delete** did.

## 89033        **Next**

89034        Historically, any *ex* command could be entered as a *+command* argument to the **next** command,  
89035        although some (for example, **insert** and **append**) were known to confuse historical  
89036        implementations. POSIX.1-2008 requires that any command be permitted and that it behave as  
89037        specified. The **next** command can accept more than one file, so usage such as:

89038        `next `ls [abc] ``

89039        is valid; it need not be valid for the **edit** or **read** commands, for example, because they expect  
89040        only one filename.

89041        Historically, the **next** command behaved differently from the **:rewind** command in that it  
89042        ignored the force flag if the **autowrite** flag was set. For consistency, POSIX.1-2008 does not  
89043        permit this behavior.

89044        Historically, the **next** command positioned the cursor as if the file had never been edited before,  
89045        regardless. POSIX.1-2008 does not permit this behavior, for consistency with the **edit** command.

89046        Implementations wanting to provide a counterpart to the **next** command that edited the  
89047        previous file have used the command **prev[ious]**, which takes no *file* argument. POSIX.1-2008  
89048        does not require this command.

## 89049        **Open**

89050        Historically, the **open** command would fail if the **open** edit option was not set. POSIX.1-2008  
89051        does not mention the **open** edit option and does not require this behavior. Some historical  
89052        implementations do not permit entering open mode from open or visual mode, only from *ex*  
89053        mode. For consistency, POSIX.1-2008 does not permit this behavior.

89054        Historically, entering open mode from the command line (that is, *vi +open*) resulted in  
89055        anomalous behaviors; for example, the *ex* file and *set* commands, and the *vi* command  
89056        <control>-G did not work. For consistency, POSIX.1-2008 does not permit this behavior.

89057        Historically, the **open** command only permitted ' / ' characters to be used as the search pattern  
89058        delimiter. For consistency, POSIX.1-2008 requires that the search delimiters used by the **s**, **global**,  
89059        and **v** commands be accepted as well.

## 89060        **Preserve**

89061        The **preserve** command does not historically cause the file to be considered unmodified for the  
89062        purposes of future commands that may exit the editor. POSIX.1-2008 requires conformance to  
89063        historical practice.

89064        Historical documentation stated that mail was not sent to the user when preserve was executed;  
89065        however, historical implementations did send mail in this case. POSIX.1-2008 requires  
89066        conformance to the historical implementations.

## Print

The writing of NUL by the **print** command is not specified as a special case because the standard developers did not want to require *ex* to support NUL characters. Historically, characters were displayed using the ARPA standard mappings, which are as follows:

1. Printable characters are left alone.
2. Control characters less than \177 are represented as '^' followed by the character offset from the '@' character in the ASCII map; for example, \007 is represented as '^G'.
3. \177 is represented as '^?' followed by '?'.

The display of characters having their eighth bit set was less standard. Existing implementations use hex (0x00), octal (\000), and a meta-bit display. (The latter displayed bytes that had their eighth bit set as the two characters "M-" followed by the seven-bit display as described above.) The latter probably has the best claim to historical practice because it was used for the **-v** option of 4 BSD and 4 BSD-derived versions of the *cat* utility since 1980.

No specific display format is required by POSIX.1-2008.

Explicit dependence on the ASCII character set has been avoided where possible, hence the use of the phrase an "implementation-defined multi-character sequence" for the display of non-printable characters in preference to the historical usage of, for instance, "^I" for the <tab>. Implementations are encouraged to conform to historical practice in the absence of any strong reason to diverge.

Historically, all *ex* commands beginning with the letter 'p' could be entered using capitalized versions of the commands; for example, **P[rint]**, **Pre[serve]**, and **Pu[t]** were all valid command names. POSIX.1-2008 permits, but does not require, this historical practice because capital forms of the commands are used by some implementations for other purposes.

## Put

Historically, an *ex* **put** command, executed from open or visual mode, was the same as the open or visual mode **P** command, if the buffer was named and was cut in character mode, and the same as the **p** command if the buffer was named and cut in line mode. If the unnamed buffer was the source of the text, the entire line from which the text was taken was usually **put**, and the buffer was handled as if in line mode, but it was possible to get extremely anomalous behavior. In addition, using the **Q** command to switch into *ex* mode, and then doing a **put** often resulted in errors as well, such as appending text that was unrelated to the (supposed) contents of the buffer. For consistency and simplicity of specification, POSIX.1-2008 does not permit these behaviors. All *ex* **put** commands are required to operate in line mode, and the contents of the buffers are not altered by changing the mode of the editor.

## Read

Historically, an *ex* **read** command executed from open or visual mode, executed in an empty file, left an empty line as the first line of the file. For consistency and simplicity of specification, POSIX.1-2008 does not permit this behavior. Historically, a **read** in open or visual mode from a program left the cursor at the last line read in, not the first. For consistency, POSIX.1-2008 does not permit this behavior.

Historical implementations of *ex* were unable to undo **read** commands that read from the output of a program. For consistency, POSIX.1-2008 does not permit this behavior.

Historically, the *ex* and *vi* message after a successful **read** or **write** command specified "characters", not "bytes". POSIX.1-2008 requires that the number of bytes be displayed, not the

number of characters, because it may be difficult in multi-byte implementations to determine the number of characters read. Implementations are encouraged to clarify the message displayed to the user.

Historically, reads were not permitted on files other than type regular, except that FIFO files could be read (probably only because they did not exist when *ex* and *vi* were originally written). Because the historical *ex* evaluated **read!** and **read !** equivalently, there can be no optional way to force the read. POSIX.1-2008 permits, but does not require, this behavior.

### Recover

Some historical implementations of the editor permitted users to recover the edit buffer contents from a previous edit session, and then exit without saving those contents (or explicitly discarding them). The intent of POSIX.1-2008 in requiring that the edit buffer be treated as already modified is to prevent this user error.

### Rewind

Historical implementations supported the **rewind** command when the user was editing the first file in the list; that is, the file that the **rewind** command would edit. POSIX.1-2008 requires conformance to historical practice.

### Substitute

Historically, *ex* accepted an **r** option to the **s** command. The effect of the **r** option was to use the last regular expression used in any command as the pattern, the same as the **~** command. The **r** option is not required by POSIX.1-2008. Historically, the **c** and **g** options were toggled; for example, the command **:s/abc/def/** was the same as **s/abc/def/ccccgggg**. For simplicity of specification, POSIX.1-2008 does not permit this behavior.

The tilde command is often used to replace the last search RE. For example, in the sequence:

```
s/red/blue/
/green
~
```

the **~** command is equivalent to:

```
s/green/blue/
```

Historically, *ex* accepted all of the following forms:

```
s/abc/def/
s/abc/def
s/abc/
s/abc
```

POSIX.1-2008 requires conformance to this historical practice.

The **s** command presumes that the **'^'** character only occupies a single column in the display. Much of the *ex* and *vi* specification presumes that the **<space>** only occupies a single column in the display. There are no known character sets for which this is not true.

Historically, the final column position for the substitute commands was based on previous column movements; a search for a pattern followed by a substitution would leave the column position unchanged, while a **0** command followed by a substitution would change the column position to the first non-**<blank>**. For consistency and simplicity of specification, POSIX.1-2008 requires that the final column position always be set to the first non-**<blank>**.

## Set

Historical implementations redisplayed all of the options for each occurrence of the **all** keyword. POSIX.1-2008 permits, but does not require, this behavior.

## Tag

No requirement is made as to where *ex* and *vi* shall look for the file referenced by the tag entry. Historical practice has been to look for the path found in the **tags** file, based on the current directory. A useful extension found in some implementations is to look based on the directory containing the tags file that held the entry, as well. No requirement is made as to which reference for the tag in the tags file is used. This is deliberate, in order to permit extensions such as multiple entries in a tags file for a tag.

Because users often specify many different tags files, some of which need not be relevant or exist at any particular time, POSIX.1-2008 requires that error messages about problem tags files be displayed only if the requested tag is not found, and then, only once for each time that the **tag** edit option is changed.

The requirement that the current edit buffer be unmodified is only necessary if the file indicated by the tag entry is not the same as the current file (as defined by the current pathname). Historically, the file would be reloaded if the filename had changed, as well as if the filename was different from the current pathname. For consistency and simplicity of specification, POSIX.1-2008 does not permit this behavior, requiring that the name be the only factor in the decision.

Historically, *vi* only searched for tags in the current file from the current cursor to the end of the file, and therefore, if the **wrapsan** option was not set, tags occurring before the current cursor were not found. POSIX.1-2008 considers this a bug, and implementations are required to search for the first occurrence in the file, regardless.

## Undo

The **undo** description deliberately uses the word “modified”. The **undo** command is not intended to undo commands that replace the contents of the edit buffer, such as **edit**, **next**, **tag**, or **recover**.

Cursor positioning after the **undo** command was inconsistent in the historical *vi*, sometimes attempting to restore the original cursor position (**global**, **undo**, and **v** commands), and sometimes, in the presence of maps, placing the cursor on the last line added or changed instead of the first. POSIX.1-2008 requires a simplified behavior for consistency and simplicity of specification.

## Version

The **version** command cannot be exactly specified since there is no widely-accepted definition of what the version information should contain. Implementations are encouraged to do something reasonably intelligent.

## Write

Historically, the *ex* and *vi* message after a successful **read** or **write** command specified “characters”, not “bytes”. POSIX.1-2008 requires that the number of bytes be displayed, not the number of characters because it may be difficult in multi-byte implementations to determine the number of characters written. Implementations are encouraged to clarify the message displayed to the user.

Implementation-defined tests are permitted so that implementations can make additional checks; for example, for locks or file modification times.

Historically, attempting to append to a nonexistent file caused an error. It has been left unspecified in POSIX.1-2008 to permit implementations to let the **write** succeed, so that the append semantics are similar to those of the historical *csh*.

Historical *vi* permitted empty edit buffers to be written. However, since the way *vi* got around dealing with “empty” files was to always have a line in the edit buffer, no matter what, it wrote them as files of a single, empty line. POSIX.1-2008 does not permit this behavior.

Historically, *ex* restored standard output and standard error to their values as of when *ex* was invoked, before writes to programs were performed. This could disturb the terminal configuration as well as be a security issue for some terminals. POSIX.1-2008 does not permit this, requiring that the program output be captured and displayed as if by the *ex* **print** command.

## Adjust Window

Historically, the line count was set to the value of the **scroll** option if the type character was end-of-file. This feature was broken on most historical implementations long ago, however, and is not documented anywhere. For this reason, POSIX.1-2008 is resolutely silent.

Historically, the **z** command was <blank>-sensitive and **z +** and **z -** did different things than **z+** and **z-** because the type could not be distinguished from a flag. (The commands **z .** and **z =** were historically invalid.) POSIX.1-2008 requires conformance to this historical practice.

Historically, the **z** command was further <blank>-sensitive in that the *count* could not be <blank>-delimited; for example, the commands **z= 5** and **z- 5** were also invalid. Because the *count* is not ambiguous with respect to either the type character or the flags, this is not permitted by POSIX.1-2008.

## Escape

Historically, *ex* filter commands only read the standard output of the commands, letting standard error appear on the terminal as usual. The *vi* utility, however, read both standard output and standard error. POSIX.1-2008 requires the latter behavior for both *ex* and *vi*, for consistency.

## Shift Left and Shift Right

Historically, it was possible to add shift characters to increase the effect of the command; for example, <<< outdented (or >>> indented) the lines 3 levels of indentation instead of the default 1. POSIX.1-2008 requires conformance to historical practice.

**<control>-D**

Historically, the <control>-D command erased the prompt, providing the user with an unbroken presentation of lines from the edit buffer. This is not required by POSIX.1-2008; implementations are encouraged to provide it if possible. Historically, the <control>-D command took, and then ignored, a *count*. POSIX.1-2008 does not permit this behavior.

**Write Line Number**

Historically, the *ex* = command, when executed in *ex* mode in an empty edit buffer, reported 0, and from open or visual mode, reported 1. For consistency and simplicity of specification, POSIX.1-2008 does not permit this behavior.

**Execute**

Historically, *ex* did not correctly handle the inclusion of text input commands (that is, **append**, **insert**, and **change**) in executed buffers. POSIX.1-2008 does not permit this exclusion for consistency.

Historically, the logical contents of the buffer being executed did not change if the buffer itself were modified by the commands being executed; that is, buffer execution did not support self-modifying code. POSIX.1-2008 requires conformance to historical practice.

Historically, the @ command took a range of lines, and the @ buffer was executed once per line, with the current line ( ' . ' ) set to each specified line. POSIX.1-2008 requires conformance to historical practice.

Some historical implementations did not notice if errors occurred during buffer execution. This, coupled with the ability to specify a range of lines for the *ex* @ command, makes it trivial to cause them to drop **core**. POSIX.1-2008 requires that implementations stop buffer execution if any error occurs, if the specified line doesn't exist, or if the contents of the edit buffer itself are replaced (for example, the buffer executes the *ex* :**edit** command).

**Regular Expressions in ex**

Historical practice is that the characters in the replacement part of the last **s** command—that is, those matched by entering a ' ~ ' in the regular expression—were not further expanded by the regular expression engine. So, if the characters contained the string "a . , " they would match 'a' followed by " . , " and not 'a' followed by any character. POSIX.1-2008 requires conformance to historical practice.

**Edit Options in ex**

The following paragraphs describe the historical behavior of some edit options that were not, for whatever reason, included in POSIX.1-2008. Implementations are strongly encouraged to only use these names if the functionality described here is fully supported.

**extended** The **extended** edit option has been used in some implementations of *vi* to provide extended regular expressions instead of basic regular expressions. This option was omitted from POSIX.1-2008 because it is not widespread historical practice.

**flash** The **flash** edit option historically caused the screen to flash instead of beeping on error. This option was omitted from POSIX.1-2008 because it is not found in some historical implementations.

|       |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|-------|------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 89269 | <b>hardtabs</b>  | The <b>hardtabs</b> edit option historically defined the number of columns between hardware tab settings. This option was omitted from POSIX.1-2008 because it was believed to no longer be generally useful.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 89270 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 89271 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 89272 | <b>modeline</b>  | The <b>modeline</b> (sometimes named <b>modelines</b> ) edit option historically caused <i>ex</i> or <i>vi</i> to read the five first and last lines of the file for editor commands. This option is a security problem, and vendors are strongly encouraged to delete it from historical implementations.                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 89273 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 89274 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 89275 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 89276 | <b>open</b>      | The <b>open</b> edit option historically disallowed the <i>ex</i> <b>open</b> and <b>visual</b> commands. This edit option was omitted because these commands are required by POSIX.1-2008.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 89277 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 89278 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 89279 | <b>optimize</b>  | The <b>optimize</b> edit option historically expedited text throughput by setting the terminal to not do automatic <carriage-return> characters when printing more than one logical line of output. This option was omitted from POSIX.1-2008 because it was intended for terminals without addressable cursors, which are rarely, if ever, still used.                                                                                                                                                                                                                                                                                                                                                                      |
| 89280 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 89281 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 89282 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 89283 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 89284 | <b>ruler</b>     | The <b>ruler</b> edit option has been used in some implementations of <i>vi</i> to present a current row/column ruler for the user. This option was omitted from POSIX.1-2008 because it is not widespread historical practice.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 89285 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 89286 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 89287 | <b>sourceany</b> | The <b>sourceany</b> edit option historically caused <i>ex</i> or <i>vi</i> to source start-up files that were owned by users other than the user running the editor. This option is a security problem, and vendors are strongly encouraged to remove it from their implementations.                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 89288 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 89289 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 89290 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 89291 | <b>timeout</b>   | The <b>timeout</b> edit option historically enabled the (now standard) feature of only waiting for a short period before returning keys that could be part of a macro. This feature was omitted from POSIX.1-2008 because its behavior is now standard, it is not widely useful, and it was rarely documented.                                                                                                                                                                                                                                                                                                                                                                                                               |
| 89292 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 89293 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 89294 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 89295 | <b>verbose</b>   | The <b>verbose</b> edit option has been used in some implementations of <i>vi</i> to cause <i>vi</i> to output error messages for common errors; for example, attempting to move the cursor past the beginning or end of the line instead of only alerting the screen. (The historical <i>vi</i> only alerted the terminal and presented no message for such errors. The historical editor option <b>terse</b> did not select when to present error messages, it only made existing error messages more or less verbose.) This option was omitted from POSIX.1-2008 because it is not widespread historical practice; however, implementors are encouraged to use it if they wish to provide error messages for naive users. |
| 89296 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 89297 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 89298 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 89299 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 89300 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 89301 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 89302 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 89303 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 89304 | <b>wraplen</b>   | The <b>wraplen</b> edit option has been used in some implementations of <i>vi</i> to specify an automatic margin measured from the left margin instead of from the right margin. This is useful when multiple screen sizes are being used to edit a single file. This option was omitted from POSIX.1-2008 because it is not widespread historical practice; however, implementors are encouraged to use it if they add this functionality.                                                                                                                                                                                                                                                                                  |
| 89305 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 89306 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 89307 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 89308 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 89309 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |

**autoindent, ai**

Historically, the command **0a** did not do any autoindentation, regardless of the current indentation of line 1. POSIX.1-2008 requires that any indentation present in line 1 be used.

**autoprint, ap**

Historically, the **autoprint** edit option was not completely consistent or based solely on modifications to the edit buffer. Exceptions were the **read** command (when reading from a file, but not from a filter), the **append**, **change**, **insert**, **global**, and **v** commands, all of which were not affected by **autoprint**, and the **tag** command, which was affected by **autoprint**. POSIX.1-2008 requires conformance to historical practice.

Historically, the **autoprint** option only applied to the last of multiple commands entered using <vertical-line> delimiters; for example, **delete** <newline> was affected by **autoprint**, but **delete | version** <newline> was not. POSIX.1-2008 requires conformance to historical practice.

**autowrite, aw**

Appending the '!' character to the *ex* **next** command to avoid performing an automatic write was not supported in historical implementations. POSIX.1-2008 requires that the behavior match the other *ex* commands for consistency.

**ignorecase, ic**

Historical implementations of case-insensitive matching (the **ignorecase** edit option) lead to counter-intuitive situations when uppercase characters were used in range expressions. Historically, the process was as follows:

1. Take a line of text from the edit buffer.
2. Convert uppercase to lowercase in text line.
3. Convert uppercase to lowercase in regular expressions, except in character class specifications.
4. Match regular expressions against text.

This would mean that, with **ignorecase** in effect, the text:

```
The cat sat on the mat
```

would be matched by

```
/^the/
```

but not by:

```
/^[A-Z]he/
```

For consistency with other commands implementing regular expressions, POSIX.1-2008 does not permit this behavior.



### paragraphs, para

The ISO POSIX-2:1993 standard made the default **paragraphs** and **sections** edit options implementation-defined, arguing they were historically oriented to the UNIX system *troff* text formatter, and a “portable user” could use the {, }, [[, ]], (, and ) commands in open or visual mode and have the cursor stop in unexpected places. POSIX.1-2008 specifies their values in the POSIX locale because the unusual grouping (they only work when grouped into two characters at a time) means that they cannot be used for general-purpose movement, regardless.

### readonly

Implementations are encouraged to provide the best possible information to the user as to the read-only status of the file, with the exception that they should not consider the current special privileges of the process. This provides users with a safety net because they must force the overwrite of read-only files, even when running with additional privileges.

The **readonly** edit option specification largely conforms to historical practice. The only difference is that historical implementations did not notice that the user had set the **readonly** edit option in cases where the file was already marked read-only for some reason, and would therefore reinitialize the **readonly** edit option the next time the contents of the edit buffer were replaced. This behavior is disallowed by POSIX.1-2008.

### report

The requirement that lines copied to a buffer interact differently than deleted lines is historical practice. For example, if the **report** edit option is set to 3, deleting 3 lines will cause a report to be written, but 4 lines must be copied before a report is written.

The requirement that the *ex* **global**, **v**, **open**, **undo**, and **visual** commands present reports based on the total number of lines added or deleted during the command execution, and that commands executed by the **global** and **v** commands not present reports, is historical practice. POSIX.1-2008 extends historical practice by requiring that buffer execution be treated similarly. The reasons for this are two-fold. Historically, only the report by the last command executed from the buffer would be seen by the user, as each new report would overwrite the last. In addition, the standard developers believed that buffer execution had more in common with **global** and **v** commands than it did with other *ex* commands, and should behave similarly, for consistency and simplicity of specification.

### showmatch, sm

The length of time the cursor spends on the matching character is unspecified because the timing capabilities of systems are often inexact and variable. The time should be long enough for the user to notice, but not long enough for the user to become annoyed. Some implementations of *vi* have added a **matchtime** option that permits users to set the number of 0.1 second intervals the cursor pauses on the matching character.

### showmode

The **showmode** option has been used in some historical implementations of *ex* and *vi* to display the current editing mode when in open or visual mode. The editing modes have generally included “command” and “input”, and sometimes other modes such as “replace” and “change”. The string was usually displayed on the bottom line of the screen at the far right-hand corner. In addition, a preceding ‘\*’ character often denoted whether the contents of the edit buffer had been modified. The latter display has sometimes been part of the **showmode** option, and sometimes based on another option. This option was not available in the 4 BSD historical

implementation of *vi*, but was viewed as generally useful, particularly to novice users, and is required by POSIX.1-2008.

The **smd** shorthand for the **showmode** option was not present in all historical implementations of the editor. POSIX.1-2008 requires it, for consistency.

Not all historical implementations of the editor displayed a mode string for command mode, differentiating command mode from text input mode by the absence of a mode string. POSIX.1-2008 permits this behavior for consistency with historical practice, but implementations are encouraged to provide a display string for both modes.

### slowopen

Historically, the **slowopen** option was automatically set if the terminal baud rate was less than 1 200 baud, or if the baud rate was 1 200 baud and the **redraw** option was not set. The **slowopen** option had two effects. First, when inserting characters in the middle of a line, characters after the cursor would not be pushed ahead, but would appear to be overwritten. Second, when creating a new line of text, lines after the current line would not be scrolled down, but would appear to be overwritten. In both cases, ending text input mode would cause the screen to be refreshed to match the actual contents of the edit buffer. Finally, terminals that were sufficiently intelligent caused the editor to ignore the **slowopen** option. POSIX.1-2008 permits most historical behavior, extending historical practice to require **slowopen** behaviors if the edit option is set by the user.

### tags

The default path for tags files is left unspecified as implementations may have their own **tags** implementations that do not correspond to the historical ones. The default **tags** option value should probably at least include the file **./tags**.

### term

Historical implementations of *ex* and *vi* ignored changes to the **term** edit option after the initial terminal information was loaded. This is permitted by POSIX.1-2008; however, implementations are encouraged to permit the user to modify their terminal type at any time.

### terse

Historically, the **terse** edit option optionally provided a shorter, less descriptive error message, for some error messages. This is permitted, but not required, by POSIX.1-2008. Historically, most common visual mode errors (for example, trying to move the cursor past the end of a line) did not result in an error message, but simply alerted the terminal. Implementations wishing to provide messages for novice users are urged to do so based on the **edit** option **verbose**, and not **terse**.

### window

In historical implementations, the default for the **window** edit option was based on the baud rate as follows:

1. If the baud rate was less than 1 200, the **edit** option **w300** set the window value; for example, the line:

```
set w300=12
```

would set the window option to 12 if the baud rate was less than 1 200.

2. If the baud rate was equal to 1 200, the **edit** option **w1200** set the window value.
3. If the baud rate was greater than 1 200, the **edit** option **w9600** set the window value.

The **w300**, **w1200**, and **w9600** options do not appear in POSIX.1-2008 because of their dependence on specific baud rates.

In historical implementations, the size of the window displayed by various commands was related to, but not necessarily the same as, the **window** edit option. For example, the size of the window was set by the *ex* command **visual 10**, but it did not change the value of the **window** edit option. However, changing the value of the **window** edit option did change the number of lines that were displayed when the screen was repainted. POSIX.1-2008 does not permit this behavior in the interests of consistency and simplicity of specification, and requires that all commands that change the number of lines that are displayed do it by setting the value of the **window** edit option.

#### **wrapmargin, wm**

Historically, the **wrapmargin** option did not affect maps inserting characters that also had associated *counts*; for example **:map K 5aABC DEF**. Unfortunately, there are widely used maps that depend on this behavior. For consistency and simplicity of specification, POSIX.1-2008 does not permit this behavior.

Historically, **wrapmargin** was calculated using the column display width of all characters on the screen. For example, an implementation using "**^I**" to represent <tab> characters when the **list** edit option was set, where '**^**' and '**I**' each took up a single column on the screen, would calculate the **wrapmargin** based on a value of 2 for each <tab>. The **number** edit option similarly changed the effective length of the line as well. POSIX.1-2008 requires conformance to historical practice.

Earlier versions of this standard allowed for implementations with bytes other than eight bits, but this has been modified in this version.

#### **FUTURE DIRECTIONS**

None.

#### **SEE ALSO**

[Section 2.9.1.1](#) (on page 2339), *ctags*, *ed*, *sed*, *sh*, *stty*, *vi*

[XBD Table 5-1](#) (on page 121), [Chapter 8](#) (on page 173), [Section 9.3](#) (on page 183), [Section 12.2](#) (on page 215)

XSH *access()*

#### **CHANGE HISTORY**

First released in Issue 2.

##### **Issue 5**

The FUTURE DIRECTIONS section is added.

##### **Issue 6**

This utility is marked as part of the User Portability Utilities option.

The obsolescent SYNOPSIS is removed, removing the *+command* and *-* options.

89467 The following new requirements on POSIX implementations derive from alignment with the  
89468 Single UNIX Specification:

- 89469 • In the **map** command description, the sequence *#digit* is added.
- 89470 • The **directory**, **edcompatible**, **redraw**, and **slowopen** edit options are added.

89471 The *ex* utility is extensively changed for alignment with the IEEE P1003.2b draft standard. This  
89472 includes changes as a result of the IEEE PASC Interpretations 1003.2 #31, #38, #49, #50, #51, #52,  
89473 #55, #56, #57, #61, #62, #63, #64, #65, and #78.

89474 The **-l** option is removed.

89475 IEEE Std 1003.1-2001/Cor 1-2002, item XCU/TC1/D6/23 is applied, correcting a URL.

89476 IEEE Std 1003.1-2001/Cor 2-2004, item XCU/TC2/D6/8 is applied, making an editorial  
89477 correction in the EXTENDED DESCRIPTION.

89478 IEEE Std 1003.1-2001/Cor 2-2004, item XCU/TC2/D6/9 is applied, removing text describing  
89479 behavior on systems with bytes consisting of more than eight bits.

#### 89480 Issue 7

89481 Austin Group Interpretation 1003.1-2001 #027 is applied, clarifying the behavior if an operand is  
89482 ' \_'.

89483 Austin Group Interpretation 1003.1-2001 #036 is applied, clarifying the behavior for BREs.

89484 Austin Group Interpretation 1003.1-2001 #121 is applied, clarifying the *ex* **write** command.

89485 Austin Group Interpretation 1003.1-2001 #156 is applied.

89486 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

89487 **NAME**

89488       expand — convert tabs to spaces

89489 **SYNOPSIS**89490       expand [-t *tablist*] [*file...*]89491 **DESCRIPTION**

89492       The *expand* utility shall write files or the standard input to the standard output with <tab>  
 89493       characters replaced with one or more <space> characters needed to pad to the next tab stop. Any  
 89494       <backspace> characters shall be copied to the output and cause the column position count for  
 89495       tab stop calculations to be decremented; the column position count shall not be decremented  
 89496       below zero.

89497 **OPTIONS**89498       The *expand* utility shall conform to XBD [Section 12.2](#) (on page 215).

89499       The following option shall be supported:

89500       -t *tablist*     Specify the tab stops. The application shall ensure that the argument *tablist* consists  
 89501       of either a single positive decimal integer or a list of tabstops. If a single number is  
 89502       given, tabs shall be set that number of column positions apart instead of the  
 89503       default 8.

89504       If a list of tabstops is given, the application shall ensure that it consists of a list of  
 89505       two or more positive decimal integers, separated by <blank> or <comma>  
 89506       characters, in ascending order. The <tab> characters shall be set at those specific  
 89507       column positions. Each tab stop *N* shall be an integer value greater than zero, and  
 89508       the list is in strictly ascending order. This is taken to mean that, from the start of a  
 89509       line of output, tabbing to position *N* shall cause the next character output to be in  
 89510       the (*N*+1)th column position on that line.

89511       In the event of *expand* having to process a <tab> at a position beyond the last of  
 89512       those specified in a multiple tab-stop list, the <tab> shall be replaced by a single  
 89513       <space> in the output.

89514 **OPERANDS**

89515       The following operand shall be supported:

89516       *file*           The pathname of a text file to be used as input.

89517 **STDIN**

89518       See the INPUT FILES section.

89519 **INPUT FILES**

89520       Input files shall be text files.

89521 **ENVIRONMENT VARIABLES**89522       The following environment variables shall affect the execution of *expand*:

89523       LANG            Provide a default value for the internationalization variables that are unset or null.  
 89524                       (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 89525                       variables used to determine the values of locale categories.)

89526       LC\_ALL          If set to a non-empty string value, override the values of all the other  
 89527                       internationalization variables.

89528       LC\_CTYPE        Determine the locale for the interpretation of sequences of bytes of text data as  
 89529                       characters (for example, single-byte as opposed to multi-byte characters in  
 89530                       arguments and input files), the processing of <tab> and <space> characters, and  
 89531                       for the determination of the width in column positions each character would

89532                    occupy on an output device.

89533            *LC\_MESSAGES*

89534                    Determine the locale that should be used to affect the format and contents of

89535                    diagnostic messages written to standard error.

89536 XSI            *NLSPATH*   Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

89537 **ASYNCHRONOUS EVENTS**

89538            Default.

89539 **STDOUT**

89540            The standard output shall be equivalent to the input files with <tab> characters converted into

89541            the appropriate number of <space> characters.

89542 **STDERR**

89543            The standard error shall be used only for diagnostic messages.

89544 **OUTPUT FILES**

89545            None.

89546 **EXTENDED DESCRIPTION**

89547            None.

89548 **EXIT STATUS**

89549            The following exit values shall be returned:

89550            0    Successful completion

89551            >0   An error occurred.

89552 **CONSEQUENCES OF ERRORS**

89553            The *expand* utility shall terminate with an error message and non-zero exit status upon

89554            encountering difficulties accessing one of the *file* operands.

89555 **APPLICATION USAGE**

89556            None.

89557 **EXAMPLES**

89558            None.

89559 **RATIONALE**

89560            The *expand* utility is useful for preprocessing text files (before sorting, looking at specific

89561            columns, and so on) that contain <tab> characters.

89562            See XBD [Section 3.103](#) (on page 50).

89563            The *tablist* option-argument consists of integers in ascending order. Utility Syntax Guideline 8

89564            mandates that *expand* shall accept the integers (within the single argument) separated using

89565            either <comma> or <blank> characters.

89566            Earlier versions of this standard allowed the following form in the SYNOPSIS:

89567            `expand [-tabstop] [-tab1,tab2,...,tabn] [file ...]`

89568            This form is no longer specified by POSIX.1-2008 but may be present in some implementations.

89569 **FUTURE DIRECTIONS**

89570            None.

89571 **SEE ALSO**89572 *tabs*, *unexpand*89573 XBD [Section 3.103](#) (on page 50), [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 215)89574 **CHANGE HISTORY**

89575 First released in Issue 4.

89576 **Issue 6**

89577 This utility is marked as part of the User Portability Utilities option.

89578 The APPLICATION USAGE section is added.

89579 The obsolescent SYNOPSIS is removed.

89580 The *LC\_CTYPE* environment variable description is updated to align with the IEEE P1003.2b draft standard.

89582 The normative text is reworded to avoid use of the term “must” for application requirements.

89583 **Issue 7**

89584 Austin Group Interpretation 1003.1-2001 #027 is applied.

89585 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

89586 The *expand* utility is moved from the User Portability Utilities option to the Base. User Portability Utilities is now an option for interactive utilities.

89588 **NAME**89589        *expr* — evaluate arguments as an expression89590 **SYNOPSIS**89591        *expr operand...*89592 **DESCRIPTION**89593        The *expr* utility shall evaluate an expression and write the result to standard output.89594 **OPTIONS**

89595        None.

89596 **OPERANDS**89597        The single expression evaluated by *expr* shall be formed from the *operand* operands, as described in the EXTENDED DESCRIPTION section. The application shall ensure that each of the expression operator symbols:

89600        (   )   |   &amp;   =   &gt;   &gt;=   &lt;   &lt;=   !=   +   -   \*   /   %   :

89601        and the symbols *integer* and *string* in the table are provided as separate arguments to *expr*.89602 **STDIN**

89603        Not used.

89604 **INPUT FILES**

89605        None.

89606 **ENVIRONMENT VARIABLES**89607        The following environment variables shall affect the execution of *expr*:89608        *LANG*        Provide a default value for the internationalization variables that are unset or null. (See XBD Section 8.2 (on page 174) for the precedence of internationalization variables used to determine the values of locale categories.)89611        *LC\_ALL*        If set to a non-empty string value, override the values of all the other internationalization variables.89613        *LC\_COLLATE*

89614                    Determine the locale for the behavior of ranges, equivalence classes, and multi-character collating elements within regular expressions and by the string comparison operators.

89617        *LC\_CTYPE*    Determine the locale for the interpretation of sequences of bytes of text data as characters (for example, single-byte as opposed to multi-byte characters in arguments) and the behavior of character classes within regular expressions.89620        *LC\_MESSAGES*

89621                    Determine the locale that should be used to affect the format and contents of diagnostic messages written to standard error.

89623        *XSI*        *NLSPATH*    Determine the location of message catalogs for the processing of *LC\_MESSAGES*.89624 **ASYNCHRONOUS EVENTS**

89625        Default.

89626 **STDOUT**89627        The *expr* utility shall evaluate the expression and write the result, followed by a <newline>, to standard output.



89629 **STDERR**

89630 The standard error shall be used only for diagnostic messages.

89631 **OUTPUT FILES**

89632 None.

89633 **EXTENDED DESCRIPTION**

89634 The formation of the expression to be evaluated is shown in the following table. The symbols  
 89635 *expr*, *expr1*, and *expr2* represent expressions formed from *integer* and *string* symbols and the  
 89636 expression operator symbols (all separate arguments) by recursive application of the constructs  
 89637 described in the table. The expressions are listed in order of increasing precedence, with equal-  
 89638 precedence operators grouped between horizontal lines. All of the operators shall be left-  
 89639 associative.

| Expression                                                                                                                                                                                | Description                                                                                                                                                                                                                                                                                                                                                                                                |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>expr1</i>   <i>expr2</i>                                                                                                                                                               | Returns the evaluation of <i>expr1</i> if it is neither null nor zero; otherwise, returns the evaluation of <i>expr2</i> if it is not null; otherwise, zero.                                                                                                                                                                                                                                               |
| <i>expr1</i> & <i>expr2</i>                                                                                                                                                               | Returns the evaluation of <i>expr1</i> if neither expression evaluates to null or zero; otherwise, returns zero.                                                                                                                                                                                                                                                                                           |
| <i>expr1</i> = <i>expr2</i><br><i>expr1</i> > <i>expr2</i><br><i>expr1</i> >= <i>expr2</i><br><i>expr1</i> < <i>expr2</i><br><i>expr1</i> <= <i>expr2</i><br><i>expr1</i> != <i>expr2</i> | Returns the result of a decimal integer comparison if both arguments are integers; otherwise, returns the result of a string comparison using the locale-specific collation sequence. The result of each comparison is 1 if the specified relationship is true, or 0 if the relationship is false.<br>Equal.<br>Greater than.<br>Greater than or equal.<br>Less than.<br>Less than or equal.<br>Not equal. |
| <i>expr1</i> + <i>expr2</i><br><i>expr1</i> – <i>expr2</i>                                                                                                                                | Addition of decimal integer-valued arguments.<br>Subtraction of decimal integer-valued arguments.                                                                                                                                                                                                                                                                                                          |
| <i>expr1</i> * <i>expr2</i><br><i>expr1</i> / <i>expr2</i><br><i>expr1</i> % <i>expr2</i>                                                                                                 | Multiplication of decimal integer-valued arguments.<br>Integer division of decimal integer-valued arguments, producing an integer result.<br>Remainder of integer division of decimal integer-valued arguments.                                                                                                                                                                                            |
| <i>expr1</i> : <i>expr2</i>                                                                                                                                                               | Matching expression; see below.                                                                                                                                                                                                                                                                                                                                                                            |
| ( <i>expr</i> )                                                                                                                                                                           | Grouping symbols. Any expression can be placed within parentheses. Parentheses can be nested to a depth of {EXPR_NEST_MAX}.                                                                                                                                                                                                                                                                                |
| <i>integer</i><br><i>string</i>                                                                                                                                                           | An argument consisting only of an (optional) unary minus followed by digits.<br>A string argument; see below.                                                                                                                                                                                                                                                                                              |

### Matching Expression

The `' : '` matching operator shall compare the string resulting from the evaluation of *expr1* with the regular expression pattern resulting from the evaluation of *expr2*. Regular expression syntax shall be that defined in XBD [Section 9.3](#) (on page 183), except that all patterns are anchored to the beginning of the string (that is, only sequences starting at the first character of a string are matched by the regular expression) and, therefore, it is unspecified whether `' ^ '` is a special character in that context. Usually, the matching operator shall return a string representing the number of characters matched (`' 0 '` on failure). Alternatively, if the pattern contains at least one regular expression subexpression `"[ \ ( . . . \ ) ]"`, the string matched by the back-reference expression `" \ 1 "` shall be returned. If the back-reference expression `" \ 1 "` does not match, then the null string shall be returned.

### String Operand

A string argument is an argument that cannot be identified as an *integer* argument or as one of the expression operator symbols shown in the OPERANDS section.

The use of string arguments **length**, **substr**, **index**, or **match** produces unspecified results.

### EXIT STATUS

The following exit values shall be returned:

- 0 The *expression* evaluates to neither null nor zero.
- 1 The *expression* evaluates to null or zero.
- 2 Invalid *expression*.
- >2 An error occurred.

### CONSEQUENCES OF ERRORS

Default.

### APPLICATION USAGE

After argument processing by the shell, *expr* is not required to be able to tell the difference between an operator and an operand except by the value. If `" $a "` is `' = '`, the command:

```
expr $a = '='
```

looks like:

```
expr = = =
```

as the arguments are passed to *expr* (and they all may be taken as the `' = '` operator). The following works reliably:

```
expr X$a = X=
```

Also note that this volume of POSIX.1-2008 permits implementations to extend utilities. The *expr* utility permits the integer arguments to be preceded with a unary minus. This means that an integer argument could look like an option. Therefore, the conforming application must employ the `"--"` construct of Guideline 10 of XBD [Section 12.2](#) (on page 215) to protect its operands if there is any chance the first operand might be a negative integer (or any string with a leading minus).

**EXAMPLES**

The *expr* utility has a rather difficult syntax:

- Many of the operators are also shell control operators or reserved words, so they have to be escaped on the command line.
- Each part of the expression is composed of separate arguments, so liberal usage of <blank> characters is required. For example:

| Invalid                 | Valid                        |
|-------------------------|------------------------------|
| <i>expr</i> 1+2         | <i>expr</i> 1 + 2            |
| <i>expr</i> "1 + 2"     | <i>expr</i> 1 + 2            |
| <i>expr</i> 1 + (2 * 3) | <i>expr</i> 1 + \( 2 \* 3 \) |

In many cases, the arithmetic and string features provided as part of the shell command language are easier to use than their equivalents in *expr*. Newly written scripts should avoid *expr* in favor of the new features within the shell; see [Section 2.5](#) (on page 2323) and [Section 2.6.4](#) (on page 2332).

The following command:

```
a=$(expr $a + 1)
```

adds 1 to the variable *a*.

The following command, for "*\$a*" equal to either */usr/abc/file* or just *file*:

```
expr $a : '.*\/(.*\)' \| $a
```

returns the last segment of a pathname (that is, *file*). Applications should avoid the character *'/'* used alone as an argument; *expr* may interpret it as the division operator.

The following command:

```
expr "//$a" : '.*\/(.*\)'
```

is a better representation of the previous example. The addition of the *"/"* characters eliminates any ambiguity about the division operator and simplifies the whole expression. Also note that pathnames may contain characters contained in the *IFS* variable and should be quoted to avoid having "*\$a*" expand into multiple arguments.

The following command:

```
expr "$VAR" : '.*'
```

returns the number of characters in *VAR*.

**RATIONALE**

In an early proposal, EREs were used in the matching expression syntax. This was changed to BREs to avoid breaking historical applications.

The use of a leading <circumflex> in the BRE is unspecified because many historical implementations have treated it as a special character, despite their system documentation. For example:

```
expr foo : ^foo expr ^foo : ^foo
```

return 3 and 0, respectively, on those systems; their documentation would imply the reverse. Thus, the anchoring condition is left unspecified to avoid breaking historical scripts relying on this undocumented feature.

89749 **FUTURE DIRECTIONS**

89750 None.

89751 **SEE ALSO**89752 [Section 2.5](#) (on page 2323), [Section 2.6.4](#) (on page 2332)89753 XBD [Chapter 8](#) (on page 173), [Section 9.3](#) (on page 183), [Section 12.2](#) (on page 215)89754 **CHANGE HISTORY**

89755 First released in Issue 2.

89756 **Issue 5**

89757 The FUTURE DIRECTIONS section is added.

89758 **Issue 6**89759 The *expr* utility is aligned with the IEEE P1003.2b draft standard, to include resolution of IEEE  
89760 PASC Interpretation 1003.2 #104.

89761 The normative text is reworded to avoid use of the term “must” for application requirements.

89762 **Issue 7**

89763 Austin Group Interpretation 1003.1-2001 #036 is applied, clarifying the behavior for BREs.

89764 The SYNOPSIS and OPERANDS sections are revised to explicitly state that the name of each of  
89765 the operands is *operand*.

**89766 NAME**

89767 false — return false value

**89768 SYNOPSIS**

89769 false

**89770 DESCRIPTION**

89771 The *false* utility shall return with a non-zero exit code.

**89772 OPTIONS**

89773 None.

**89774 OPERANDS**

89775 None.

**89776 STDIN**

89777 Not used.

**89778 INPUT FILES**

89779 None.

**89780 ENVIRONMENT VARIABLES**

89781 None.

**89782 ASYNCHRONOUS EVENTS**

89783 Default.

**89784 STDOUT**

89785 Not used.

**89786 STDERR**

89787 Not used.

**89788 OUTPUT FILES**

89789 None.

**89790 EXTENDED DESCRIPTION**

89791 None.

**89792 EXIT STATUS**

89793 The *false* utility shall always exit with a value other than zero.

**89794 CONSEQUENCES OF ERRORS**

89795 Default.

**89796 APPLICATION USAGE**

89797 None.

**89798 EXAMPLES**

89799 None.

**89800 RATIONALE**

89801 None.

**89802 FUTURE DIRECTIONS**

89803 None.

**89804 SEE ALSO**

89805 *true*

**CHANGE HISTORY**

89806  
89807 First released in Issue 2.

**Issue 6**

89808  
89809 IEEE Std 1003.1-2001/Cor 1-2002, item XCU/TC1/D6/24 is applied, changing the STDERR  
89810 section from “None.” to “Not used.” for alignment with [Section 1.4](#) (on page 2310).

89811 **NAME**89812 `fc` — process the command history list89813 **SYNOPSIS**

```
89814 UP fc [-r] [-e editor] [first [last]]
89815 fc -l [-nr] [first [last]]
89816 fc -s [old=new] [first]
```

89817 **DESCRIPTION**

89818 The `fc` utility shall list, or shall edit and re-execute, commands previously entered to an  
 89819 interactive `sh`.

89820 The command history list shall reference commands by number. The first number in the list is  
 89821 selected arbitrarily. The relationship of a number to its command shall not change except when  
 89822 the user logs in and no other process is accessing the list, at which time the system may reset the  
 89823 numbering to start the oldest retained command at another number (usually 1). When the  
 89824 number reaches an implementation-defined upper limit, which shall be no smaller than the  
 89825 value in `HISTSIZE` or 32767 (whichever is greater), the shell may wrap the numbers, starting the  
 89826 next command with a lower number (usually 1). However, despite this optional wrapping of  
 89827 numbers, `fc` shall maintain the time-ordering sequence of the commands. For example, if four  
 89828 commands in sequence are given the numbers 32766, 32767, 1 (wrapped), and 2 as they are  
 89829 executed, command 32767 is considered the command previous to 1, even though its number is  
 89830 higher.

89831 When commands are edited (when the `-l` option is not specified), the resulting lines shall be  
 89832 entered at the end of the history list and then re-executed by `sh`. The `fc` command that caused the  
 89833 editing shall not be entered into the history list. If the editor returns a non-zero exit status, this  
 89834 shall suppress the entry into the history list and the command re-execution. Any command line  
 89835 variable assignments or redirection operators used with `fc` shall affect both the `fc` command itself  
 89836 as well as the command that results; for example:

```
89837 fc -s -- -l 2>/dev/null
```

89838 reinvokes the previous command, suppressing standard error for both `fc` and the previous  
 89839 command.

89840 **OPTIONS**

89841 The `fc` utility shall conform to XBD [Section 12.2](#) (on page 215).

89842 The following options shall be supported:

- 89843 **-e editor** Use the editor named by *editor* to edit the commands. The *editor* string is a utility  
 89844 name, subject to search via the `PATH` variable (see XBD [Chapter 8](#), on page 173).  
 89845 The value in the `FCEDIT` variable shall be used as a default when `-e` is not  
 89846 specified. If `FCEDIT` is null or unset, `ed` shall be used as the editor.
- 89847 **-l** (The letter ell.) List the commands rather than invoking an editor on them. The  
 89848 commands shall be written in the sequence indicated by the *first* and *last* operands,  
 89849 as affected by `-r`, with each command preceded by the command number.
- 89850 **-n** Suppress command numbers when listing with `-l`.
- 89851 **-r** Reverse the order of the commands listed (with `-l`) or edited (with neither `-l` nor  
 89852 `-s`).

89853            **-s**            Re-execute the command without invoking an editor.

89854    **OPERANDS**

89855            The following operands shall be supported:

89856            *first, last*        Select the commands to list or edit. The number of previous commands that can be  
89857                                accessed shall be determined by the value of the *HISTSIZE* variable. The value of  
89858                                *first* or *last* or both shall be one of the following:

89859            **[+]number**        A positive number representing a command number; command  
89860                                numbers can be displayed with the **-l** option.

89861            **-number**            A negative decimal number representing the command that was  
89862                                executed *number* of commands previously. For example, **-1** is the  
89863                                immediately previous command.

89864            *string*             A string indicating the most recently entered command that begins  
89865                                with that string. If the *old=new* operand is not also specified with **-s**,  
89866                                the string form of the *first* operand cannot contain an embedded  
89867                                `<equals-sign>`.

89868            When the synopsis form with **-s** is used:

- 89869                        • If *first* is omitted, the previous command shall be used.

89870            For the synopsis forms without **-s**:

- 89871                        • If *last* is omitted, *last* shall default to the previous command when **-l** is  
89872                                specified; otherwise, it shall default to *first*.

- 89873                        • If *first* and *last* are both omitted, the previous 16 commands shall be listed or  
89874                                the previous single command shall be edited (based on the **-l** option).

- 89875                        • If *first* and *last* are both present, all of the commands from *first* to *last* shall be  
89876                                edited (without **-l**) or listed (with **-l**). Editing multiple commands shall be  
89877                                accomplished by presenting to the editor all of the commands at one time,  
89878                                each command starting on a new line. If *first* represents a newer command  
89879                                than *last*, the commands shall be listed or edited in reverse sequence,  
89880                                equivalent to using **-r**. For example, the following commands on the first  
89881                                line are equivalent to the corresponding commands on the second:

89882                        `fc -r 10 20        fc        30 40`  
89883                        `fc        20 10        fc -r 40 30`

- 89884                        • When a range of commands is used, it shall not be an error to specify *first* or  
89885                                *last* values that are not in the history list; *fc* shall substitute the value  
89886                                representing the oldest or newest command in the list, as appropriate. For  
89887                                example, if there are only ten commands in the history list, numbered 1 to 10:

89888                        `fc -l`  
89889                        `fc 1 99`

89890                        shall list and edit, respectively, all ten commands.

89891            *old=new*            Replace the first occurrence of string *old* in the commands to be re-executed by the  
89892                                string *new*.



89893 **STDIN**

89894 Not used.

89895 **INPUT FILES**

89896 None.

89897 **ENVIRONMENT VARIABLES**89898 The following environment variables shall affect the execution of *fc*:

89899 *FCEDIT* This variable, when expanded by the shell, shall determine the default value for  
 89900 the *-e editor* option's *editor* option-argument. If *FCEDIT* is null or unset, *ed* shall be  
 89901 used as the editor.

89902 *HISTFILE* Determine a pathname naming a command history file. If the *HISTFILE* variable is  
 89903 not set, the shell may attempt to access or create a file *.sh\_history* in the directory  
 89904 referred to by the *HOME* environment variable. If the shell cannot obtain both read  
 89905 and write access to, or create, the history file, it shall use an unspecified  
 89906 mechanism that allows the history to operate properly. (References to history "file"  
 89907 in this section shall be understood to mean this unspecified mechanism in such  
 89908 cases.) An implementation may choose to access this variable only when  
 89909 initializing the history file; this initialization shall occur when *fc* or *sh* first attempt  
 89910 to retrieve entries from, or add entries to, the file, as the result of commands issued  
 89911 by the user, the file named by the *ENV* variable, or implementation-defined system  
 89912 start-up files. In some historical shells, the history file is initialized just after the  
 89913 *ENV* file has been processed. Therefore, it is implementation-defined whether  
 89914 changes made to *HISTFILE* after the history file has been initialized are effective.  
 89915 Implementations may choose to disable the history list mechanism for users with  
 89916 appropriate privileges who do not set *HISTFILE*; the specific circumstances under  
 89917 which this occurs are implementation-defined. If more than one instance of the  
 89918 shell is using the same history file, it is unspecified how updates to the history file  
 89919 from those shells interact. As entries are deleted from the history file, they shall be  
 89920 deleted oldest first. It is unspecified when history file entries are physically  
 89921 removed from the history file.

89922 *HISTSIZE* Determine a decimal number representing the limit to the number of previous  
 89923 commands that are accessible. If this variable is unset, an unspecified default  
 89924 greater than or equal to 128 shall be used. The maximum number of commands in  
 89925 the history list is unspecified, but shall be at least 128. An implementation may  
 89926 choose to access this variable only when initializing the history file, as described  
 89927 under *HISTFILE*. Therefore, it is unspecified whether changes made to *HISTSIZE*  
 89928 after the history file has been initialized are effective.

89929 *LANG* Provide a default value for the internationalization variables that are unset or null.  
 89930 (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 89931 variables used to determine the values of locale categories.)

89932 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
 89933 internationalization variables.

89934 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
 89935 characters (for example, single-byte as opposed to multi-byte characters in  
 89936 arguments and input files).

89937 *LC\_MESSAGES*

89938 Determine the locale that should be used to affect the format and contents of  
 89939 diagnostic messages written to standard error.

89940 XSI **NLSPATH** Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

89941 **ASYNCHRONOUS EVENTS**

89942 Default.

89943 **STDOUT**

89944 When the **-l** option is used to list commands, the format of each command in the list shall be as

89945 follows:

89946 "%d\t%s\n", *<line number>*, *<command>*

89947 If both the **-l** and **-n** options are specified, the format of each command shall be:

89948 "\t%s\n", *<command>*

89949 If the *<command>* consists of more than one line, the lines after the first shall be displayed as:

89950 "\t%s\n", *<continued-command>*

89951 **STDERR**

89952 The standard error shall be used only for diagnostic messages.

89953 **OUTPUT FILES**

89954 None.

89955 **EXTENDED DESCRIPTION**

89956 None.

89957 **EXIT STATUS**

89958 The following exit values shall be returned:

89959 0 Successful completion of the listing.

89960 >0 An error occurred.

89961 Otherwise, the exit status shall be that of the commands executed by *fc*.

89962 **CONSEQUENCES OF ERRORS**

89963 Default.

89964 **APPLICATION USAGE**

89965 Since editors sometimes use file descriptors as integral parts of their editing, redirecting their file

89966 descriptors as part of the *fc* command can produce unexpected results. For example, if *vi* is the

89967 *FCEDIT* editor, the command:

89968 *fc -s | more*

89969 does not work correctly on many systems.

89970 Users on windowing systems may want to have separate history files for each window by

89971 setting *HISTFILE* as follows:

89972 *HISTFILE=\$HOME/.sh\_hist\$\$*

89973 **EXAMPLES**

89974 None.

89975 **RATIONALE**

89976 This utility is based on the *fc* built-in of the KornShell.

89977 An early proposal specified the **-e** option as [**-e** *editor* [*old= new* ]], which is not historical

89978 practice. Historical practice in *fc* of either [**-e** *editor*] or [**-e** - [*old= new* ]] is acceptable, but not

89979 both together. To clarify this, a new option **-s** was introduced replacing the [**-e** -]. This resolves

the conflict and makes *fc* conform to the Utility Syntax Guidelines.

**HISTFILE** Some implementations of the KornShell check for the superuser and do not create a history file unless *HISTFILE* is set. This is done primarily to avoid creating unlinked files in the root file system when logging in during single-user mode. *HISTFILE* must be set for the superuser to have history.

**HISTSIZE** Needed to limit the size of history files. It is the intent of the standard developers that when two shells share the same history file, commands that are entered in one shell shall be accessible by the other shell. Because of the difficulties of synchronization over a network, the exact nature of the interaction is unspecified.

The initialization process for the history file can be dependent on the system start-up files, in that they may contain commands that effectively preempt the settings the user has for *HISTFILE* and *HISTSIZE*. For example, function definition commands are recorded in the history file. If the system administrator includes function definitions in some system start-up file called before the *ENV* file, the history file is initialized before the user can influence its characteristics. In some historical shells, the history file is initialized just after the *ENV* file has been processed. Because of these situations, the text requires the initialization process to be implementation-defined.

Consideration was given to omitting the *fc* utility in favor of the command line editing feature in *sh*. For example, in *vi* editing mode, typing "<ESC> v" is equivalent to:

```
EDITOR=vi fc
```

However, the *fc* utility allows the user the flexibility to edit multiple commands simultaneously (such as *fc 10 20*) and to use editors other than those supported by *sh* for command line editing.

In the KornShell, the alias *r* ("re-do") is preset to *fc -e* – (equivalent to the POSIX *fc -s*). This is probably an easier command name to remember than *fc* ("fix command"), but it does not meet the Utility Syntax Guidelines. Renaming *fc* to *hist* or *redo* was considered, but since this description closely matches historical KornShell practice already, such a renaming was seen as gratuitous. Users are free to create aliases whenever odd historical names such as *fc*, *awk*, *cat*, *grep*, or *yacc* are standardized by POSIX.

Command numbers have no ordering effects; they are like serial numbers. The *-r* option and *-number* operand address the sequence of command execution, regardless of serial numbers. So, for example, if the command number wrapped back to 1 at some arbitrary point, there would be no ambiguity associated with traversing the wrap point. For example, if the command history were:

```
32766: echo 1
32767: echo 2
1: echo 3
```

the number *-2* refers to command 32767 because it is the second previous command, regardless of serial number.

## FUTURE DIRECTIONS

None.

## SEE ALSO

*sh*

XBD [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 215)

**90022 CHANGE HISTORY**

90023 First released in Issue 4.

**90024 Issue 5**

90025 The FUTURE DIRECTIONS section is added.

**90026 Issue 6**

90027 This utility is marked as part of the User Portability Utilities option.

90028 In the ENVIRONMENT VARIABLES section, the text “user’s home directory” is updated to  
90029 “directory referred to by the *HOME* environment variable”.

**90030 Issue 7**

90031 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

90032 **NAME**90033 `fg` — run jobs in the foreground90034 **SYNOPSIS**90035 UP `fg [job_id]`90036 **DESCRIPTION**90037 If job control is enabled (see the description of `set -m`), the `fg` utility shall move a background job  
90038 from the current environment (see [Section 2.12](#), on page 2353) into the foreground.90039 Using `fg` to place a job into the foreground shall remove its process ID from the list of those  
90040 “known in the current shell execution environment”; see [Section 2.9.3.1](#) (on page 2342).90041 **OPTIONS**

90042 None.

90043 **OPERANDS**

90044 The following operand shall be supported:

90045 *job\_id* Specify the job to be run as a foreground job. If no *job\_id* operand is given, the  
90046 *job\_id* for the job that was most recently suspended, placed in the background, or  
90047 run as a background job shall be used. The format of *job\_id* is described in XBD  
90048 [Section 3.204](#) (on page 65).90049 **STDIN**

90050 Not used.

90051 **INPUT FILES**

90052 None.

90053 **ENVIRONMENT VARIABLES**90054 The following environment variables shall affect the execution of `fg`:90055 *LANG* Provide a default value for the internationalization variables that are unset or null.  
90056 (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
90057 variables used to determine the values of locale categories.)90058 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
90059 internationalization variables.90060 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
90061 characters (for example, single-byte as opposed to multi-byte characters in  
90062 arguments).90063 *LC\_MESSAGES*90064 Determine the locale that should be used to affect the format and contents of  
90065 diagnostic messages written to standard error.90066 XSI *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.90067 **ASYNCHRONOUS EVENTS**

90068 Default.

90069 **STDOUT**90070 The `fg` utility shall write the command line of the job to standard output in the following format:90071 `"%s\n", <command>`

**90072 STDERR**

90073 The standard error shall be used only for diagnostic messages.

**90074 OUTPUT FILES**

90075 None.

**90076 EXTENDED DESCRIPTION**

90077 None.

**90078 EXIT STATUS**

90079 The following exit values shall be returned:

90080 0 Successful completion.

90081 >0 An error occurred.

**90082 CONSEQUENCES OF ERRORS**

90083 If job control is disabled, the *fg* utility shall exit with an error and no job shall be placed in the  
90084 foreground.

**90085 APPLICATION USAGE**

90086 The *fg* utility does not work as expected when it is operating in its own utility execution  
90087 environment because that environment has no applicable jobs to manipulate. See the  
90088 APPLICATION USAGE section for *bg*. For this reason, *fg* is generally implemented as a shell  
90089 regular built-in.

**90090 EXAMPLES**

90091 None.

**90092 RATIONALE**

90093 The extensions to the shell specified in this volume of POSIX.1-2008 have mostly been based on  
90094 features provided by the KornShell. The job control features provided by *bg*, *fg*, and *jobs* are also  
90095 based on the KornShell. The standard developers examined the characteristics of the C shell  
90096 versions of these utilities and found that differences exist. Despite widespread use of the C shell,  
90097 the KornShell versions were selected for this volume of POSIX.1-2008 to maintain a degree of  
90098 uniformity with the rest of the KornShell features selected (such as the very popular command  
90099 line editing features).

**90100 FUTURE DIRECTIONS**

90101 None.

**90102 SEE ALSO**

90103 [Section 2.9.3.1](#) (on page 2342), [Section 2.12](#) (on page 2353), *bg*, *kill*, *jobs*, *wait*

90104 XBD [Section 3.204](#) (on page 65), [Chapter 8](#) (on page 173)

**90105 CHANGE HISTORY**

90106 First released in Issue 4.

**90107 Issue 6**

90108 This utility is marked as part of the User Portability Utilities option.

90109 The APPLICATION USAGE section is added.

90110 The JC marking is removed from the SYNOPSIS since job control is mandatory in this version.

**NAME**

*file* — determine file type

**SYNOPSIS**

*file* [-dh] [-M *file*] [-m *file*] *file*...

*file* -i [-h] *file*...

**DESCRIPTION**

The *file* utility shall perform a series of tests in sequence on each specified *file* in an attempt to classify it:

1. If *file* does not exist, cannot be read, or its file status could not be determined, the output shall indicate that the file was processed, but that its type could not be determined.
2. If the file is not a regular file, its file type shall be identified. The file types directory, FIFO, socket, block special, and character special shall be identified as such. Other implementation-defined file types may also be identified. If *file* is a symbolic link, by default the link shall be resolved and *file* shall test the type of file referenced by the symbolic link. (See the **-h** and **-i** options below.)
3. If the length of *file* is zero, it shall be identified as an empty file.
4. The *file* utility shall examine an initial segment of *file* and shall make a guess at identifying its contents based on position-sensitive tests. (The answer is not guaranteed to be correct; see the **-d**, **-M**, and **-m** options below.)
5. The *file* utility shall examine *file* and make a guess at identifying its contents based on context-sensitive default system tests. (The answer is not guaranteed to be correct.)
6. The file shall be identified as a data file.

If *file* does not exist, cannot be read, or its file status could not be determined, the output shall indicate that the file was processed, but that its type could not be determined.

If *file* is a symbolic link, by default the link shall be resolved and *file* shall test the type of file referenced by the symbolic link.

**OPTIONS**

The *file* utility shall conform to XBD [Section 12.2](#) (on page 215), except that the order of the **-m**, **-d**, and **-M** options shall be significant.

The following options shall be supported by the implementation:

- |                       |                                                                                                                                                                                                                                                                                                              |
|-----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>-d</b>             | Apply any position-sensitive default system tests and context-sensitive default system tests to the file. This is the default if no <b>-M</b> or <b>-m</b> option is specified.                                                                                                                              |
| <b>-h</b>             | When a symbolic link is encountered, identify the file as a symbolic link. If <b>-h</b> is not specified and <i>file</i> is a symbolic link that refers to a nonexistent file, <i>file</i> shall identify the file as a symbolic link, as if <b>-h</b> had been specified.                                   |
| <b>-i</b>             | If a file is a regular file, do not attempt to classify the type of the file further, but identify the file as specified in the STDOUT section.                                                                                                                                                              |
| <b>-M <i>file</i></b> | Specify the name of a file containing position-sensitive tests that shall be applied to a file in order to classify it (see the EXTENDED DESCRIPTION). No position-sensitive default system tests nor context-sensitive default system tests shall be applied unless the <b>-d</b> option is also specified. |

90152        **-m file**        Specify the name of a file containing position-sensitive tests that shall be applied to  
 90153                                a file in order to classify it (see the EXTENDED DESCRIPTION).

90154        If the **-m** option is specified without specifying the **-d** option or the **-M** option, position-  
 90155        sensitive default system tests shall be applied after the position-sensitive tests specified by the  
 90156        **-m** option. If the **-M** option is specified with the **-d** option, the **-m** option, or both, or the **-m**  
 90157        option is specified with the **-d** option, the concatenation of the position-sensitive tests specified  
 90158        by these options shall be applied in the order specified by the appearance of these options. If a  
 90159        **-M** or **-m file** option-argument is **-**, the results are unspecified.

## 90160 OPERANDS

90161        The following operand shall be supported:

90162        *file*                A pathname of a file to be tested.

## 90163 STDIN

90164        The standard input shall be used if a *file* operand is **'-'** and the implementation treats the **'-'**  
 90165        as meaning standard input. Otherwise, the standard input shall not be used.

## 90166 INPUT FILES

90167        The *file* can be any file type.

## 90168 ENVIRONMENT VARIABLES

90169        The following environment variables shall affect the execution of *file*:

90170        *LANG*                Provide a default value for the internationalization variables that are unset or null.  
 90171                                (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 90172                                variables used to determine the values of locale categories.)

90173        *LC\_ALL*                If set to a non-empty string value, override the values of all the other  
 90174                                internationalization variables.

90175        *LC\_CTYPE*            Determine the locale for the interpretation of sequences of bytes of text data as  
 90176                                characters (for example, single-byte as opposed to multi-byte characters in  
 90177                                arguments and input files).

90178        *LC\_MESSAGES*

90179                                Determine the locale that should be used to affect the format and contents of  
 90180                                diagnostic messages written to standard error and informative messages written to  
 90181                                standard output.

90182        *XSI NLSPATH*        Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

## 90183 ASYNCHRONOUS EVENTS

90184        Default.

## 90185 STDOUT

90186        In the POSIX locale, the following format shall be used to identify each operand, *file* specified:

90187        "%s: %s\n", <file>, <type>

90188        The values for <type> are unspecified, except that in the POSIX locale, if *file* is identified as one  
 90189        of the types listed in the following table, <type> shall contain (but is not limited to) the  
 90190        corresponding string, unless the file is identified by a position-sensitive test specified by a **-M** or  
 90191        **-m** option. Each <space> shown in the strings shall be exactly one <space>.



90192

**Table 4-9** File Utility Output Strings

|       | <b>If <i>file</i> is:</b>                                    | <b>&lt;type&gt; shall contain the string:</b> | <b>Notes</b> |
|-------|--------------------------------------------------------------|-----------------------------------------------|--------------|
| 90193 | Nonexistent                                                  | cannot open                                   |              |
| 90194 | Block special                                                | block special                                 | 1            |
| 90195 | Character special                                            | character special                             | 1            |
| 90196 | Directory                                                    | directory                                     | 1            |
| 90197 | FIFO                                                         | fifo                                          | 1            |
| 90198 | Socket                                                       | socket                                        | 1            |
| 90199 | Symbolic link                                                | symbolic link to                              | 1            |
| 90200 | Regular file                                                 | regular file                                  | 1,2          |
| 90201 | Empty regular file                                           | empty                                         | 3            |
| 90202 | Regular file that cannot be read                             | cannot open                                   | 3            |
| 90203 | Executable binary                                            | executable                                    | 3,4,6        |
| 90204 | <i>ar</i> archive library (see <i>ar</i> )                   | archive                                       | 3,4,6        |
| 90205 | Extended <i>cpio</i> format (see <i>pax</i> )                | cpio archive                                  | 3,4,6        |
| 90206 | Extended <i>tar</i> format (see <b>ustar</b> in <i>pax</i> ) | tar archive                                   | 3,4,6        |
| 90207 | Shell script                                                 | commands text                                 | 3,5,6        |
| 90208 | C-language source                                            | c program text                                | 3,5,6        |
| 90209 | FORTRAN source                                               | fortran program text                          | 3,5,6        |
| 90210 | Regular file whose type cannot be determined                 | data                                          | 3            |

90212 **Notes:**

- 90213 1. This is a file type test.
- 90214 2. This test is applied only if the **-i** option is specified.
- 90215 3. This test is applied only if the **-i** option is not specified.
- 90216 4. This is a position-sensitive default system test.
- 90217 5. This is a context-sensitive default system test.
- 90218 6. Position-sensitive default system tests and context-sensitive default system tests are not
- 90219 applied if the **-M** option is specified unless the **-d** option is also specified.

90220 In the POSIX locale, if *file* is identified as a symbolic link (see the **-h** option), the following

90221 alternative output format shall be used:

90222 "%s: %s %s\n", <file>, <type>, <contents of link>"

90223 If the file named by the *file* operand does not exist, cannot be read, or the type of the file named

90224 by the *file* operand cannot be determined, this shall not be considered an error that affects the

90225 exit status.

90226 **STDERR**

90227 The standard error shall be used only for diagnostic messages.

90228 **OUTPUT FILES**

90229 None.

## EXTENDED DESCRIPTION

A file specified as an option-argument to the **-m** or **-M** options shall contain one position-sensitive test per line, which shall be applied to the file. If the test succeeds, the message field of the line shall be printed and no further tests shall be applied, with the exception that tests on immediately following lines beginning with a single '**>**' character shall be applied.

Each line shall be composed of the following four <tab>-separated fields. (Implementations may allow any combination of one or more white-space characters other than <newline> to act as field separators.)

*offset*      An unsigned number (optionally preceded by a single '**>**' character) specifying the *offset*, in bytes, of the value in the file that is to be compared against the *value* field of the line. If the file is shorter than the specified offset, the test shall fail.

If the *offset* begins with the character '**>**', the test contained in the line shall not be applied to the file unless the test on the last line for which the *offset* did not begin with a '**>**' was successful. By default, the *offset* shall be interpreted as an unsigned decimal number. With a leading 0x or 0X, the *offset* shall be interpreted as a hexadecimal number; otherwise, with a leading 0, the *offset* shall be interpreted as an octal number.

*type*      The type of the value in the file to be tested. The type shall consist of the type specification characters **d**, **s**, and **u**, specifying signed decimal, string, and unsigned decimal, respectively.

The *type* string shall be interpreted as the bytes from the file starting at the specified *offset* and including the same number of bytes specified by the *value* field. If insufficient bytes remain in the file past the *offset* to match the *value* field, the test shall fail.

The type specification characters **d** and **u** can be followed by an optional unsigned decimal integer that specifies the number of bytes represented by the type. The type specification characters **d** and **u** can be followed by an optional **C**, **S**, **I**, or **L**, indicating that the value is of type **char**, **short**, **int**, or **long**, respectively.

The default number of bytes represented by the type specifiers **d**, **f**, and **u** shall correspond to their respective C-language types as follows. If the system claims conformance to the C-Language Development Utilities option, those specifiers shall correspond to the default sizes used in the *c99* utility. Otherwise, the default sizes shall be implementation-defined.

For the type specifier characters **d** and **u**, the default number of bytes shall correspond to the size of a basic integer type of the implementation. For these specifier characters, the implementation shall support values of the optional number of bytes to be converted corresponding to the number of bytes in the C-language types **char**, **short**, **int**, or **long**. These numbers can also be specified by an application as the characters **C**, **S**, **I**, and **L**, respectively. The byte order used when interpreting numeric values is implementation-defined, but shall correspond to the order in which a constant of the corresponding type is stored in memory on the system.

All type specifiers, except for **s**, can be followed by a mask specifier of the form **&number**. The mask value shall be AND'ed with the value of the input file before the comparison with the *value* field of the line is made. By default, the mask shall be interpreted as an unsigned decimal number. With a leading 0x or 0X, the mask shall be interpreted as an unsigned hexadecimal number; otherwise, with a leading

|       |                   |                                                                                                                  |
|-------|-------------------|------------------------------------------------------------------------------------------------------------------|
| 90277 |                   | 0, the mask shall be interpreted as an unsigned octal number.                                                    |
| 90278 |                   | The strings <b>byte</b> , <b>short</b> , <b>long</b> , and <b>string</b> shall also be supported as type fields, |
| 90279 |                   | being interpreted as dC, dS, dL, and s, respectively.                                                            |
| 90280 | <i>value</i>      | The <i>value</i> to be compared with the value from the file.                                                    |
| 90281 |                   | If the specifier from the type field is s or <b>string</b> , then interpret the value as a string.               |
| 90282 |                   | Otherwise, interpret it as a number. If the value is a string, then the test shall                               |
| 90283 |                   | succeed only when a string value exactly matches the bytes from the file.                                        |
| 90284 |                   | If the <i>value</i> is a string, it can contain the following sequences:                                         |
| 90285 | <i>\character</i> | The <backslash>-escape sequences as specified in XBD Table 5-1                                                   |
| 90286 |                   | (on page 121) ('\\', '\a', '\b', '\f', '\n', '\r', '\t',                                                         |
| 90287 |                   | '\v'). In addition, the escape sequence '\ ' (the <backslash>                                                    |
| 90288 |                   | character followed by a <space> character) shall be recognized to                                                |
| 90289 |                   | represent a <space> character. The results of using any other                                                    |
| 90290 |                   | character, other than an octal digit, following the <backslash>                                                  |
| 90291 |                   | are unspecified.                                                                                                 |
| 90292 | <i>\octal</i>     | Octal sequences that can be used to represent characters with                                                    |
| 90293 |                   | specific coded values. An octal sequence shall consist of a                                                      |
| 90294 |                   | <backslash> followed by the longest sequence of one, two, or                                                     |
| 90295 |                   | three octal-digit characters (01234567).                                                                         |
| 90296 |                   | By default, any value that is not a string shall be interpreted as a signed decimal                              |
| 90297 |                   | number. Any such value, with a leading 0x or 0X, shall be interpreted as an                                      |
| 90298 |                   | unsigned hexadecimal number; otherwise, with a leading zero, the value shall be                                  |
| 90299 |                   | interpreted as an unsigned octal number.                                                                         |
| 90300 |                   | If the value is not a string, it can be preceded by a character indicating the                                   |
| 90301 |                   | comparison to be performed. Permissible characters and the comparisons they                                      |
| 90302 |                   | specify are as follows:                                                                                          |
| 90303 | =                 | The test shall succeed if the value from the file equals the <i>value</i> field.                                 |
| 90304 | <                 | The test shall succeed if the value from the file is less than the <i>value</i> field.                           |
| 90305 | >                 | The test shall succeed if the value from the file is greater than the <i>value</i> field.                        |
| 90306 | &                 | The test shall succeed if all of the set bits in the <i>value</i> field are set in the value                     |
| 90307 |                   | from the file.                                                                                                   |
| 90308 | ^                 | The test shall succeed if at least one of the set bits in the <i>value</i> field is not set in                   |
| 90309 |                   | the value from the file.                                                                                         |
| 90310 | x                 | The test shall succeed if the file is large enough to contain a value of the type                                |
| 90311 |                   | specified starting at the offset specified.                                                                      |
| 90312 | <i>message</i>    | The <i>message</i> to be printed if the test succeeds. The <i>message</i> shall be interpreted                   |
| 90313 |                   | using the notation for the <i>printf</i> formatting specification; see <i>printf</i> . If the <i>value</i>       |
| 90314 |                   | field was a string, then the value from the file shall be the argument for the <i>printf</i>                     |
| 90315 |                   | formatting specification; otherwise, the value from the file shall be the argument.                              |

**EXIT STATUS**

The following exit values shall be returned:

0 Successful completion.

>0 An error occurred.

**CONSEQUENCES OF ERRORS**

Default.

**APPLICATION USAGE**

The *file* utility can only be required to guess at many of the file types because only exhaustive testing can determine some types with certainty. For example, binary data on some implementations might match the initial segment of an executable or a *tar* archive.

Note that the table indicates that the output contains the stated string. Systems may add text before or after the string. For executables, as an example, the machine architecture and various facts about how the file was link-edited may be included. Note also that on systems that recognize shell script files starting with "#!" as executable files, these may be identified as executable binary files rather than as shell scripts.

**EXAMPLES**

Determine whether an argument is a binary executable file:

```
file -- "$1" | grep -q '.*executable' &&
 printf "%s is executable.\n" "$1"
```

**RATIONALE**

The *-f* option was omitted because the same effect can (and should) be obtained using the *xargs* utility.

Historical versions of the *file* utility attempt to identify the following types of files: symbolic link, directory, character special, block special, socket, *tar* archive, *cpio* archive, SCCS archive, archive library, empty, *compress* output, *pack* output, binary data, C source, FORTRAN source, assembler source, *nroff*/*troff*/*eqn*/*tbl* source *troff* output, shell script, C shell script, English text, ASCII text, various executables, APL workspace, compiled terminfo entries, and CURSES screen images. Only those types that are reasonably well specified in POSIX or are directly related to POSIX utilities are listed in the table.

Historical systems have used a “magic file” named */etc/magic* to help identify file types. Because it is generally useful for users and scripts to be able to identify special file types, the *-m* flag and a portable format for user-created magic files has been specified. No requirement is made that an implementation of *file* use this method of identifying files, only that users be permitted to add their own classifying tests.

In addition, three options have been added to historical practice. The *-d* flag has been added to permit users to cause their tests to follow any default system tests. The *-i* flag has been added to permit users to test portably for regular files in shell scripts. The *-M* flag has been added to permit users to ignore any default system tests.

The POSIX.1-2008 description of default system tests and the interaction between the *-d*, *-M*, and *-m* options did not clearly indicate that there were two types of “default system tests”. The “position-sensitive tests” determine file types by looking for certain string or binary values at specific offsets in the file being examined. These position-sensitive tests were implemented in historical systems using the magic file described above. Some of these tests are now built into the *file* utility itself on some implementations so the output can provide more detail than can be provided by magic files. For example, a magic file can easily identify a **core** file on most implementations, but cannot name the program file that dropped the core. A magic file could

produce output such as:

```
/home/dwc/core: ELF 32-bit MSB core file SPARC Version 1
```

but by building the test into the *file* utility, you could get output such as:

```
/home/dwc/core: ELF 32-bit MSB core file SPARC Version 1, from 'testprog'
```

These extended built-in tests are still to be treated as position-sensitive default system tests even if they are not listed in */etc/magic* or any other magic file.

The context-sensitive default system tests were always built into the *file* utility. These tests looked for language constructs in text files trying to identify shell scripts, C, FORTRAN, and other computer language source files, and even plain text files. With the addition of the *-m* and *-M* options the distinction between position-sensitive and context-sensitive default system tests became important because the order of testing is important. The context-sensitive system default tests should never be applied before any position-sensitive tests even if the *-d* option is specified before a *-m* option or *-M* option due to the high probability that the context-sensitive system default tests will incorrectly identify arbitrary text files as text files before position-sensitive tests specified by the *-m* or *-M* option would be applied to give a more accurate identification.

Leaving the meaning of *-M* – and *-m* – unspecified allows an existing prototype of these options to continue to work in a backwards-compatible manner. (In that implementation, *-M* – was roughly equivalent to *-d* in POSIX.1-2008.)

The historical *-c* option was omitted as not particularly useful to users or portable shell scripts. In addition, a reasonable implementation of the *file* utility would report any errors found each time the magic file is read.

The historical format of the magic file was the same as that specified by the Rationale in the ISO POSIX-2:1993 standard for the *offset*, *value*, and *message* fields; however, it used less precise type fields than the format specified by the current normative text. The new type field values are a superset of the historical ones.

The following is an example magic file:

```
0 short 070707 cpio archive
0 short 0143561 Byte-swapped cpio archive
0 string 070707 ASCII cpio archive
0 long 0177555 Very old archive
0 short 0177545 Old archive
0 short 017437 Old packed data
0 string \037\036 Packed data
0 string \377\037 Compacted data
0 string \037\235 Compressed data
>2 byte&0x80 >0 Block compressed
>2 byte&0x1f x %d bits
0 string \032\001 Compiled Termino Entry
0 short 0433 Curses screen image
0 short 0434 Curses screen image
0 string <ar> System V Release 1 archive
0 string !<arch>\n___.SYMDEF Archive random library
0 string !<arch> Archive
0 string ARF_BEGARF PHIGS clear text archive
0 long 0x137A2950 Scalable OpenFont binary
0 long 0x137A2951 Encrypted scalable OpenFont binary
```

- 90408 The use of a basic integer data type is intended to allow the implementation to choose a word  
90409 size commonly used by applications on that architecture.
- 90410 Earlier versions of this standard allowed for implementations with bytes other than eight bits,  
90411 but this has been modified in this version.
- 90412 **FUTURE DIRECTIONS**
- 90413 None.
- 90414 **SEE ALSO**
- 90415 *ar, ls, pax, printf*
- 90416 XBD [Table 5-1](#) (on page 121), [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 215)
- 90417 **CHANGE HISTORY**
- 90418 First released in Issue 4.
- 90419 **Issue 6**
- 90420 This utility is marked as part of the User Portability Utilities option.
- 90421 Options and an EXTENDED DESCRIPTION are added as specified in the IEEE P1003.2b draft  
90422 standard.
- 90423 IEEE PASC Interpretations 1003.2 #192 and #178 are applied.
- 90424 IEEE Std 1003.1-2001/Cor 1-2002, item XCU/TC1/D6/25 is applied, making major changes to  
90425 address ambiguities raised in defect reports.
- 90426 IEEE Std 1003.1-2001/Cor 1-2002, item XCU/TC1/D6/26 is applied, making it clear in the  
90427 OPTIONS section that the **-m**, **-d**, and **-M** options do not comply with Guideline 11 of the  
90428 Utility Syntax Guidelines.
- 90429 IEEE Std 1003.1-2001/Cor 2-2004, item XCU/TC2/D6/10 is applied, clarifying the specification  
90430 characters.
- 90431 IEEE Std 1003.1-2001/Cor 2-2004, item XCU/TC2/D6/11 is applied, allowing application  
90432 developers to create portable magic files that can match characters in strings, and allowing  
90433 common extensions found in existing implementations.
- 90434 IEEE Std 1003.1-2001/Cor 2-2004, item XCU/TC2/D6/12 is applied, removing text describing  
90435 behavior on systems with bytes consisting of more than eight bits.
- 90436 **Issue 7**
- 90437 Austin Group Interpretation 1003.1-2001 #092 is applied.
- 90438 SD5-XCU-ERN-4 is applied, adding further entries in the Notes column in [Table 4-9](#).
- 90439 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.
- 90440 The *file* utility is moved from the User Portability Utilities option to the Base. User Portability  
90441 Utilities is now an option for interactive utilities.
- 90442 The EXAMPLES section is revised to correct an error with the pathname "\$1".

90443 **NAME**90444 `find` — find files90445 **SYNOPSIS**90446 `find [-H|-L] path... [operand_expression...]`90447 **DESCRIPTION**

90448 The *find* utility shall recursively descend the directory hierarchy from each file specified by *path*,  
 90449 evaluating a Boolean expression composed of the primaries described in the OPERANDS section  
 90450 for each file encountered. Each *path* operand shall be evaluated unaltered as it was provided,  
 90451 including all trailing <slash> characters; all pathnames for other files encountered in the  
 90452 hierarchy shall consist of the concatenation of the current *path* operand, a <slash> if the current  
 90453 *path* operand did not end in one, and the filename relative to the *path* operand. The relative  
 90454 portion shall contain no dot or dot-dot components, no trailing <slash> characters, and only  
 90455 single <slash> characters between pathname components.

90456 The *find* utility shall be able to descend to arbitrary depths in a file hierarchy and shall not fail  
 90457 due to path length limitations (unless a *path* operand specified by the application exceeds  
 90458 {PATH\_MAX} requirements).

90459 The *find* utility shall detect infinite loops; that is, entering a previously visited directory that is an  
 90460 ancestor of the last file encountered. When it detects an infinite loop, *find* shall write a  
 90461 diagnostic message to standard error and shall either recover its position in the hierarchy or  
 90462 terminate.

90463 If a file is removed from or added to the directory hierarchy being searched it is unspecified +  
 90464 whether or not *find* includes that file in its search.

90465 **OPTIONS**90466 The *find* utility shall conform to XBD [Section 12.2](#) (on page 215).

90467 The following options shall be supported by the implementation:

90468 **-H** Cause the file information and file type evaluated for each symbolic link  
 90469 encountered as a *path* operand on the command line to be those of the file  
 90470 referenced by the link, and not the link itself. If the referenced file does not exist,  
 90471 the file information and type shall be for the link itself. File information and type  
 90472 for symbolic links encountered during the traversal of a file hierarchy shall be that  
 90473 of the link itself.

90474 **-L** Cause the file information and file type evaluated for each symbolic link  
 90475 encountered as a *path* operand on the command line or encountered during the  
 90476 traversal of a file hierarchy to be those of the file referenced by the link, and not the  
 90477 link itself. If the referenced file does not exist, the file information and type shall be  
 90478 for the link itself.

90479 Specifying more than one of the mutually-exclusive options **-H** and **-L** shall not be considered  
 90480 an error. The last option specified shall determine the behavior of the utility. If neither the **-H**  
 90481 nor the **-L** option is specified, then the file information and type for symbolic links encountered  
 90482 as a *path* operand on the command line or encountered during the traversal of a file hierarchy  
 90483 shall be that of the link itself.

90484 **OPERANDS**

90485 The following operands shall be supported:

90486 The first operand and subsequent operands up to but not including the first operand that starts  
 90487 with a `'-'`, or is a `'!'` or a `'('`, shall be interpreted as *path* operands. If the first operand starts  
 90488 with a `'-'`, or is a `'!'` or a `'('`, the behavior is unspecified. Each *path* operand is a pathname of

a starting point in the file hierarchy.

The first operand that starts with a `'-'`, or is a `'!'` or a `'('`, and all subsequent arguments shall be interpreted as an *expression* made up of the following primaries and operators. In the descriptions, wherever *n* is used as a primary argument, it shall be interpreted as a decimal integer optionally preceded by a plus (`'+'`) or minus-sign (`'-'`) sign, as follows:

`+n` More than *n*.

`n` Exactly *n*.

`-n` Less than *n*.

The following primaries shall be supported:

**-name *pattern***  
The primary shall evaluate as true if the basename of the current pathname matches *pattern* using the pattern matching notation described in [Section 2.13](#) (on page 2354). The additional rules in [Section 2.13.3](#) (on page 2355) do not apply as this is a matching operation, not an expansion.

**-path *pattern***  
The primary shall evaluate as true if the current pathname matches *pattern* using the pattern matching notation described in [Section 2.13](#) (on page 2354). The additional rules in [Section 2.13.3](#) (on page 2355) do not apply as this is a matching operation, not an expansion.

**-nouser** The primary shall evaluate as true if the file belongs to a user ID for which the `getpwuid()` function defined in the System Interfaces volume of POSIX.1-2008 (or equivalent) returns NULL.

**-nogroup** The primary shall evaluate as true if the file belongs to a group ID for which the `getgrgid()` function defined in the System Interfaces volume of POSIX.1-2008 (or equivalent) returns NULL.

**-xdev** The primary shall always evaluate as true; it shall cause *find* not to continue descending past directories that have a different device ID (*st\_dev*, see the `stat()` function defined in the System Interfaces volume of POSIX.1-2008). If any **-xdev** primary is specified, it shall apply to the entire expression even if the **-xdev** primary would not normally be evaluated.

**-prune** The primary shall always evaluate as true; it shall cause *find* not to descend the current pathname if it is a directory. If the **-depth** primary is specified, the **-prune** primary shall have no effect.

**-perm [*-*]*mode***  
The *mode* argument is used to represent file mode bits. It shall be identical in format to the *symbolic\_mode* operand described in *chmod*, and shall be interpreted as follows. To start, a template shall be assumed with all file mode bits cleared. An *op* symbol of `'+'` shall set the appropriate mode bits in the template; `'-'` shall clear the appropriate bits; `'='` shall set the appropriate mode bits, without regard to the contents of the file mode creation mask of the process. The *op* symbol of `'-'` cannot be the first character of *mode*; this avoids ambiguity with the optional leading `<hyphen>`. Since the initial mode is all bits off, there are not any symbolic modes that need to use `'-'` as the first character.

If the `<hyphen>` is omitted, the primary shall evaluate as true when the file permission bits exactly match the value of the resulting template.



90534 Otherwise, if *mode* is prefixed by a <hyphen>, the primary shall evaluate as true if  
 90535 at least all the bits in the resulting template are set in the file permission bits.

90536 **-perm** [-]*onum*  
 90537 If the <hyphen> is omitted, the primary shall evaluate as true when the file mode  
 90538 bits exactly match the value of the octal number *onum* (see the description of the  
 90539 octal *mode* in *chmod*). Otherwise, if *onum* is prefixed by a <hyphen>, the primary  
 90540 shall evaluate as true if at least all of the bits specified in *onum* are set. In both  
 90541 cases, the behavior is unspecified when *onum* exceeds 07777.

90542 **-type** *c* The primary shall evaluate as true if the type of the file is *c*, where *c* is 'b', 'c',  
 90543 'd', 'l', 'p', 'f', or 's' for block special file, character special file, directory,  
 90544 symbolic link, FIFO, regular file, or socket, respectively.

90545 **-links** *n* The primary shall evaluate as true if the file has *n* links.

90546 **-user** *uname* The primary shall evaluate as true if the file belongs to the user *uname*. If *uname* is  
 90547 a decimal integer and the *getpwnam*() (or equivalent) function does not return a  
 90548 valid user name, *uname* shall be interpreted as a user ID.

90549 **-group** *gname*  
 90550 The primary shall evaluate as true if the file belongs to the group *gname*. If *gname*  
 90551 is a decimal integer and the *getgrnam*() (or equivalent) function does not return a  
 90552 valid group name, *gname* shall be interpreted as a group ID.

90553 **-size** *n*[*c*] The primary shall evaluate as true if the file size in bytes, divided by 512 and  
 90554 rounded up to the next integer, is *n*. If *n* is followed by the character '*c*', the size  
 90555 shall be in bytes.

90556 **-atime** *n* The primary shall evaluate as true if the file access time subtracted from the  
 90557 initialization time, divided by 86 400 (with any remainder discarded), is *n*.

90558 **-ctime** *n* The primary shall evaluate as true if the time of last change of file status  
 90559 information subtracted from the initialization time, divided by 86 400 (with any  
 90560 remainder discarded), is *n*.

90561 **-mtime** *n* The primary shall evaluate as true if the file modification time subtracted from the  
 90562 initialization time, divided by 86 400 (with any remainder discarded), is *n*.

90563 **-exec** *utility\_name* [*argument* ...];  
 90564 **-exec** *utility\_name* [*argument* ...] {} +  
 90565 The end of the primary expression shall be punctuated by a <semicolon> or by a  
 90566 <plus-sign>. Only a <plus-sign> that immediately follows an argument  
 90567 containing only the two characters "{}" shall punctuate the end of the primary  
 90568 expression. Other uses of the <plus-sign> shall not be treated as special.

90569 If the primary expression is punctuated by a <semicolon>, the utility *utility\_name*  
 90570 shall be invoked once for each pathname and the primary shall evaluate as true if  
 90571 the utility returns a zero value as exit status. A *utility\_name* or *argument* containing  
 90572 only the two characters "{}" shall be replaced by the current pathname. If a  
 90573 *utility\_name* or *argument* string contains the two characters "{}", but not just the  
 90574 two characters "{}", it is implementation-defined whether *find* replaces those two  
 90575 characters or uses the string without change.

90576 If the primary expression is punctuated by a <plus-sign>, the primary shall always  
 90577 evaluate as true, and the pathnames for which the primary is evaluated shall be  
 90578 aggregated into sets. The utility *utility\_name* shall be invoked once for each set of  
 90579 aggregated pathnames. Each invocation shall begin after the last pathname in the

90580 set is aggregated, and shall be completed before the *find* utility exits and before the  
 90581 first pathname in the next set (if any) is aggregated for this primary, but it is  
 90582 otherwise unspecified whether the invocation occurs before, during, or after the  
 90583 evaluations of other primaries. If any invocation returns a non-zero value as exit  
 90584 status, the *find* utility shall return a non-zero exit status. An argument containing  
 90585 only the two characters "{}" shall be replaced by the set of aggregated  
 90586 pathnames, with each pathname passed as a separate argument to the invoked  
 90587 utility in the same order that it was aggregated. The size of any set of two or more  
 90588 pathnames shall be limited such that execution of the utility does not cause the  
 90589 system's {ARG\_MAX} limit to be exceeded. If more than one argument containing  
 90590 the two characters "{}" is present, the behavior is unspecified.

90591 The current directory for the invocation of *utility\_name* shall be the same as the  
 90592 current directory when the *find* utility was started. If the *utility\_name* names any of  
 90593 the special built-in utilities (see [Section 2.14](#), on page 2356), the results are  
 90594 undefined.

90595 **-ok** *utility\_name* [*argument* ...];  
 90596 The **-ok** primary shall be equivalent to **-exec**, except that the use of a <plus-sign>  
 90597 to punctuate the end of the primary expression need not be supported, and *find*  
 90598 shall request affirmation of the invocation of *utility\_name* using the current file as  
 90599 an argument by writing to standard error as described in the STDERR section. If  
 90600 the response on standard input is affirmative, the utility shall be invoked.  
 90601 Otherwise, the command shall not be invoked and the value of the **-ok** operand  
 90602 shall be false.

90603 **-print** The primary shall always evaluate as true; it shall cause the current pathname to  
 90604 be written to standard output.

90605 **-newer** *file* The primary shall evaluate as true if the modification time of the current file is  
 90606 more recent than the modification time of the file named by the pathname *file*.

90607 **-depth** The primary shall always evaluate as true; it shall cause descent of the directory  
 90608 hierarchy to be done so that all entries in a directory are acted on before the  
 90609 directory itself. If a **-depth** primary is not specified, all entries in a directory shall  
 90610 be acted on after the directory itself. If any **-depth** primary is specified, it shall  
 90611 apply to the entire expression even if the **-depth** primary would not normally be  
 90612 evaluated.

90613 The primaries can be combined using the following operators (in order of decreasing  
 90614 precedence):

90615 (*expression*) True if *expression* is true.

90616 **! expression** Negation of a primary; the unary NOT operator.

90617 *expression* [**-a**] *expression*  
 90618 Conjunction of primaries; the AND operator is implied by the juxtaposition of two  
 90619 primaries or made explicit by the optional **-a** operator. The second expression shall  
 90620 not be evaluated if the first expression is false.

90621 *expression* **-o** *expression*  
 90622 Alternation of primaries; the OR operator. The second expression shall not be  
 90623 evaluated if the first expression is true.

90624 If no *expression* is present, **-print** shall be used as the expression. Otherwise, if the given  
 90625 expression does not contain any of the primaries **-exec**, **-ok**, or **-print**, the given expression

90626 shall be effectively replaced by:

90627 ( *given\_expression* ) -print

90628 The **-user**, **-group**, and **-newer** primaries each shall evaluate their respective arguments only  
90629 once.

90630 When the file type evaluated for the current file is a symbolic link, the results of evaluating the  
90631 **-perm** primary are implementation-defined.

90632 **STDIN**

90633 If the **-ok** primary is used, the response shall be read from the standard input. An entire line  
90634 shall be read as the response. Otherwise, the standard input shall not be used.

90635 **INPUT FILES**

90636 None.

90637 **ENVIRONMENT VARIABLES**

90638 The following environment variables shall affect the execution of *find*:

90639 **LANG** Provide a default value for the internationalization variables that are unset or null.  
90640 (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
90641 variables used to determine the values of locale categories.)

90642 **LC\_ALL** If set to a non-empty string value, override the values of all the other  
90643 internationalization variables.

90644 **LC\_COLLATE**

90645 Determine the locale for the behavior of ranges, equivalence classes, and multi-  
90646 character collating elements used in the pattern matching notation for the **-n**  
90647 option and in the extended regular expression defined for the **yesexpr** locale  
90648 keyword in the *LC\_MESSAGES* category.

90649 **LC\_CTYPE** This variable determines the locale for the interpretation of sequences of bytes of  
90650 text data as characters (for example, single-byte as opposed to multi-byte  
90651 characters in arguments), the behavior of character classes within the pattern  
90652 matching notation used for the **-n** option, and the behavior of character classes  
90653 within regular expressions used in the extended regular expression defined for the  
90654 **yesexpr** locale keyword in the *LC\_MESSAGES* category.

90655 **LC\_MESSAGES**

90656 Determine the locale used to process affirmative responses, and the locale used to  
90657 affect the format and contents of diagnostic messages and prompts written to  
90658 standard error.

90659 XSI **NLSPATH** Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

90660 **PATH** Determine the location of the *utility\_name* for the **-exec** and **-ok** primaries, as  
90661 described in XBD [Chapter 8](#) (on page 173).

90662 **ASYNCHRONOUS EVENTS**

90663 Default.

90664 **STDOUT**

90665 The **-print** primary shall cause the current pathnames to be written to standard output. The  
90666 format shall be:

90667 "%s\n", <path>

90668 **STDERR**

90669 The **-ok** primary shall write a prompt to standard error containing at least the *utility\_name* to be  
 90670 invoked and the current pathname. In the POSIX locale, the last non-<blank> in the prompt shall  
 90671 be ' ? ' . The exact format used is unspecified.

90672 Otherwise, the standard error shall be used only for diagnostic messages.

90673 **OUTPUT FILES**

90674 None.

90675 **EXTENDED DESCRIPTION**

90676 None.

90677 **EXIT STATUS**

90678 The following exit values shall be returned:

90679 0 All *path* operands were traversed successfully.

90680 >0 An error occurred.

90681 **CONSEQUENCES OF ERRORS**

90682 Default.

90683 **APPLICATION USAGE**

90684 When used in operands, pattern matching notation, <semicolon>, <left-parenthesis>, and  
 90685 <right-parenthesis> characters are special to the shell and must be quoted (see [Section 2.2](#), on  
 90686 page 2320).

90687 The bit that is traditionally used for sticky (historically 01000) is specified in the **-perm** primary  
 90688 using the octal number argument form. Since this bit is not defined by this volume of  
 90689 POSIX.1-2008, applications must not assume that it actually refers to the traditional sticky bit.

90690 **EXAMPLES**

90691 1. The following commands are equivalent:

90692 find .  
 90693 find . -print

90694 They both write out the entire directory hierarchy from the current directory.

90695 2. The following command:

90696 find / \( -name tmp -o -name '\*.xx' \) -atime +7 -exec rm {} \;

90697 removes all files named **tmp** or ending in **.xx** that have not been accessed for seven or  
 90698 more 24-hour periods.

90699 3. The following command:

90700 find . -perm -o+w,+s

90701 prints (**-print** is assumed) the names of all files in or below the current directory, with all  
 90702 of the file permission bits S\_ISUID, S\_ISGID, and S\_IWOTH set.

90703 4. The following command:

90704 find . -name SCCS -prune -o -print

90705 recursively prints pathnames of all files in the current directory and below, but skips  
 90706 directories named SCCS and files in them.

5. The following command:

```
find . -print -name SCCS -prune
```

behaves as in the previous example, but prints the names of the SCCS directories.

6. The following command is roughly equivalent to the **-nt** extension to *test*:

```
if [-n "$(find file1 -prune -newer file2)"]; then
 printf %s\\n "file1 is newer than file2"
fi
```

7. The descriptions of **-atime**, **-ctime**, and **-mtime** use the terminology *n* “86 400 second periods (days)”. For example, a file accessed at 23:59 is selected by:

```
find . -atime -1 -print
```

at 00:01 the next day (less than 24 hours later, not more than one day ago); the midnight boundary between days has no effect on the 24-hour calculation.

8. The following command:

```
find . ! -name . -prune -name '*.old' -exec \
 sh -c 'mv "$@" ../old/' sh {} +
```

performs the same task as:

```
mv /*.old ../old /*.old ../old/
```

while avoiding an “Argument list too long” error if there are a large number of files ending with **.old** and without running *mv* if there are no such files (and avoiding “No such file or directory” errors if **../old** does not exist or no files match **/\*.old** or **../old**).

The alternative:

```
find . ! -name . -prune -name '*.old' -exec mv {} ../old/ \;
```

is less efficient if there are many files to move because it executes one *mv* command per file.

9. On systems configured to mount removable media on directories under **/media**, the following command searches the file hierarchy for files larger than 100 000 KB without searching any mounted removable media:

```
find / -path /media -prune -o -size +200000 -print
```

10. Except for the root directory, and **"/"** on implementations where **"/"** does not refer to the root directory, no pattern given to **-name** will match a **<slash>**, because trailing **<slash>** characters are ignored when computing the basename of the file under evaluation. Given two empty directories named **foo** and **bar**, the following command:

```
find foo/// bar/// -name foo -o -name 'bar?*'
```

prints only the line **foo///**.

## RATIONALE

The **-a** operator was retained as an optional operator for compatibility with historical shell scripts, even though it is redundant with expression concatenation.

The descriptions of the **'-'** modifier on the *mode* and *onum* arguments to the **-perm** primary agree with historical practice on BSD and System V implementations. System V and BSD documentation both describe it in terms of checking additional bits; in fact, it uses the same bits, but checks for having at least all of the matching bits set instead of having exactly the matching

bits set.

The exact format of the interactive prompts is unspecified. Only the general nature of the contents of prompts are specified because:

- Implementations may desire more descriptive prompts than those used on historical implementations.
- Since the historical prompt strings do not terminate with <newline> characters, there is no portable way for another program to interact with the prompts of this utility via pipes.

Therefore, an application using this prompting option relies on the system to provide the most suitable dialog directly with the user, based on the general guidelines specified.

The **-name** *file* operand was changed to use the shell pattern matching notation so that *find* is consistent with other utilities using pattern matching.

The **-size** operand refers to the size of a file, rather than the number of blocks it may occupy in the file system. The intent is that the *st\_size* field defined in the System Interfaces volume of POSIX.1-2008 should be used, not the *st\_blocks* found in historical implementations. There are at least two reasons for this:

1. In both System V and BSD, *find* only uses *st\_size* in size calculations for the operands specified by this volume of POSIX.1-2008. (BSD uses *st\_blocks* only when processing the **-ls** primary.)
2. Users usually think of file size in terms of bytes, which is also the unit used by the *ls* utility for the output from the **-l** option. (In both System V and BSD, *ls* uses *st\_size* for the **-l** option size field and uses *st\_blocks* for the *ls -s* calculations. This volume of POSIX.1-2008 does not specify *ls -s*.)

The descriptions of **-atime**, **-ctime**, and **-mtime** were changed from the SVID description of *n* “days” to *n* being the result of the integer division of the time difference in seconds by 86 400. The description is also different in terms of the exact timeframe for the *n* case (*versus* the *+n* or *-n*), but it matches all known historical implementations. It refers to one 86 400 second period in the past, not any time from the beginning of that period to the current time. For example, **-atime 2** is true if the file was accessed any time in the period from 72 hours to 48 hours ago.

Historical implementations do not modify “{ }” when it appears as a substring of an **-exec** or **-ok** *utility\_name* or argument string. There have been numerous user requests for this extension, so this volume of POSIX.1-2008 allows the desired behavior. At least one recent implementation does support this feature, but encountered several problems in managing memory allocation and dealing with multiple occurrences of “{ }” in a string while it was being developed, so it is not yet required behavior.

Assuming the presence of **-print** was added to correct a historical pitfall that plagues novice users, it is entirely upwards-compatible from the historical System V *find* utility. In its simplest form (*find directory*), it could be confused with the historical BSD fast *find*. The BSD developers agreed that adding **-print** as a default expression was the correct decision and have added the fast *find* functionality within a new utility called *locate*.

Historically, the **-L** option was implemented using the primary **-follow**. The **-H** and **-L** options were added for two reasons. First, they offer a finer granularity of control and consistency with other programs that walk file hierarchies. Second, the **-follow** primary always evaluated to true. As they were historically really global variables that took effect before the traversal began, some valid expressions had unexpected results. An example is the expression **-print -o -follow**. Because **-print** always evaluates to true, the standard order of evaluation implies that **-follow** would never be evaluated. This was never the case. Historical practice for the **-follow** primary,

however, is not consistent. Some implementations always follow symbolic links on the command line whether **-follow** is specified or not. Others follow symbolic links on the command line only if **-follow** is specified. Both behaviors are provided by the **-H** and **-L** options, but scripts using the current **-follow** primary would be broken if the **-follow** option is specified to work either way.

Since the **-L** option resolves all symbolic links and the **-type l** primary is true for symbolic links that still exist after symbolic links have been resolved, the command:

```
find -L . -type l
```

prints a list of symbolic links reachable from the current directory that do not resolve to accessible files.

A feature of SVR4's *find* utility was the **-exec** primary's **+** terminator. This allowed filenames containing special characters (especially <newline> characters) to be grouped together without the problems that occur if such filenames are piped to *xargs*. Other implementations have added other ways to get around this problem, notably a **-print0** primary that wrote filenames with a null byte terminator. This was considered here, but not adopted. Using a null terminator meant that any utility that was going to process *find*'s **-print0** output had to add a new option to parse the null terminators it would now be reading.

The **"-exec ... {} +"** syntax adopted was a result of IEEE PASC Interpretation 1003.2 #210. It should be noted that this is an incompatible change to IEEE Std 1003.2-1992. For example, the following command printed all files with a **'-'** after their name if they are regular files, and a **'+'** otherwise:

```
find / -type f -exec echo {} - ';' -o -exec echo {} + ';' -
```

The change invalidates usage like this. Even though the previous standard stated that this usage would work, in practice many did not support it and the standard developers felt it better to now state that this was not allowable.

## FUTURE DIRECTIONS

None.

## SEE ALSO

[Section 2.2](#) (on page 2320), [Section 2.13](#) (on page 2354), [Section 2.14](#) (on page 2356), [chmod](#), [mv](#), [pax](#), [sh](#), [test](#)

XBD [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 215)

XSH [fstatat\(\)](#), [getgrgid\(\)](#), [getpwuid\(\)](#)

## CHANGE HISTORY

First released in Issue 2.

### Issue 5

The FUTURE DIRECTIONS section is added.

### Issue 6

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The **-perm** [**-l**onum primary is supported.

The *find* utility is aligned with the IEEE P1003.2b draft standard, to include processing of symbolic links and changes to the description of the **atime**, **ctime**, and **mtime** operands.

IEEE PASC Interpretation 1003.2 #210 is applied, extending the **-exec** operand.

90837 IEEE Std 1003.1-2001/Cor 2-2004, item XCU/TC2/D6/13 is applied, updating the RATIONALE  
90838 section to be consistent with the normative text.

90839 **Issue 7**

90840 Austin Group Interpretation 1003.1-2001 #126 is applied, changing the description of the  
90841 *LC\_MESSAGES* environment variable.

90842 Austin Group Interpretation 1003.1-2001 #127 is applied, rephrasing the description of the **-exec**  
90843 primary to be “immediately follows”.

90844 Austin Group Interpretation 1003.1-2001 #185 is applied, clarifying the requirements for the **-H**  
90845 and **-L** options.

90846 Austin Group Interpretation 1003.1-2001 #186 is applied, clarifying the requirements for the  
90847 evaluation of *path* operands.

90848 Austin Group Interpretation 1003.1-2001 #195 is applied, clarifying the interpretation of the first  
90849 operand.

90850 SD5-XCU-ERN-48 is applied, clarifying the **-L** option in the case that the referenced file does not  
90851 exist.

90852 SD5-XCU-ERN-89 is applied, updating the OPERANDS section.

90853 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

90854 SD5-XCU-ERN-117 is applied, clarifying the **-perm** operand.

90855 SD5-XCU-ERN-122 is applied, adding a new EXAMPLE.

90856 The description of the **-name** primary is revised and the **-path** primary is added (with a new  
90857 example).

90858 POSIX.1-2008, Technical Corrigendum 1, XCU/TC1-2008/0086 [365], XCU/TC1-2008/0087 +  
90859 [310], XCU/TC1-2008/0088 [309,310,430], XCU/TC1-2008/0089 [235], and XCU/TC1-2008/0090 +  
90860 [445] are applied.



90861 **NAME**

90862 fold — filter for folding lines

90863 **SYNOPSIS**90864 fold [-bs] [-w *width*] [*file...*]90865 **DESCRIPTION**

90866 The *fold* utility is a filter that shall fold lines from its input files, breaking the lines to have a  
 90867 maximum of *width* column positions (or bytes, if the **-b** option is specified). Lines shall be  
 90868 broken by the insertion of a <newline> such that each output line (referred to later in this section  
 90869 as a *segment*) is the maximum width possible that does not exceed the specified number of  
 90870 column positions (or bytes). A line shall not be broken in the middle of a character. The behavior  
 90871 is undefined if *width* is less than the number of columns any single character in the input would  
 90872 occupy.

90873 If the <carriage-return>, <backspace>, or <tab> characters are encountered in the input, and the  
 90874 **-b** option is not specified, they shall be treated specially:

90875 <backspace> The current count of line width shall be decremented by one, although the count  
 90876 never shall become negative. The *fold* utility shall not insert a <newline>  
 90877 immediately before or after any <backspace>, unless the following character has a  
 90878 width greater than 1 and would cause the line width to exceed *width*.

90879 <carriage-return>  
 90880 The current count of line width shall be set to zero. The *fold* utility shall not insert a  
 90881 <newline> immediately before or after any <carriage-return>.

90882 <tab> Each <tab> encountered shall advance the column position pointer to the next tab  
 90883 stop. Tab stops shall be at each column position *n* such that *n* modulo 8 equals 1.

90884 **OPTIONS**90885 The *fold* utility shall conform to XBD [Section 12.2](#) (on page 215).

90886 The following options shall be supported:

90887 **-b** Count *width* in bytes rather than column positions.

90888 **-s** If a segment of a line contains a <blank> within the first *width* column positions (or  
 90889 bytes), break the line after the last such <blank> meeting the width constraints. If  
 90890 there is no <blank> meeting the requirements, the **-s** option shall have no effect for  
 90891 that output segment of the input line.

90892 **-w *width*** Specify the maximum line length, in column positions (or bytes if **-b** is specified).  
 90893 The results are unspecified if *width* is not a positive decimal number. The default  
 90894 value shall be 80.

90895 **OPERANDS**

90896 The following operand shall be supported:

90897 *file* A pathname of a text file to be folded. If no *file* operands are specified, the standard  
 90898 input shall be used.

90899 **STDIN**

90900 The standard input shall be used if no *file* operands are specified, and shall be used if a *file*  
 90901 operand is '-' and the implementation treats the '-' as meaning standard input. Otherwise,  
 90902 the standard input shall not be used. See the INPUT FILES section.

90903 **INPUT FILES**

90904 If the **-b** option is specified, the input files shall be text files except that the lines are not limited  
 90905 to {LINE\_MAX} bytes in length. If the **-b** option is not specified, the input files shall be text files.

90906 **ENVIRONMENT VARIABLES**

90907 The following environment variables shall affect the execution of *fold*:

90908 **LANG** Provide a default value for the internationalization variables that are unset or null.  
 90909 (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 90910 variables used to determine the values of locale categories.)

90911 **LC\_ALL** If set to a non-empty string value, override the values of all the other  
 90912 internationalization variables.

90913 **LC\_CTYPE** Determine the locale for the interpretation of sequences of bytes of text data as  
 90914 characters (for example, single-byte as opposed to multi-byte characters in  
 90915 arguments and input files), and for the determination of the width in column  
 90916 positions each character would occupy on a constant-width font output device.

90917 **LC\_MESSAGES**  
 90918 Determine the locale that should be used to affect the format and contents of  
 90919 diagnostic messages written to standard error.

90920 XSI **NLSPATH** Determine the location of message catalogs for the processing of **LC\_MESSAGES**.

90921 **ASYNCHRONOUS EVENTS**

90922 Default.

90923 **STDOUT**

90924 The standard output shall be a file containing a sequence of characters whose order shall be  
 90925 preserved from the input files, possibly with inserted <newline> characters.

90926 **STDERR**

90927 The standard error shall be used only for diagnostic messages.

90928 **OUTPUT FILES**

90929 None.

90930 **EXTENDED DESCRIPTION**

90931 None.

90932 **EXIT STATUS**

90933 The following exit values shall be returned:

90934 0 All input files were processed successfully.

90935 >0 An error occurred.

90936 **CONSEQUENCES OF ERRORS**

90937 Default.

**APPLICATION USAGE**

The *cut* and *fold* utilities can be used to create text files out of files with arbitrary line lengths. The *cut* utility should be used when the number of lines (or records) needs to remain constant. The *fold* utility should be used when the contents of long lines need to be kept contiguous.

The *fold* utility is frequently used to send text files to printers that truncate, rather than fold, lines wider than the printer is able to print (usually 80 or 132 column positions).

**EXAMPLES**

An example invocation that submits a file of possibly long lines to the printer (under the assumption that the user knows the line width of the printer to be assigned by *lp*):

```
fold -w 132 bigfile | lp
```

**RATIONALE**

Although terminal input in canonical processing mode requires the erase character (frequently set to <backspace>) to erase the previous character (not byte or column position), terminal output is not buffered and is extremely difficult, if not impossible, to parse correctly; the interpretation depends entirely on the physical device that actually displays/prints/stores the output. In all known internationalized implementations, the utilities producing output for mixed column-width output assume that a <backspace> character backs up one column position and outputs enough <backspace> characters to return to the start of the character when <backspace> is used to provide local line motions to support underlining and emboldening operations. Since *fold* without the **-b** option is dealing with these same constraints, <backspace> is always treated as backing up one column position rather than backing up one character.

Historical versions of the *fold* utility assumed 1 byte was one character and occupied one column position when written out. This is no longer always true. Since the most common usage of *fold* is believed to be folding long lines for output to limited-length output devices, this capability was preserved as the default case. The **-b** option was added so that applications could *fold* files with arbitrary length lines into text files that could then be processed by the standard utilities. Note that although the width for the **-b** option is in bytes, a line is never split in the middle of a character. (It is unspecified what happens if a width is specified that is too small to hold a single character found in the input followed by a <newline>.)

The tab stops are hardcoded to be every eighth column to meet historical practice. No new method of specifying other tab stops was invented.

**FUTURE DIRECTIONS**

None.

**SEE ALSO**

*cut*

XBD [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 215)

**CHANGE HISTORY**

First released in Issue 4.

**Issue 6**

The normative text is reworded to avoid use of the term “must” for application requirements.

**Issue 7**

Austin Group Interpretation 1003.1-2001 #092 is applied.

90980  
90981  
90982

Austin Group Interpretation 1003.1-2001 #204 is applied, updating the DESCRIPTION to clarify when a <newline> can be inserted before or after a <backspace>.

SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

90983 **NAME**90984 fort77 — FORTRAN compiler (**FORTRAN**)90985 **SYNOPSIS**

```
90986 FD fort77 [-c] [-g] [-L directory]... [-O optlevel] [-o outfile] [-s]
90987 [-w] operand...
```

90988 **DESCRIPTION**

90989 The *fort77* utility is the interface to the FORTRAN compilation system; it shall accept the full  
 90990 FORTRAN-77 language defined by the ANSI X3.9-1978 standard. The system conceptually  
 90991 consists of a compiler and link editor. The files referenced by *operands* are compiled and linked  
 90992 to produce an executable file. It is unspecified whether the linking occurs entirely within the  
 90993 operation of *fort77*; some implementations may produce objects that are not fully resolved until  
 90994 the file is executed.

90995 If the *-c* option is present, for all pathname operands of the form *file.f*, the files:

90996  $\$(\text{basename } \textit{pathname.f}) .o$

90997 shall be created or overwritten as the result of successful compilation. If the *-c* option is not  
 90998 specified, it is unspecified whether such *.o* files are created or deleted for the *file.f* operands.

90999 If there are no options that prevent link editing (such as *-c*) and all operands compile and link  
 91000 without error, the resulting executable file shall be written into the file named by the *-o* option  
 91001 (if present) or to the file **a.out**. The executable file shall be created as specified in the System  
 91002 Interfaces volume of POSIX.1-2008, except that the file permissions shall be set to:

91003  $S\_IRWXO \mid S\_IRWXG \mid S\_IRWXU$

91004 and that the bits specified by the *umask* of the process shall be cleared.

91005 **OPTIONS**

91006 The *fort77* utility shall conform to XBD [Section 12.2](#) (on page 215), except that:

- 91007 • The *-l library* operands have the format of options, but their position within a list of  
 91008 operands affects the order in which libraries are searched.
- 91009 • The order of specifying the multiple *-L* options is significant.
- 91010 • Conforming applications shall specify each option separately; that is, grouping option  
 91011 letters (for example, *-cg*) need not be recognized by all implementations.

91012 The following options shall be supported:

- |       |                   |                                                                                                                                                                                                                                                                                                                                               |
|-------|-------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 91013 | <b>-c</b>         | Suppress the link-edit phase of the compilation, and do not remove any object files that are produced.                                                                                                                                                                                                                                        |
| 91014 |                   |                                                                                                                                                                                                                                                                                                                                               |
| 91015 | <b>-g</b>         | Produce symbolic information in the object or executable files; the nature of this information is unspecified, and may be modified by implementation-defined interactions with other options.                                                                                                                                                 |
| 91016 |                   |                                                                                                                                                                                                                                                                                                                                               |
| 91017 |                   |                                                                                                                                                                                                                                                                                                                                               |
| 91018 | <b>-s</b>         | Produce object or executable files, or both, from which symbolic and other information not required for proper execution using the <i>exec</i> family of functions defined in the System Interfaces volume of POSIX.1-2008 has been removed (stripped). If both <i>-g</i> and <i>-s</i> options are present, the action taken is unspecified. |
| 91019 |                   |                                                                                                                                                                                                                                                                                                                                               |
| 91020 |                   |                                                                                                                                                                                                                                                                                                                                               |
| 91021 |                   |                                                                                                                                                                                                                                                                                                                                               |
| 91022 | <b>-o outfile</b> | Use the pathname <i>outfile</i> , instead of the default <b>a.out</b> , for the executable file produced. If the <i>-o</i> option is present with <i>-c</i> , the result is unspecified.                                                                                                                                                      |
| 91023 |                   |                                                                                                                                                                                                                                                                                                                                               |

- 91024        **-L *directory***    Change the algorithm of searching for the libraries named in **-I** operands to look in  
 91025        the directory named by the *directory* pathname before looking in the usual places.  
 91026        Directories named in **-L** options shall be searched in the specified order. At least  
 91027        ten instances of this option shall be supported in a single *fort77* command  
 91028        invocation. If a directory specified by a **-L** option contains a file named **libf.a**, the  
 91029        results are unspecified.
- 91030        **-O *optlevel***    Specify the level of code optimization. If the *optlevel* option-argument is the digit  
 91031        '0', all special code optimizations shall be disabled. If it is the digit '1', the  
 91032        nature of the optimization is unspecified. If the **-O** option is omitted, the nature of  
 91033        the system's default optimization is unspecified. It is unspecified whether code  
 91034        generated in the presence of the **-O 0** option is the same as that generated when  
 91035        **-O** is omitted. Other *optlevel* values may be supported.
- 91036        **-w**                Suppress warnings.
- 91037        Multiple instances of **-L** options can be specified.

## 91038 OPERANDS

- 91039        An *operand* is either in the form of a pathname or the form **-I *library***. At least one operand of the  
 91040        pathname form shall be specified. The following operands shall be supported:
- 91041        *file.f*            The pathname of a FORTRAN source file to be compiled and optionally passed to  
 91042        the link editor. The filename operand shall be of this form if the **-c** option is used.
- 91043        *file.a*            A library of object files typically produced by *ar*, and passed directly to the link  
 91044        editor. Implementations may recognize implementation-defined suffixes other  
 91045        than **.a** as denoting object file libraries.
- 91046        *file.o*            An object file produced by *fort77 -c* and passed directly to the link editor.  
 91047        Implementations may recognize implementation-defined suffixes other than **.o** as  
 91048        denoting object files.
- 91049        The processing of other files is implementation-defined.
- 91050        **-I *library***        (The letter ell.) Search the library named:
- 91051                *liblibrary.a*
- 91052                A library is searched when its name is encountered, so the placement of a **-I**  
 91053                operand is significant. Several standard libraries can be specified in this manner, as  
 91054                described in the EXTENDED DESCRIPTION section. Implementations may  
 91055                recognize implementation-defined suffixes other than **.a** as denoting libraries.

## 91056 STDIN

- 91057        Not used.

## 91058 INPUT FILES

- 91059        The input file shall be one of the following: a text file containing FORTRAN source code; an  
 91060        object file in the format produced by *fort77 -c*; or a library of object files, in the format produced  
 91061        by archiving zero or more object files, using *ar*. Implementations may supply additional utilities  
 91062        that produce files in these formats. Additional input files are implementation-defined.
- 91063        A <tab> encountered within the first six characters on a line of source code shall cause the  
 91064        compiler to interpret the following character as if it were the seventh character on the line (that  
 91065        is, in column 7).

91066 **ENVIRONMENT VARIABLES**91067 The following environment variables shall affect the execution of *fort77*:

91068 *LANG* Provide a default value for the internationalization variables that are unset or null.  
 91069 (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 91070 variables used to determine the values of locale categories.)

91071 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
 91072 internationalization variables.

91073 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
 91074 characters (for example, single-byte as opposed to multi-byte characters in  
 91075 arguments and input files).

91076 *LC\_MESSAGES*

91077 Determine the locale that should be used to affect the format and contents of  
 91078 diagnostic messages written to standard error.

91079 XSI *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

91080 *TMPDIR* Determine the pathname that should override the default directory for temporary  
 91081 files, if any.

91082 **ASYNCHRONOUS EVENTS**

91083 Default.

91084 **STDOUT**

91085 Not used.

91086 **STDERR**

91087 The standard error shall be used only for diagnostic messages. If more than one *file* operand  
 91088 ending in *.f* (or possibly other unspecified suffixes) is given, for each such file:

91089 "%s:\n", <*file*>

91090 may be written to allow identification of the diagnostic message with the appropriate input file.

91091 This utility may produce warning messages about certain conditions that do not warrant  
 91092 returning an error (non-zero) exit value.

91093 **OUTPUT FILES**

91094 Object files, listing files, and executable files shall be produced in unspecified formats.

91095 **EXTENDED DESCRIPTION**91096 **Standard Libraries**91097 The *fort77* utility shall recognize the following *-l* operand for the standard library:

91098 *-l f* This library contains all functions referenced in the ANSI X3.9-1978 standard. This  
 91099 operand shall not be required to be present to cause a search of this library.

91100 In the absence of options that inhibit invocation of the link editor, such as *-c*, the *fort77* utility  
 91101 shall cause the equivalent of a *-l f* operand to be passed to the link editor as the last *-l* operand,  
 91102 causing it to be searched after all other object files and libraries are loaded.

91103 It is unspecified whether the library *libf.a* exists as a regular file. The implementation may  
 91104 accept as *-l* operands names of objects that do not exist as regular files.

**External Symbols**

The FORTRAN compiler and link editor shall support the significance of external symbols up to a length of at least 31 bytes; case folding is permitted. The action taken upon encountering symbols exceeding the implementation-defined maximum symbol length is unspecified.

The compiler and link editor shall support a minimum of 511 external symbols per source or object file, and a minimum of 4095 external symbols total. A diagnostic message is written to standard output if the implementation-defined limit is exceeded; other actions are unspecified.

**EXIT STATUS**

The following exit values shall be returned:

0 Successful compilation or link edit.

>0 An error occurred.

**CONSEQUENCES OF ERRORS**

When *fort77* encounters a compilation error, it shall write a diagnostic to standard error and continue to compile other source code operands. It shall return a non-zero exit status, but it is implementation-defined whether an object module is created. If the link edit is unsuccessful, a diagnostic message shall be written to standard error, and *fort77* shall exit with a non-zero status.

**APPLICATION USAGE**

None.

**EXAMPLES**

The following usage example compiles **xyz.f** and creates the executable file **foo**:

```
fort77 -o foo xyz.f
```

The following example compiles **xyz.f** and creates the object file **xyz.o**:

```
fort77 -c xyz.f
```

The following example compiles **xyz.f** and creates the executable file **a.out**:

```
fort77 xyz.f
```

The following example compiles **xyz.f**, links it with **b.o**, and creates the executable **a.out**:

```
fort77 xyz.f b.o
```

**RATIONALE**

The name of this utility was chosen as *fort77* to parallel the renaming of the C compiler. The name *f77* was not chosen to avoid problems with historical implementations. The ANSI X3.9-1978 standard was selected as a normative reference because the ISO/IEC version of FORTRAN-77 has been superseded by the ISO/IEC 1539:1991 standard.

The file inclusion and symbol definition **#define** mechanisms used by the *c99* utility were not included in this volume of POSIX.1-2008—even though they are commonly implemented—since there is no requirement that the FORTRAN compiler use the C preprocessor.

The **-onetrip** option was not included in this volume of POSIX.1-2008, even though many historical compilers support it, because it is derived from FORTRAN-66; it is an anachronism that should not be perpetuated.

Some implementations produce compilation listings. This aspect of FORTRAN has been left unspecified because there was controversy concerning the various methods proposed for implementing it: a **-V** option overlapped with historical vendor practice and a naming convention of creating files with **.l** suffixes collided with historical *lex* file naming practice.



There is no **-I** option in this version of this volume of POSIX.1-2008 to specify a directory for file inclusion. An **INCLUDE** directive has been a part of the Fortran-90 discussions, but an interface supporting that standard is not in the current scope.

It is noted that many FORTRAN compilers produce an object module even when compilation errors occur; during a subsequent compilation, the compiler may patch the object module rather than recompiling all the code. Consequently, it is left to the implementor whether or not an object file is created.

A reference to MIL-STD-1753 was removed from an early proposal in response to a request from the POSIX FORTRAN-binding standard developers. It was not the intention of the standard developers to require certification of the FORTRAN compiler, and IEEE Std 1003.9-1992 does not specify the military standard or any special preprocessing requirements. Furthermore, use of that document would have been inappropriate for an international standard.

The specification of optimization has been subject to changes through early proposals. At one time, **-O** and **-N** were Booleans: optimize and do not optimize (with an unspecified default). Some historical practice led this to be changed to:

**-O 0**           No optimization.

**-O 1**           Some level of optimization.

**-O n**           Other, unspecified levels of optimization.

It is not always clear whether “good code generation” is the same thing as optimization. Simple optimizations of local actions do not usually affect the semantics of a program. The **-O 0** option has been included to accommodate the very particular nature of scientific calculations in a highly optimized environment; compilers make errors. Some degree of optimization is expected, even if it is not documented here, and the ability to shut it off completely could be important when porting an application. An implementation may treat **-O 0** as “do less than normal” if it wishes, but this is only meaningful if any of the operations it performs can affect the semantics of a program. It is highly dependent on the implementation whether doing less than normal is logical. It is not the intent of the **-O 0** option to ask for inefficient code generation, but rather to assure that any semantically visible optimization is suppressed.

The specification of standard library access is consistent with the C compiler specification. Implementations are not required to have **/usr/lib/libf.a**, as many historical implementations do, but if not they are required to recognize **f** as a token.

External symbol size limits are in normative text; conforming applications need to know these limits. However, the minimum maximum symbol length should be taken as a constraint on a conforming application, not on an implementation, and consequently the action taken for a symbol exceeding the limit is unspecified. The minimum size for the external symbol table was added for similar reasons.

The CONSEQUENCES OF ERRORS section clearly specifies the behavior of the compiler when compilation or link-edit errors occur. The behavior of several historical implementations was examined, and the choice was made to be silent on the status of the executable, or **a.out**, file in the face of compiler or linker errors. If a linker writes the executable file, then links it on disk with *lseek()*s and *write()*s, the partially linked executable file can be left on disk and its execute bits turned off if the link edit fails. However, if the linker links the image in memory before writing the file to disk, it need not touch the executable file (if it already exists) because the link edit fails. Since both approaches are historical practice, a conforming application shall rely on the exit status of *fort77*, rather than on the existence or mode of the executable file.

The **-g** and **-s** options are not specified as mutually-exclusive. Historically, these two options

91193 have been mutually-exclusive, but because both are so loosely specified, it seemed appropriate  
91194 to leave their interaction unspecified.

91195 The requirement that conforming applications specify compiler options separately is to reserve  
91196 the multi-character option name space for vendor-specific compiler options, which are known to  
91197 exist in many historical implementations. Implementations are not required to recognize, for  
91198 example, `-gc` as if it were `-g -c`; nor are they forbidden from doing so. The SYNOPSIS shows all  
91199 of the options separately to highlight this requirement on applications.

91200 Echoing filenames to standard error is considered a diagnostic message because it would  
91201 otherwise be difficult to associate an error message with the erring file. They are described with  
91202 “may” to allow implementations to use other methods of identifying files and to parallel the  
91203 description in *c99*.

#### 91204 FUTURE DIRECTIONS

91205 A compilation system based on the ISO/IEC 1539:1991 standard may be considered for a future  
91206 version; it may have a different utility name from *fort77*.

#### 91207 SEE ALSO

91208 *ar*, *asa*, *c99*, *umask*

91209 XBD [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 215)

91210 XSH *exec*

#### 91211 CHANGE HISTORY

91212 First released in Issue 4.

#### 91213 Issue 6

91214 This utility is marked as part of the FORTRAN Development Utilities option.

91215 The normative text is reworded to avoid use of the term “must” for application requirements.

#### 91216 Issue 7

91217 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

91218 **NAME**91219 **fuser** — list process IDs of all processes that have one or more files open91220 **SYNOPSIS**91221 XSI `fuser [-cfu] file...`91222 **DESCRIPTION**

91223 The *fuser* utility shall write to standard output the process IDs of processes running on the local  
 91224 system that have one or more named files open. For block special devices, all processes using  
 91225 any file on that device are listed.

91226 The *fuser* utility shall write to standard error additional information about the named files  
 91227 indicating how the file is being used.

91228 Any output for processes running on remote systems that have a named file open is unspecified.

91229 A user may need appropriate privileges to invoke the *fuser* utility.

91230 **OPTIONS**

91231 The *fuser* utility shall conform to XBD [Section 12.2](#) (on page 215).

91232 The following options shall be supported:

91233 **-c** The file is treated as a mount point and the utility shall report on any files open in  
 91234 the file system.

91235 **-f** The report shall be only for the named files.

91236 **-u** The user name, in parentheses, associated with each process ID written to standard  
 91237 output shall be written to standard error.

91238 **OPERANDS**

91239 The following operand shall be supported:

91240 *file* A pathname on which the file or file system is to be reported.

91241 **STDIN**

91242 Not used.

91243 **INPUT FILES**

91244 The user database.

91245 **ENVIRONMENT VARIABLES**

91246 The following environment variables shall affect the execution of *fuser*:

91247 **LANG** Provide a default value for the internationalization variables that are unset or null.  
 91248 (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 91249 variables used to determine the values of locale categories.)

91250 **LC\_ALL** If set to a non-empty string value, override the values of all the other  
 91251 internationalization variables.

91252 **LC\_CTYPE** Determine the locale for the interpretation of sequences of bytes of text data as  
 91253 characters (for example, single-byte as opposed to multi-byte characters in  
 91254 arguments).

91255 **LC\_MESSAGES**

91256 Determine the locale that should be used to affect the format and contents of  
 91257 diagnostic messages written to standard error.

91258 *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

91259 **ASYNCHRONOUS EVENTS**

91260 Default.

91261 **STDOUT**

91262 The *fuser* utility shall write the process ID for each process using each file given as an operand to

91263 standard output in the following format:

91264 "%d", <*process\_id*>

91265 **STDERR**

91266 The *fuser* utility shall write diagnostic messages to standard error.

91267 The *fuser* utility also shall write the following to standard error:

91268 • The pathname of each named file is written followed immediately by a <colon>.

91269 • For each process ID written to standard output, the character 'c' shall be written to

91270 standard error if the process is using the file as its current directory and the character 'r'

91271 shall be written to standard error if the process is using the file as its root directory.

91272 Implementations may write other alphabetic characters to indicate other uses of files.

91273 • When the *-u* option is specified, characters indicating the use of the file shall be followed

91274 immediately by the user name, in parentheses, corresponding to the real user ID of the

91275 process. If the user name cannot be resolved from the real user ID of the process, the real

91276 user ID of the process shall be written instead of the user name.

91277 When standard output and standard error are directed to the same file, the output shall be

91278 interleaved so that the filename appears at the start of each line, followed by the process ID and

91279 characters indicating the use of the file. Then, if the *-u* option is specified, the user name or user

91280 ID for each process using that file shall be written.

91281 A <newline> shall be written to standard error after the last output described above for each *file*

91282 operand.

91283 **OUTPUT FILES**

91284 None.

91285 **EXTENDED DESCRIPTION**

91286 None.

91287 **EXIT STATUS**

91288 The following exit values shall be returned:

91289 0 Successful completion.

91290 >0 An error occurred.

91291 **CONSEQUENCES OF ERRORS**

91292 Default.

**APPLICATION USAGE**

None.

**EXAMPLES**

The command:

```
fuser -fu .
```

writes to standard output the process IDs of processes that are using the current directory and writes to standard error an indication of how those processes are using the directory and the user names associated with the processes that are using the current directory.

```
fuser -c <mount point>
```

writes to standard output the process IDs of processes that are using any file in the file system which is mounted on <mount point> and writes to standard error an indication of how those processes are using the files.

```
fuser <mount point>
```

writes to standard output the process IDs of processes that are using the file which is named by <mount point> and writes to standard error an indication of how those processes are using the file.

```
fuser <block device>
```

writes to standard output the process IDs of processes that are using any file which is on the device named by <block device> and writes to standard error an indication of how those processes are using the file.

```
fuser -f <block device>
```

writes to standard output the process IDs of processes that are using the file <block device> itself and writes to standard error an indication of how those processes are using the file.

**RATIONALE**

The definition of the *fuser* utility follows existing practice.

**FUTURE DIRECTIONS**

None.

**SEE ALSO**

XBD [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 215)

**CHANGE HISTORY**

First released in Issue 5.

**Issue 7**

SD5-XCU-ERN-90 is applied, updating the EXAMPLES section.

SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

91327 **NAME**

91328       gencat — generate a formatted message catalog

91329 **SYNOPSIS**91330       gencat *catfile* *msgfile*...91331 **DESCRIPTION**

91332       The *gencat* utility shall merge the message text source file *msgfile* into a formatted message  
 91333       catalog *catfile*. The file *catfile* shall be created if it does not already exist. If *catfile* does exist, its  
 91334       messages shall be included in the new *catfile*. If set and message numbers collide, the new  
 91335       message text defined in *msgfile* shall replace the old message text currently contained in *catfile*.

91336 **OPTIONS**

91337       None.

91338 **OPERANDS**

91339       The following operands shall be supported:

91340       *catfile*       A pathname of the formatted message catalog. If '-' is specified, standard output  
 91341       shall be used. The format of the message catalog produced is unspecified.

91342       *msgfile*       A pathname of a message text source file. If '-' is specified for an instance of  
 91343       *msgfile*, standard input shall be used. The format of message text source files is  
 91344       defined in the EXTENDED DESCRIPTION section.

91345 **STDIN**91346       The standard input shall not be used unless a *msgfile* operand is specified as '-'.91347 **INPUT FILES**

91348       The input files shall be text files.

91349 **ENVIRONMENT VARIABLES**91350       The following environment variables shall affect the execution of *gencat*:

91351       *LANG*       Provide a default value for the internationalization variables that are unset or null.  
 91352       (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 91353       variables used to determine the values of locale categories.)

91354       *LC\_ALL*       If set to a non-empty string value, override the values of all the other  
 91355       internationalization variables.

91356       *LC\_CTYPE*   Determine the locale for the interpretation of sequences of bytes of text data as  
 91357       characters (for example, single-byte as opposed to multi-byte characters in  
 91358       arguments and input files).

91359       *LC\_MESSAGES*

91360       Determine the locale that should be used to affect the format and contents of  
 91361       diagnostic messages written to standard error.

91362       *NLSPATH*   Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

91363 **ASYNCHRONOUS EVENTS**

91364       Default.

91365 **STDOUT**91366       The standard output shall not be used unless the *catfile* operand is specified as '-'.

91367 **STDERR**

91368 The standard error shall be used only for diagnostic messages.

91369 **OUTPUT FILES**

91370 None.

91371 **EXTENDED DESCRIPTION**

91372 The content of a message text file shall be in the format defined as follows. Note that the fields of  
 91373 a message text source line are separated by a single <blank> character. Any other <blank>  
 91374 characters are considered to be part of the subsequent field.

91375 **\$set** *n comment*

91376 This line specifies the set identifier of the following messages until the next **\$set** or  
 91377 end-of-file appears. The *n* denotes the set identifier, which is defined as a number  
 91378 in the range [1, {NL\_SETMAX}] (see the <limits.h> header defined in the Base  
 91379 Definitions volume of POSIX.1-2008). The application shall ensure that set  
 91380 identifiers are presented in ascending order within a single source file, but need  
 91381 not be contiguous. Any string following the set identifier shall be treated as a  
 91382 comment. If no **\$set** directive is specified in a message text source file, all messages  
 91383 shall be located in an implementation-defined default message set NL\_SETD (see  
 91384 the <nl\_types.h> header defined in the Base Definitions volume of POSIX.1-2008).

91385 **\$delset** *n comment*

91386 This line deletes message set *n* from an existing message catalog. The *n* denotes the  
 91387 set number [1, {NL\_SETMAX}]. Any string following the set number shall be  
 91388 treated as a comment.

91389 **\$ comment** A line beginning with ' \$ ' followed by a <blank> shall be treated as a comment.

91390 *m message-text*

91391 The *m* denotes the message identifier, which is defined as a number in the range [1,  
 91392 {NL\_MSGMAX}] (see the <limits.h> header). The *message-text* shall be stored in the  
 91393 message catalog with the set identifier specified by the last **\$set** directive, and with  
 91394 message identifier *m*. If the *message-text* is empty, and a <blank> field separator is  
 91395 present, an empty string shall be stored in the message catalog. If a message source  
 91396 line has a message number, but neither a field separator nor *message-text*, the  
 91397 existing message with that number (if any) shall be deleted from the catalog. The  
 91398 application shall ensure that message identifiers are in ascending order within a  
 91399 single set, but need not be contiguous. The application shall ensure that the length  
 91400 of *message-text* is in the range [0, {NL\_TEXTMAX}] (see the <limits.h> header).

91401 **\$quote** *n* This line specifies an optional quote character *c*, which can be used to surround  
 91402 *message-text* so that trailing <space> characters or null (empty) messages are visible  
 91403 in a message source line. By default, or if an empty **\$quote** directive is supplied, no  
 91404 quoting of *message-text* shall be recognized.

91405 Empty lines in a message text source file shall be ignored. The effects of lines starting with any  
 91406 character other than those defined above are implementation-defined.

91407 Text strings can contain the special characters and escape sequences defined in the following  
 91408 table:

91409  
91410  
91411  
91412  
91413  
91414  
91415  
91416  
91417  
  
91418  
91419  
91420  
  
91421  
91422  
  
91423  
91424  
  
91425  
  
91426  
  
91427  
91428  
91429  
91430  
  
91431  
91432  
  
91433  
91434  
91435  
91436  
  
91437  
91438  
  
91439  
91440  
  
91441  
91442  
  
91443  
91444  
  
91445  
  
91446  
91447  
  
91448  
91449

| Description       | Symbol | Sequence |
|-------------------|--------|----------|
| <newline>         | NL(LF) | \n       |
| Horizontal-tab    | HT     | \t       |
| <vertical-tab>    | VT     | \v       |
| <backspace>       | BS     | \b       |
| <carriage-return> | CR     | \r       |
| <form-feed>       | FF     | \f       |
| Backslash         | \      | \\       |
| Bit pattern       | ddd    | \ddd     |

The escape sequence "\ddd" consists of <backslash> followed by one, two, or three octal digits, which shall be taken to specify the value of the desired character. If the character following a <backslash> is not one of those specified, the <backslash> shall be ignored.

A <backslash> followed by a <newline> is also used to continue a string on the following line. Thus, the following two lines describe a single message string:

```
1 This line continues \
to the next line
```

which shall be equivalent to:

```
1 This line continues to the next line
```

**EXIT STATUS**

The following exit values shall be returned:

- 0 Successful completion.
- >0 An error occurred.

**CONSEQUENCES OF ERRORS**

Default.

**APPLICATION USAGE**

Message catalogs produced by *gencat* are binary encoded, meaning that their portability cannot be guaranteed between different types of machine. Thus, just as C programs need to be recompiled for each type of machine, so message catalogs must be recreated via *gencat*.

**EXAMPLES**

None.

**RATIONALE**

None.

**FUTURE DIRECTIONS**

None.

**SEE ALSO**

*iconv*

XBD [Chapter 8](#) (on page 173), [<limits.h>](#), [<nl\\_types.h>](#)

**CHANGE HISTORY**

First released in Issue 3.

**Issue 6**

The normative text is reworded to avoid use of the term “must” for application requirements.



91450 **Issue 7**  
91451

The *gencat* utility is moved from the XSI option to the Base.

## 91452 NAME

91453 get — get a version of an SCCS file (**DEVELOPMENT**)

## 91454 SYNOPSIS

91455 xsi `get [-begkmnlLpst] [-c cutoff] [-i list] [-r SID] [-x list] file...`

## 91456 DESCRIPTION

91457 The *get* utility shall generate a text file from each named SCCS *file* according to the specifications  
91458 given by its options.91459 The generated text shall normally be written into a file called the **g-file** whose name is derived  
91460 from the SCCS filename by simply removing the leading "s.". 

## 91461 OPTIONS

91462 The *get* utility shall conform to XBD [Section 12.2](#) (on page 215).

91463 The following options shall be supported:

91464 **-r *SID*** Indicate the SCCS Identification String (SID) of the version (delta) of an SCCS file  
91465 to be retrieved. The table shows, for the most useful cases, what version of an  
91466 SCCS file is retrieved (as well as the SID of the version to be eventually created by  
91467 *delta* if the **-e** option is also used), as a function of the SID specified.91468 **-c *cutoff*** Indicate the *cutoff* date-time, in the form:91469 `YY[MM[DD[HH[MM[SS]]]]]`91470 For the YY component, values in the range [69,99] shall refer to years 1969 to 1999  
91471 inclusive, and values in the range [00,68] shall refer to years 2000 to 2068 inclusive.91472 **Note:** It is expected that in a future version of this standard the default century inferred  
91473 from a 2-digit year will change. (This would apply to all commands accepting a  
91474 2-digit year as input.)91475 No changes (deltas) to the SCCS file that were created after the specified *cutoff*  
91476 date-time shall be included in the generated text file. Units omitted from the date-  
91477 time default to their maximum possible values; for example, **-c 7502** is equivalent  
91478 to **-c 750228235959**.91479 Any number of non-numeric characters may separate the various 2-digit pieces of  
91480 the *cutoff* date-time. This feature allows the user to specify a *cutoff* date in the form:  
91481 **-c "77/2/2 9:22:25"**.91482 **-e** Indicate that the *get* is for the purpose of editing or making a change (delta) to the  
91483 SCCS file via a subsequent use of *delta*. The **-e** option used in a *get* for a particular  
91484 version (SID) of the SCCS file shall prevent further *get* commands from editing on  
91485 the same SID until *delta* is executed or the **j** (joint edit) flag is set in the SCCS file.  
91486 Concurrent use of *get -e* for different SIDs is always allowed.91487 If the **g-file** generated by *get* with a **-e** option is accidentally ruined in the process  
91488 of editing, it may be regenerated by re-executing the *get* command with the **-k**  
91489 option in place of the **-e** option.91490 SCCS file protection specified via the ceiling, floor, and authorized user list stored  
91491 in the SCCS file shall be enforced when the **-e** option is used.91492 **-b** Use with the **-e** option to indicate that the new delta should have an SID in a new  
91493 branch as shown in the table below. This option shall be ignored if the **b** flag is not  
91494 present in the file or if the retrieved delta is not a leaf delta. (A leaf delta is one that  
91495 has no successors on the SCCS file tree.)

|       |                |                                                                                                                                                                        |
|-------|----------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 91496 |                | <b>Note:</b> A branch delta may always be created from a non-leaf delta.                                                                                               |
| 91497 | <b>-i list</b> | Indicate a <i>list</i> of deltas to be included (forced to be applied) in the creation of the generated file. The <i>list</i> has the following syntax:                |
| 91498 |                |                                                                                                                                                                        |
| 91499 |                | <code>&lt;list&gt; ::= &lt;range&gt;   &lt;list&gt; , &lt;range&gt;</code>                                                                                             |
| 91500 |                | <code>&lt;range&gt; ::= SID   SID - SID</code>                                                                                                                         |
| 91501 |                | SID, the SCCS Identification of a delta, may be in any form shown in the "SID Specified" column of the table in the EXTENDED DESCRIPTION section, except               |
| 91502 |                | that the result of supplying a partial SID is unspecified. A diagnostic message shall                                                                                  |
| 91503 |                | be written if the first SID in the range is not an ancestor of the second SID in the                                                                                   |
| 91504 |                | range.                                                                                                                                                                 |
| 91505 |                |                                                                                                                                                                        |
| 91506 | <b>-x list</b> | Indicate a <i>list</i> of deltas to be excluded (forced not to be applied) in the creation of the generated file. See the <b>-i</b> option for the <i>list</i> format. |
| 91507 |                |                                                                                                                                                                        |
| 91508 | <b>-k</b>      | Suppress replacement of identification keywords (see below) in the retrieved text by their value. The <b>-k</b> option shall be implied by the <b>-e</b> option.       |
| 91509 |                |                                                                                                                                                                        |
| 91510 | <b>-l</b>      | Write a delta summary into an <b>l-file</b> .                                                                                                                          |
| 91511 | <b>-L</b>      | Write a delta summary to standard output. All informative output that normally is                                                                                      |
| 91512 |                | written to standard output shall be written to standard error instead, unless the <b>-s</b>                                                                            |
| 91513 |                | option is used, in which case it shall be suppressed.                                                                                                                  |
| 91514 | <b>-p</b>      | Write the text retrieved from the SCCS file to the standard output. No <b>g-file</b> shall                                                                             |
| 91515 |                | be created. All informative output that normally goes to the standard output shall                                                                                     |
| 91516 |                | go to standard error instead, unless the <b>-s</b> option is used, in which case it shall                                                                              |
| 91517 |                | disappear.                                                                                                                                                             |
| 91518 | <b>-s</b>      | Suppress all informative output normally written to standard output. However,                                                                                          |
| 91519 |                | fatal error messages (which shall always be written to the standard error) shall                                                                                       |
| 91520 |                | remain unaffected.                                                                                                                                                     |
| 91521 | <b>-m</b>      | Precede each text line retrieved from the SCCS file by the SID of the delta that                                                                                       |
| 91522 |                | inserted the text line in the SCCS file. The format shall be:                                                                                                          |
| 91523 |                | <code>"%s\t%s", &lt;SID&gt;, &lt;text line&gt;</code>                                                                                                                  |
| 91524 | <b>-n</b>      | Precede each generated text line with the <b>%M%</b> identification keyword value (see                                                                                 |
| 91525 |                | below). The format shall be:                                                                                                                                           |
| 91526 |                | <code>"%s\t%s", &lt;%M% value&gt;, &lt;text line&gt;</code>                                                                                                            |
| 91527 |                | When both the <b>-m</b> and <b>-n</b> options are used, the <code>&lt;text line&gt;</code> shall be replaced by the                                                    |
| 91528 |                | <b>-m</b> option-generated format.                                                                                                                                     |
| 91529 | <b>-g</b>      | Suppress the actual retrieval of text from the SCCS file. It is primarily used to                                                                                      |
| 91530 |                | generate an <b>l-file</b> , or to verify the existence of a particular SID.                                                                                            |
| 91531 | <b>-t</b>      | Use to access the most recently created (top) delta in a given release (for example,                                                                                   |
| 91532 |                | <b>-r 1</b> ), or release and level (for example, <b>-r 1.2</b> ).                                                                                                     |

91533 **OPERANDS**

91534 The following operands shall be supported:

91535 *file* A pathname of an existing SCCS file or a directory. If *file* is a directory, the *get*  
 91536 utility shall behave as though each file in the directory were specified as a named  
 91537 file, except that non-SCCS files (last component of the pathname does not begin  
 91538 with **s**.) and unreadable files shall be silently ignored.

91539 If exactly one *file* operand appears, and it is **'-'**, the standard input shall be read;  
 91540 each line of the standard input is taken to be the name of an SCCS file to be  
 91541 processed. Non-SCCS files and unreadable files shall be silently ignored.

91542 **STDIN**

91543 The standard input shall be a text file used only if the *file* operand is specified as **'-'**. Each line  
 91544 of the text file shall be interpreted as an SCCS pathname.

91545 **INPUT FILES**

91546 The SCCS files shall be files of an unspecified format.

91547 **ENVIRONMENT VARIABLES**91548 The following environment variables shall affect the execution of *get*:

91549 *LANG* Provide a default value for the internationalization variables that are unset or null.  
 91550 (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 91551 variables used to determine the values of locale categories.)

91552 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
 91553 internationalization variables.

91554 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
 91555 characters (for example, single-byte as opposed to multi-byte characters in  
 91556 arguments and input files).

91557 *LC\_MESSAGES*

91558 Determine the locale that should be used to affect the format and contents of  
 91559 diagnostic messages written to standard error, and informative messages written  
 91560 to standard output (or standard error, if the **-p** option is used).

91561 *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

91562 *TZ* Determine the timezone in which the times and dates written in the SCCS file are  
 91563 evaluated. If the *TZ* variable is unset or NULL, an unspecified system default  
 91564 timezone is used.

91565 **ASYNCHRONOUS EVENTS**

91566 Default.

91567 **STDOUT**

91568 For each file processed, *get* shall write to standard output the SID being accessed and the  
 91569 number of lines retrieved from the SCCS file, in the following format:

91570 "%s\n%d lines\n", &lt;SID&gt;, &lt;number of lines&gt;

91571 If the **-e** option is used, the SID of the delta to be made shall appear after the SID accessed and  
 91572 before the number of lines generated, in the POSIX locale:

91573 "%s\nnew delta %s\n%d lines\n", <SID accessed>,  
 91574 <SID to be made>, <number of lines>

91575 If there is more than one named file or if a directory or standard input is named, each pathname

91576 shall be written before each of the lines shown in one of the preceding formats:

91577 `"\n%s:\n", <pathname>`

91578 If the `-L` option is used, a delta summary shall be written following the format specified below  
91579 for **l-files**.

91580 If the `-i` option is used, included deltas shall be listed following the notation, in the POSIX  
91581 locale:

91582 `"Included:\n"`

91583 If the `-x` option is used, excluded deltas shall be listed following the notation, in the POSIX  
91584 locale:

91585 `"Excluded:\n"`

91586 If the `-p` or `-L` options are specified, the standard output shall consist of the text retrieved from  
91587 the SCCS file.

## 91588 STDERR

91589 The standard error shall be used only for diagnostic messages, except if the `-p` or `-L` options are  
91590 specified, it shall include all informative messages normally sent to standard output.

## 91591 OUTPUT FILES

91592 Several auxiliary files may be created by *get*. These files are known generically as the **g-file**, **l-**  
91593 **file**, **p-file**, and **z-file**. The letter before the <hyphen> is called the *tag*. An auxiliary filename  
91594 shall be formed from the SCCS filename: the application shall ensure that the last component of  
91595 all SCCS filenames is of the form *s.module-name*; the auxiliary files shall be named by replacing  
91596 the leading *s* with the tag. The **g-file** shall be an exception to this scheme: the **g-file** is named by  
91597 removing the *s.* prefix. For example, for *s.xyz.c*, the auxiliary filenames would be *xyz.c*, *l.xyz.c*,  
91598 *p.xyz.c*, and *z.xyz.c*, respectively.

91599 The **g-file**, which contains the generated text, shall be created in the current directory (unless the  
91600 `-p` option is used). A **g-file** shall be created in all cases, whether or not any lines of text were  
91601 generated by the *get*. It shall be owned by the real user. If the `-k` option is used or implied, the  
91602 **g-file** shall be writable by the owner only (read-only for everyone else); otherwise, it shall be  
91603 read-only. Only the real user need have write permission in the current directory.

91604 The **l-file** shall contain a table showing which deltas were applied in generating the retrieved  
91605 text. The **l-file** shall be created in the current directory if the `-l` option is used; it shall be read-  
91606 only and it is owned by the real user. Only the real user need have write permission in the  
91607 current directory.

91608 Lines in the **l-file** shall have the following format:

91609 `"%c%c%cΔ%s\t%sΔ%s\n", <code1>, <code2>, <code3>,  
91610 <SID>, <date-time>, <login>`

91611 where the entries are:

91612 `<code1>` A <space> if the delta was applied; ' \* ' otherwise.

91613 `<code2>` A <space> if the delta was applied or was not applied and ignored; ' \* ' if the delta  
91614 was not applied and was not ignored.

91615 `<code3>` A character indicating a special reason why the delta was or was not applied:

91616 **I** Included.

91617                   X   Excluded.

91618                   C   Cut off (by a `-c` option).

91619       <date-time>   Date and time (using the format of the *date* utility's `%Y/%m/%d %T` conversion  
91620                   specification format) of creation.

91621       <login>       Login name of person who created *delta*.

91622       The comments and MR data shall follow on subsequent lines, indented one <tab>. A blank line  
91623       shall terminate each entry.

91624       The **p-file** shall be used to pass information resulting from a *get* with a `-e` option along to *delta*.  
91625       Its contents shall also be used to prevent a subsequent execution of *get* with a `-e` option for the  
91626       same SID until *delta* is executed or the joint edit flag, *j*, is set in the SCCS file. The **p-file** shall be  
91627       created in the directory containing the SCCS file and the application shall ensure that the  
91628       effective user has write permission in that directory. It shall be writable by owner only, and  
91629       owned by the effective user. Each line in the **p-file** shall have the following format:

91630       "%sΔ%sΔ%sΔ%s%s\n", <g-file SID>,  
91631                <SID of new delta>, <login-name of real user>,  
91632                <date-time>, <i-value>, <x-value>

91633       where <i-value> uses the format " " if no `-i` option was specified, and shall use the format:

91634       "Δ-i%s", <-i option option-argument>

91635       if a `-i` option was specified and <x-value> uses the format " " if no `-x` option was specified, and  
91636       shall use the format:

91637       "Δ-x%s", <-x option option-argument>

91638       if a `-x` option was specified. There can be an arbitrary number of lines in the **p-file** at any time;  
91639       no two lines shall have the same new delta SID.

91640       The **z-file** shall serve as a lock-out mechanism against simultaneous updates. Its contents shall  
91641       be the binary process ID of the command (that is, *get*) that created it. The **z-file** shall be created  
91642       in the directory containing the SCCS file for the duration of *get*. The same protection restrictions  
91643       as those for the **p-file** shall apply for the **z-file**. The **z-file** shall be created read-only.

## 91644 EXTENDED DESCRIPTION

| Determination of SCCS Identification String |                    |                                             |               |                            |  |
|---------------------------------------------|--------------------|---------------------------------------------|---------------|----------------------------|--|
| SID* Specified                              | -b Keyletter Used† | Other Conditions                            | SID Retrieved | SID of Delta to be Created |  |
| none‡                                       | no                 | R defaults to mR                            | mR.mL         | mR.(mL+1)                  |  |
| none‡                                       | yes                | R defaults to mR                            | mR.mL         | mR.mL.(mB+1).1             |  |
| R                                           | no                 | R > mR                                      | mR.mL         | R.1***                     |  |
| R                                           | no                 | R = mR                                      | mR.mL         | mR.(mL+1)                  |  |
| R                                           | yes                | R > mR                                      | mR.mL         | mR.mL.(mB+1).1             |  |
| R                                           | yes                | R = mR                                      | mR.mL         | mR.mL.(mB+1).1             |  |
| R                                           | –                  | R < mR and R does not exist                 | hR.mL**       | hR.mL.(mB+1).1             |  |
| R                                           | –                  | Trunk successor in release > R and R exists | R.mL          | R.mL.(mB+1).1              |  |
| R.L                                         | no                 | No trunk successor                          | R.L           | R.(L+1)                    |  |
| R.L                                         | yes                | No trunk successor                          | R.L           | R.L.(mB+1).1               |  |
| R.L                                         | –                  | Trunk successor in release ≥ R              | R.L           | R.L.(mB+1).1               |  |
| R.L.B                                       | no                 | No branch successor                         | R.L.B.mS      | R.L.B.(mS+1)               |  |
| R.L.B                                       | yes                | No branch successor                         | R.L.B.mS      | R.L.(mB+1).1               |  |
| R.L.B.S                                     | no                 | No branch successor                         | R.L.B.S       | R.L.B.(S+1)                |  |
| R.L.B.S                                     | yes                | No branch successor                         | R.L.B.S       | R.L.(mB+1).1               |  |
| R.L.B.S                                     | –                  | Branch successor                            | R.L.B.S       | R.L.(mB+1).1               |  |

91667 \* R, L, B, and S are the release, level, branch, and sequence components of the SID, respectively; m means maximum. Thus, for example, R.mL means “the maximum level number within release R”; R.L.(mB+1).1 means “the first sequence number on the new branch (that is, maximum branch number plus one) of level L within release R”. Note that if the SID specified is of the form R.L, R.L.B, or R.L.B.S, each of the specified components shall exist.

91673 \*\* hR is the highest existing release that is lower than the specified, nonexistent, release R.

91674 \*\*\* This is used to force creation of the first delta in a new release.

91675 † The -b option is effective only if the b flag is present in the file. An entry of ‘–’ means “irrelevant”.

91677 ‡ This case applies if the d (default SID) flag is not present in the file. If the d flag is present in the file, then the SID obtained from the d flag is interpreted as if it had been specified on the command line. Thus, one of the other cases in this table applies.

## 91680 System Date and Time

91681 When a **g-file** is generated, the creation time of deltas in the SCCS file may be taken into  
 91682 account. If any of these times are apparently in the future, the behavior is unspecified.

## 91683 Identification Keywords

91684 Identifying information shall be inserted into the text retrieved from the SCCS file by replacing  
 91685 identification keywords with their value wherever they occur. The following keywords may be  
 91686 used in the text stored in an SCCS file:

|       |     |                                                                                               |
|-------|-----|-----------------------------------------------------------------------------------------------|
| 91687 | %M% | Module name: either the value of the <b>m</b> flag in the file, or if absent, the name of the |
| 91688 |     | SCCS file with the leading <b>s.</b> removed.                                                 |
| 91689 | %I% | SCCS identification (SID) (%R%.%L% or %R%.%L%.%B%.%S%) of the retrieved                       |
| 91690 |     | text.                                                                                         |
| 91691 | %R% | Release.                                                                                      |
| 91692 | %L% | Level.                                                                                        |
| 91693 | %B% | Branch.                                                                                       |
| 91694 | %S% | Sequence.                                                                                     |
| 91695 | %D% | Current date (YY/MM/DD).                                                                      |
| 91696 | %H% | Current date (MM/DD/YY).                                                                      |
| 91697 | %T% | Current time (HH:MM:SS).                                                                      |
| 91698 | %E% | Date newest applied delta was created (YY/MM/DD).                                             |
| 91699 | %G% | Date newest applied delta was created (MM/DD/YY).                                             |
| 91700 | %U% | Time newest applied delta was created (HH:MM:SS).                                             |
| 91701 | %Y% | Module type: value of the <b>t</b> flag in the SCCS file.                                     |
| 91702 | %F% | SCCS filename.                                                                                |
| 91703 | %P% | SCCS absolute pathname.                                                                       |
| 91704 | %Q% | The value of the <b>q</b> flag in the file.                                                   |
| 91705 | %C% | Current line number. This keyword is intended for identifying messages output by              |
| 91706 |     | the program, such as “this should not have happened” type errors. It is not                   |
| 91707 |     | intended to be used on every line to provide sequence numbers.                                |
| 91708 | %Z% | The four-character string "@ (# )" recognizable by <i>what</i> .                              |
| 91709 | %W% | A shorthand notation for constructing <i>what</i> strings:                                    |
| 91710 |     | %W%=%Z%%M%<tab>%I%                                                                            |
| 91711 | %A% | Another shorthand notation for constructing <i>what</i> strings:                              |
| 91712 |     | %A%=%Z%%Y%%M%%I%%Z%                                                                           |

## 91713 EXIT STATUS

91714 The following exit values shall be returned:

91715 0 Successful completion.



91716 >0 An error occurred.

## 91717 CONSEQUENCES OF ERRORS

91718 Default.

## 91719 APPLICATION USAGE

91720 Problems can arise if the system date and time have been modified (for example, put forward  
91721 and then back again, or unsynchronized clocks across a network) and can also arise when  
91722 different values of the *TZ* environment variable are used.

91723 Problems of a similar nature can also arise for the operation of the *delta* utility, which compares  
91724 the previous file body against the working file as part of its normal operation.

## 91725 EXAMPLES

91726 None.

## 91727 RATIONALE

91728 None.

## 91729 FUTURE DIRECTIONS

91730 None.

## 91731 SEE ALSO

91732 *admin, delta, prs, what*

91733 XBD [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 215)

## 91734 CHANGE HISTORY

91735 First released in Issue 2.

### 91736 Issue 5

91737 A correction is made to the first format string in STDOUT.

91738 The interpretation of the *YY* component of the *-c cutoff* argument is noted.

### 91739 Issue 6

91740 The obsolescent SYNOPSIS is removed, removing the *-lp* option.

91741 The normative text is reworded to avoid use of the term “must” for application requirements.

91742 The Open Group Corrigendum U025/5 is applied, correcting text in the OPTIONS section.

91743 The Open Group Corrigendum U048/1 is applied.

91744 The Open Group Interpretation PIN4C.00014 is applied.

91745 The Open Group Base Resolution bwg2001-007 is applied as follows:

- 91746 • The EXTENDED DESCRIPTION section is updated to make partial SID handling
- 91747 unspecified, reflecting common usage, and to clarify SID ranges.
- 91748 • New text is added to the EXTENDED DESCRIPTION and APPLICATION USAGE sections
- 91749 regarding how the system date and time may be taken into account.
- 91750 • The *TZ* environment variable is added to the ENVIRONMENT VARIABLES section.

### 91751 Issue 7

91752 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

**NAME**

getconf — get configuration values

**SYNOPSIS**

getconf [-v *specification*] *system\_var*

getconf [-v *specification*] *path\_var pathname*

**DESCRIPTION**

In the first synopsis form, the *getconf* utility shall write to the standard output the value of the variable specified by the *system\_var* operand.

In the second synopsis form, the *getconf* utility shall write to the standard output the value of the variable specified by the *path\_var* operand for the path specified by the *pathname* operand.

The value of each configuration variable shall be determined as if it were obtained by calling the function from which it is defined to be available by this volume of POSIX.1-2008 or by the System Interfaces volume of POSIX.1-2008 (see the OPERANDS section). The value shall reflect conditions in the current operating environment.

**OPTIONS**

The *getconf* utility shall conform to XBD [Section 12.2](#) (on page 215).

The following option shall be supported:

**-v *specification***

Indicate a specific specification and version for which configuration variables shall be determined. If this option is not specified, the values returned correspond to an implementation default conforming compilation environment.

If the command:

```
getconf _POSIX_V7_ILP32_OFF32
```

does not write "-1\n" or "undefined\n" to standard output, then commands of the form:

```
getconf -v POSIX_V7_ILP32_OFF32 ...
```

determine values for configuration variables corresponding to the POSIX\_V7\_ILP32\_OFF32 compilation environment specified in [c99](#), the EXTENDED DESCRIPTION.

If the command:

```
getconf _POSIX_V7_ILP32_OFFBIG
```

does not write "-1\n" or "undefined\n" to standard output, then commands of the form:

```
getconf -v POSIX_V7_ILP32_OFFBIG ...
```

determine values for configuration variables corresponding to the POSIX\_V7\_ILP32\_OFFBIG compilation environment specified in [c99](#), the EXTENDED DESCRIPTION.

If the command:

```
getconf _POSIX_V7_LP64_OFF64
```

does not write "-1\n" or "undefined\n" to standard output, then commands of the form:

91794 `getconf -v POSIX_V7_LP64_OFF64 ...`  
 91795 determine values for configuration variables corresponding to the  
 91796 POSIX\_V7\_LP64\_OFF64 compilation environment specified in [c99](#), the  
 91797 EXTENDED DESCRIPTION.

91798 If the command:

91799 `getconf _POSIX_V7_LPBIG_OFFBIG`

91800 does not write "-1\n" or "undefined\n" to standard output, then commands of  
 91801 the form:

91802 `getconf -v POSIX_V7_LPBIG_OFFBIG ...`

91803 determine values for configuration variables corresponding to the  
 91804 POSIX\_V7\_LPBIG\_OFFBIG compilation environment specified in [c99](#), the  
 91805 EXTENDED DESCRIPTION.

## 91806 OPERANDS

91807 The following operands shall be supported:

91808 *path\_var* A name of a configuration variable. All of the variables in the Variable column of  
 91809 the table in the DESCRIPTION of the *fpathconf()* function defined in the System  
 91810 Interfaces volume of POSIX.1-2008, without the enclosing braces, shall be  
 91811 supported. The implementation may add other local variables.

91812 *pathname* A pathname for which the variable specified by *path\_var* is to be determined.

91813 *system\_var* A name of a configuration variable. All of the following variables shall be  
 91814 supported:

- 91815 • The names in the Variable column of the table in the DESCRIPTION of the
- 91816 *sysconf()* function in the System Interfaces volume of POSIX.1-2008, except
- 91817 for the entries corresponding to `_SC_CLK_TCK`, `_SC_GETGR_R_SIZE_MAX`,
- 91818 and `_SC_GETPW_R_SIZE_MAX`, without the enclosing braces.

91819 For compatibility with earlier versions, the following variable names shall  
 91820 also be supported:

91821 `POSIX2_C_BIND`  
 91822 `POSIX2_C_DEV`  
 91823 `POSIX2_CHAR_TERM`  
 91824 `POSIX2_FORT_DEV`  
 91825 `POSIX2_FORT_RUN`  
 91826 `POSIX2_LOCALEDEF`  
 91827 `POSIX2_SW_DEV`  
 91828 `POSIX2_UPE`  
 91829 `POSIX2_VERSION`

91830 and shall be equivalent to the same name prefixed with an <underscore>.  
 91831 This requirement may be removed in a future version.

- 91832 • The names of the symbolic constants used as the *name* argument of the
- 91833 *confstr()* function in the System Interfaces volume of POSIX.1-2008, without
- 91834 the `_CS_` prefix.

91835 • The names of the symbolic constants listed under the headings “Maximum  
 91836 Values” and “Minimum Values” in the description of the `<limits.h>` header  
 91837 in the Base Definitions volume of POSIX.1-2008, without the enclosing  
 91838 braces.

91839 For compatibility with earlier versions, the following variable names shall  
 91840 also be supported:

91841 POSIX2\_BC\_BASE\_MAX  
 91842 POSIX2\_BC\_DIM\_MAX  
 91843 POSIX2\_BC\_SCALE\_MAX  
 91844 POSIX2\_BC\_STRING\_MAX  
 91845 POSIX2\_COLL\_WEIGHTS\_MAX  
 91846 POSIX2\_EXPR\_NEST\_MAX  
 91847 POSIX2\_LINE\_MAX  
 91848 POSIX2\_RE\_DUP\_MAX

91849 and shall be equivalent to the same name prefixed with an `<underscore>`.  
 91850 This requirement may be removed in a future version.

91851 The implementation may add other local values.

## 91852 STDIN

91853 Not used.

## 91854 INPUT FILES

91855 None.

## 91856 ENVIRONMENT VARIABLES

91857 The following environment variables shall affect the execution of *getconf*:

91858 *LANG* Provide a default value for the internationalization variables that are unset or null.  
 91859 (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 91860 variables used to determine the values of locale categories.)

91861 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
 91862 internationalization variables.

91863 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
 91864 characters (for example, single-byte as opposed to multi-byte characters in  
 91865 arguments).

91866 *LC\_MESSAGES*

91867 Determine the locale that should be used to affect the format and contents of  
 91868 diagnostic messages written to standard error.

91869 XSI *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

## 91870 ASYNCHRONOUS EVENTS

91871 Default.

## 91872 STDOUT

91873 If the specified variable is defined on the system and its value is described to be available from  
 91874 the *confstr()* function defined in the System Interfaces volume of POSIX.1-2008, its value shall be  
 91875 written in the following format:

91876 "%s\n", *<value>*

91877 Otherwise, if the specified variable is defined on the system, its value shall be written in the

91878 following format:

91879 "%d\n", <value>

91880 If the specified variable is valid, but is undefined on the system, *getconf* shall write using the

91881 following format:

91882 "undefined\n"

91883 If the variable name is invalid or an error occurs, nothing shall be written to standard output.

91884 **STDERR**

91885 The standard error shall be used only for diagnostic messages.

91886 **OUTPUT FILES**

91887 None.

91888 **EXTENDED DESCRIPTION**

91889 None.

91890 **EXIT STATUS**

91891 The following exit values shall be returned:

91892 0 The specified variable is valid and information about its current state was written

91893 successfully.

91894 >0 An error occurred.

91895 **CONSEQUENCES OF ERRORS**

91896 Default.

91897 **APPLICATION USAGE**

91898 None.

91899 **EXAMPLES**

91900 The following example illustrates the value of {NGROUPS\_MAX}:

91901 `getconf NGROUPS_MAX`

91902 The following example illustrates the value of {NAME\_MAX} for a specific directory:

91903 `getconf NAME_MAX /usr`

91904 The following example shows how to deal more carefully with results that might be unspecified:

91905 `if value=$(getconf PATH_MAX /usr); then`

91906 `if [ "$value" = "undefined" ]; then`

91907 `echo PATH_MAX in /usr is indeterminate.`

91908 `else`

91909 `echo PATH_MAX in /usr is $value.`

91910 `fi`

91911 `else`

91912 `echo Error in getconf.`

91913 `fi`

91914 **RATIONALE**

91915 The original need for this utility, and for the *confstr()* function, was to provide a way of finding

91916 the configuration-defined default value for the *PATH* environment variable. Since *PATH* can be

91917 modified by the user to include directories that could contain utilities replacing the standard

91918 utilities, shell scripts need a way to determine the system-supplied *PATH* environment variable

91919 value that contains the correct search path for the standard utilities. It was later suggested that

access to the other variables described in this volume of POSIX.1-2008 could also be useful to applications.

This functionality of *getconf* would not be adequately subsumed by another command such as:

```
grep var /etc/conf
```

because such a strategy would provide correct values for neither those variables that can vary at runtime, nor those that can vary depending on the path.

Early proposal versions of *getconf* specified exit status 1 when the specified variable was valid, but not defined on the system. The output string "undefined" is now used to specify this case with exit code 0 because so many things depend on an exit code of zero when an invoked utility is successful.

## FUTURE DIRECTIONS

None.

## SEE ALSO

*c99*

XBD [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 215), [<limits.h>](#)

XSH [confstr\(\)](#), [fpathconf\(\)](#), [sysconf\(\)](#), [system\(\)](#)

## CHANGE HISTORY

First released in Issue 4.

### Issue 5

In the OPERANDS section:

- {NL\_MAX} is changed to {NL\_NMAX}.
- Entries beginning NL\_ are deleted from the list of standard configuration variables.
- The list of variables previously marked UX is merged with the list marked EX.
- Operands are added to support new Option Groups.
- Operands are added so that *getconf* can determine supported programming environments.

### Issue 6

The Open Group Corrigendum U029/4 is applied, correcting the example command in the last paragraph of the OPTIONS section.

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- Operands are added to determine supported programming environments.

This reference page is updated for alignment with the ISO/IEC 9899:1999 standard. Specifically, new macros for *c99* programming environments are introduced.

XSI marked *system\_var* (XBS5\_\*) values are marked LEGACY.

IEEE Std 1003.1-2001/Cor 1-2002, item XCU/TC1/D6/27 is applied, correcting the descriptions of *path\_var* and *system\_var* in the OPERANDS section.

### Issue 7

SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

The EXAMPLES section is corrected.

 91959

POSIX.1-2008, Technical Corrigendum 1, XCU/TC1-2008/0091 [125] is applied.

+ 

91960 **NAME**

91961       getopts — parse utility options

91962 **SYNOPSIS**91963       getopts *optstring name* [*arg...*]91964 **DESCRIPTION**

91965       The *getopts* utility shall retrieve options and option-arguments from a list of parameters. It shall  
 91966       support the Utility Syntax Guidelines 3 to 10, inclusive, described in XBD [Section 12.2](#) (on page  
 91967       215).

91968       Each time it is invoked, the *getopts* utility shall place the value of the next option in the shell  
 91969       variable specified by the *name* operand and the index of the next argument to be processed in the  
 91970       shell variable *OPTIND*. Whenever the shell is invoked, *OPTIND* shall be initialized to 1.

91971       When the option requires an option-argument, the *getopts* utility shall place it in the shell  
 91972       variable *OPTARG*. If no option was found, or if the option that was found does not have an  
 91973       option-argument, *OPTARG* shall be unset.

91974       If an option character not contained in the *optstring* operand is found where an option character  
 91975       is expected, the shell variable specified by *name* shall be set to the <question-mark> ('?')  
 91976       character. In this case, if the first character in *optstring* is a <colon> (':'), the shell variable  
 91977       *OPTARG* shall be set to the option character found, but no output shall be written to standard  
 91978       error; otherwise, the shell variable *OPTARG* shall be unset and a diagnostic message shall be  
 91979       written to standard error. This condition shall be considered to be an error detected in the way  
 91980       arguments were presented to the invoking application, but shall not be an error in *getopts*  
 91981       processing.

91982       If an option-argument is missing:

- 91983       • If the first character of *optstring* is a <colon>, the shell variable specified by *name* shall be  
 91984       set to the <colon> character and the shell variable *OPTARG* shall be set to the option  
 91985       character found.
- 91986       • Otherwise, the shell variable specified by *name* shall be set to the <question-mark>  
 91987       character, the shell variable *OPTARG* shall be unset, and a diagnostic message shall be  
 91988       written to standard error. This condition shall be considered to be an error detected in the  
 91989       way arguments were presented to the invoking application, but shall not be an error in  
 91990       *getopts* processing; a diagnostic message shall be written as stated, but the exit status shall  
 91991       be zero.

91992       When the end of options is encountered, the *getopts* utility shall exit with a return value greater  
 91993       than zero; the shell variable *OPTIND* shall be set to the index of the first operand, or the value  
 91994       "\$#"+1 if there are no operands; the *name* variable shall be set to the <question-mark> character. |  
 91995       Any of the following shall identify the end of options: the first "--" argument that is not an |  
 91996       option-argument, finding an argument that is not an option-argument and does not begin with a  
 91997       '-' , or encountering an error.

91998       The shell variables *OPTIND* and *OPTARG* shall be local to the caller of *getopts* and shall not be  
 91999       exported by default.

92000       The shell variable specified by the *name* operand, *OPTIND*, and *OPTARG* shall affect the current  
 92001       shell execution environment; see [Section 2.12](#) (on page 2353).

92002       If the application sets *OPTIND* to the value 1, a new set of parameters can be used: either the  
 92003       current positional parameters or new *arg* values. Any other attempt to invoke *getopts* multiple  
 92004       times in a single shell execution environment with parameters (positional parameters or *arg*  
 92005       operands) that are not the same in all invocations, or with an *OPTIND* value modified to be a



92006 value other than 1, produces unspecified results.

## 92007 OPTIONS

92008 None.

## 92009 OPERANDS

92010 The following operands shall be supported:

92011 *optstring* A string containing the option characters recognized by the utility invoking *getopts*.  
 92012 If a character is followed by a <colon>, the option shall be expected to have an  
 92013 argument, which should be supplied as a separate argument. Applications should  
 92014 specify an option character and its option-argument as separate arguments, but  
 92015 *getopts* shall interpret the characters following an option character requiring  
 92016 arguments as an argument whether or not this is done. An explicit null option-  
 92017 argument need not be recognized if it is not supplied as a separate argument when  
 92018 *getopts* is invoked. (See also the *getopt()* function defined in the System Interfaces  
 92019 volume of POSIX.1-2008.) The characters <question-mark> and <colon> shall not  
 92020 be used as option characters by an application. The use of other option characters  
 92021 that are not alphanumeric produces unspecified results. If the option-argument is  
 92022 not supplied as a separate argument from the option character, the value in  
 92023 *OPTARG* shall be stripped of the option character and the '-'. The first character  
 92024 in *optstring* determines how *getopts* behaves if an option character is not known or  
 92025 an option-argument is missing.

92026 *name* The name of a shell variable that shall be set by the *getopts* utility to the option  
 92027 character that was found.

92028 The *getopts* utility by default shall parse positional parameters passed to the invoking shell  
 92029 procedure. If *args* are given, they shall be parsed instead of the positional parameters.

## 92030 STDIN

92031 Not used.

## 92032 INPUT FILES

92033 None.

## 92034 ENVIRONMENT VARIABLES

92035 The following environment variables shall affect the execution of *getopts*:

92036 *LANG* Provide a default value for the internationalization variables that are unset or null.  
 92037 (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 92038 variables used to determine the values of locale categories.)

92039 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
 92040 internationalization variables.

92041 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
 92042 characters (for example, single-byte as opposed to multi-byte characters in  
 92043 arguments and input files).

92044 *LC\_MESSAGES*

92045 Determine the locale that should be used to affect the format and contents of  
 92046 diagnostic messages written to standard error.

92047 XSI *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

92048 *OPTIND* This variable shall be used by the *getopts* utility as the index of the next argument  
 92049 to be processed.

92050 **ASYNCHRONOUS EVENTS**

92051 Default.

92052 **STDOUT**

92053 Not used.

92054 **STDERR**

92055 Whenever an error is detected and the first character in the *optstring* operand is not a <colon>  
 92056 (' : '), a diagnostic message shall be written to standard error with the following information in  
 92057 an unspecified format:

- 92058 • The invoking program name shall be identified in the message. The invoking program  
 92059 name shall be the value of the shell special parameter 0 (see [Section 2.5.2](#), on page 2324) at  
 92060 the time the *getopts* utility is invoked. A name equivalent to:

92061 `basename "$0"`

92062 may be used.

- 92063 • If an option is found that was not specified in *optstring*, this error is identified and the  
 92064 invalid option character shall be identified in the message.

- 92065 • If an option requiring an option-argument is found, but an option-argument is not found,  
 92066 this error shall be identified and the invalid option character shall be identified in the  
 92067 message.

92068 **OUTPUT FILES**

92069 None.

92070 **EXTENDED DESCRIPTION**

92071 None.

92072 **EXIT STATUS**

92073 The following exit values shall be returned:

92074 `0` An option, specified or unspecified by *optstring*, was found.92075 `>0` The end of options was encountered or an error occurred.92076 **CONSEQUENCES OF ERRORS**

92077 Default.

92078 **APPLICATION USAGE**

92079 Since *getopts* affects the current shell execution environment, it is generally provided as a shell  
 92080 regular built-in. If it is called in a subshell or separate utility execution environment, such as one  
 92081 of the following:

92082 `(getopts abc value "$@")`92083 `nohup getopts ...`92084 `find . -exec getopts ... \;`

92085 it does not affect the shell variables in the caller's environment.

92086 Note that shell functions share *OPTIND* with the calling shell even though the positional  
 92087 parameters are changed. If the calling shell and any of its functions uses *getopts* to parse  
 92088 arguments, the results are unspecified.

**EXAMPLES**

The following example script parses and displays its arguments:

```
aflag=
bflag=
while getopts ab: name
do
 case $name in
 a) aflag=1;;
 b) bflag=1
 bval="$OPTARG";;
 ?) printf "Usage: %s: [-a] [-b value] args\n" $0
 exit 2;;
 esac
done
if [! -z "$aflag"]; then
 printf "Option -a specified\n"
fi
if [! -z "$bflag"]; then
 printf 'Option -b "%s" specified\n' "$bval"
fi
shift $(($OPTIND - 1))
printf "Remaining arguments are: %s\n" "$@"
```

**RATIONALE**

The *getopts* utility was chosen in preference to the System V *getopt* utility because *getopts* handles option-arguments containing <blank> characters.

The *OPTARG* variable is not mentioned in the ENVIRONMENT VARIABLES section because it does not affect the execution of *getopts*; it is one of the few “output-only” variables used by the standard utilities.

The <colon> is not allowed as an option character because that is not historical behavior, and it violates the Utility Syntax Guidelines. The <colon> is now specified to behave as in the KornShell version of the *getopts* utility; when used as the first character in the *optstring* operand, it disables diagnostics concerning missing option-arguments and unexpected option characters. This replaces the use of the *OPTERR* variable that was specified in an early proposal.

The formats of the diagnostic messages produced by the *getopts* utility and the *getopt()* function are not fully specified because implementations with superior (“friendlier”) formats objected to the formats used by some historical implementations. The standard developers considered it important that the information in the messages used be uniform between *getopts* and *getopt()*. Exact duplication of the messages might not be possible, particularly if a utility is built on another system that has a different *getopt()* function, but the messages must have specific information included so that the program name, invalid option character, and type of error can be distinguished by a user.

Only a rare application program intercepts a *getopts* standard error message and wants to parse it. Therefore, implementations are free to choose the most usable messages they can devise. The following formats are used by many historical implementations:

```
%s: illegal option -- %c\n", <program name>, <option character>

%s: option requires an argument -- %c\n", <program name>, \
 <option character>
```

92136 Historical shells with built-in versions of *getopt()* or *getopts* have used different formats,  
92137 frequently not even indicating the option character found in error.

92138 **FUTURE DIRECTIONS**

92139 None.

92140 **SEE ALSO**

92141 [Section 2.5.2](#) (on page 2324)

92142 XBD [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 215)

92143 XSH *getopt()*

92144 **CHANGE HISTORY**

92145 First released in Issue 4.

92146 **Issue 6**

92147 The normative text is reworded to avoid use of the term “must” for application requirements.

92148 **Issue 7**

92149 POSIX.1-2008, Technical Corrigendum 1, XCU/TC1-2008/0092 [159] is applied.

+

**NAME**

grep — search a file for a pattern

**SYNOPSIS**

```
grep [-E|-F] [-c|-l|-q] [-insvx] -e pattern_list
 [-e pattern_list]... [-f pattern_file]... [file...]

grep [-E|-F] [-c|-l|-q] [-insvx] [-e pattern_list]...
 -f pattern_file [-f pattern_file]... [file...]

grep [-E|-F] [-c|-l|-q] [-insvx] pattern_list [file...]
```

**DESCRIPTION**

The *grep* utility shall search the input files, selecting lines matching one or more patterns; the types of patterns are controlled by the options specified. The patterns are specified by the **-e** option, **-f** option, or the *pattern\_list* operand. The *pattern\_list*'s value shall consist of one or more patterns separated by <newline> characters; the *pattern\_file*'s contents shall consist of one or more patterns terminated by a <newline> character. By default, an input line shall be selected if any pattern, treated as an entire basic regular expression (BRE) as described in XBD [Section 9.3](#) (on page 183), matches any part of the line excluding the terminating <newline>; a null BRE shall match every line. By default, each selected input line shall be written to the standard output.

Regular expression matching shall be based on text lines. Since a <newline> separates or terminates patterns (see the **-e** and **-f** options below), regular expressions cannot contain a <newline>. Similarly, since patterns are matched against individual lines (excluding the terminating <newline> characters) of the input, there is no way for a pattern to match a <newline> found in the input.

**OPTIONS**

The *grep* utility shall conform to XBD [Section 12.2](#) (on page 215).

The following options shall be supported:

**-E** Match using extended regular expressions. Treat each pattern specified as an ERE, as described in XBD [Section 9.4](#) (on page 188). If any entire ERE pattern matches some part of an input line excluding the terminating <newline>, the line shall be matched. A null ERE shall match every line.

**-F** Match using fixed strings. Treat each pattern specified as a string instead of a regular expression. If an input line contains any of the patterns as a contiguous sequence of bytes, the line shall be matched. A null string shall match every line.

**-c** Write only a count of selected lines to standard output.

**-e pattern\_list**

Specify one or more patterns to be used during the search for input. The application shall ensure that patterns in *pattern\_list* are separated by a <newline>. A null pattern can be specified by two adjacent <newline> characters in *pattern\_list*. Unless the **-E** or **-F** option is also specified, each pattern shall be treated as a BRE, as described in XBD [Section 9.3](#) (on page 183). Multiple **-e** and **-f** options shall be accepted by the *grep* utility. All of the specified patterns shall be used when matching lines, but the order of evaluation is unspecified.

**-f pattern\_file**

Read one or more patterns from the file named by the pathname *pattern\_file*. Patterns in *pattern\_file* shall be terminated by a <newline>. A null pattern can be specified by an empty line in *pattern\_file*. Unless the **-E** or **-F** option is also

- 92196 specified, each pattern shall be treated as a BRE, as described in XBD [Section 9.3](#)  
 92197 (on page 183).
- 92198 **-i** Perform pattern matching in searches without regard to case; see XBD [Section 9.2](#)  
 92199 (on page 182).
- 92200 **-l** (The letter ell.) Write only the names of files containing selected lines to standard  
 92201 output. Pathnames shall be written once per file searched. If the standard input is  
 92202 searched, a pathname of "(standard input)" shall be written, in the POSIX  
 92203 locale. In other locales, "standard input" may be replaced by something more  
 92204 appropriate in those locales.
- 92205 **-n** Precede each output line by its relative line number in the file, each file starting at  
 92206 line 1. The line number counter shall be reset for each file processed.
- 92207 **-q** Quiet. Nothing shall be written to the standard output, regardless of matching  
 92208 lines. Exit with zero status if an input line is selected.
- 92209 **-s** Suppress the error messages ordinarily written for nonexistent or unreadable files.  
 92210 Other error messages shall not be suppressed.
- 92211 **-v** Select lines not matching any of the specified patterns. If the **-v** option is not  
 92212 specified, selected lines shall be those that match any of the specified patterns.
- 92213 **-x** Consider only input lines that use all characters in the line excluding the  
 92214 terminating <newline> to match an entire fixed string or regular expression to be  
 92215 matching lines.

## 92216 OPERANDS

92217 The following operands shall be supported:

- 92218 *pattern\_list* Specify one or more patterns to be used during the search for input. This operand  
 92219 shall be treated as if it were specified as **-e pattern\_list**.
- 92220 *file* A pathname of a file to be searched for the patterns. If no *file* operands are  
 92221 specified, the standard input shall be used.

## 92222 STDIN

92223 The standard input shall be used if no *file* operands are specified, and shall be used if a *file*  
 92224 operand is '-' and the implementation treats the '-' as meaning standard input. Otherwise,  
 92225 the standard input shall not be used. See the INPUT FILES section.

## 92226 INPUT FILES

92227 The input files shall be text files.

## 92228 ENVIRONMENT VARIABLES

92229 The following environment variables shall affect the execution of *grep*:

- 92230 *LANG* Provide a default value for the internationalization variables that are unset or null.  
 92231 (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 92232 variables used to determine the values of locale categories.)
- 92233 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
 92234 internationalization variables.
- 92235 *LC\_COLLATE*  
 92236 Determine the locale for the behavior of ranges, equivalence classes, and multi-  
 92237 character collating elements within regular expressions.

|       |                               |                                                                                                                                                                                                                                                            |
|-------|-------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 92238 | <i>LC_CTYPE</i>               | Determine the locale for the interpretation of sequences of bytes of text data as characters (for example, single-byte as opposed to multi-byte characters in arguments and input files) and the behavior of character classes within regular expressions. |
| 92239 |                               |                                                                                                                                                                                                                                                            |
| 92240 |                               |                                                                                                                                                                                                                                                            |
| 92241 |                               |                                                                                                                                                                                                                                                            |
| 92242 | <i>LC_MESSAGES</i>            |                                                                                                                                                                                                                                                            |
| 92243 |                               | Determine the locale that should be used to affect the format and contents of diagnostic messages written to standard error.                                                                                                                               |
| 92244 |                               |                                                                                                                                                                                                                                                            |
| 92245 | XSI <i>NLSPATH</i>            | Determine the location of message catalogs for the processing of <i>LC_MESSAGES</i> .                                                                                                                                                                      |
| 92246 | <b>ASYNCHRONOUS EVENTS</b>    |                                                                                                                                                                                                                                                            |
| 92247 |                               | Default.                                                                                                                                                                                                                                                   |
| 92248 | <b>STDOUT</b>                 |                                                                                                                                                                                                                                                            |
| 92249 |                               | If the <b>-l</b> option is in effect, the following shall be written for each file containing at least one selected input line:                                                                                                                            |
| 92250 |                               |                                                                                                                                                                                                                                                            |
| 92251 |                               | <code>"%s\n", &lt;file&gt;</code>                                                                                                                                                                                                                          |
| 92252 |                               | Otherwise, if more than one <i>file</i> argument appears, and <b>-q</b> is not specified, the <i>grep</i> utility shall prefix each output line by:                                                                                                        |
| 92253 |                               |                                                                                                                                                                                                                                                            |
| 92254 |                               | <code>"%s: ", &lt;file&gt;</code>                                                                                                                                                                                                                          |
| 92255 |                               | The remainder of each output line shall depend on the other options specified:                                                                                                                                                                             |
| 92256 |                               | <ul style="list-style-type: none"> <li>• If the <b>-c</b> option is in effect, the remainder of each output line shall contain:</li> </ul>                                                                                                                 |
| 92257 |                               | <code>"%d\n", &lt;count&gt;</code>                                                                                                                                                                                                                         |
| 92258 |                               | <ul style="list-style-type: none"> <li>• Otherwise, if <b>-c</b> is not in effect and the <b>-n</b> option is in effect, the following shall be written to standard output:</li> </ul>                                                                     |
| 92259 |                               |                                                                                                                                                                                                                                                            |
| 92260 |                               | <code>"%d: ", &lt;line number&gt;</code>                                                                                                                                                                                                                   |
| 92261 |                               | <ul style="list-style-type: none"> <li>• Finally, the following shall be written to standard output:</li> </ul>                                                                                                                                            |
| 92262 |                               | <code>"%s", &lt;selected-line contents&gt;</code>                                                                                                                                                                                                          |
| 92263 | <b>STDERR</b>                 |                                                                                                                                                                                                                                                            |
| 92264 |                               | The standard error shall be used only for diagnostic messages.                                                                                                                                                                                             |
| 92265 | <b>OUTPUT FILES</b>           |                                                                                                                                                                                                                                                            |
| 92266 |                               | None.                                                                                                                                                                                                                                                      |
| 92267 | <b>EXTENDED DESCRIPTION</b>   |                                                                                                                                                                                                                                                            |
| 92268 |                               | None.                                                                                                                                                                                                                                                      |
| 92269 | <b>EXIT STATUS</b>            |                                                                                                                                                                                                                                                            |
| 92270 |                               | The following exit values shall be returned:                                                                                                                                                                                                               |
| 92271 | 0                             | One or more lines were selected.                                                                                                                                                                                                                           |
| 92272 | 1                             | No lines were selected.                                                                                                                                                                                                                                    |
| 92273 | >1                            | An error occurred.                                                                                                                                                                                                                                         |
| 92274 | <b>CONSEQUENCES OF ERRORS</b> |                                                                                                                                                                                                                                                            |
| 92275 |                               | If the <b>-q</b> option is specified, the exit status shall be zero if an input line is selected, even if an error was detected. Otherwise, default actions shall be performed.                                                                            |
| 92276 |                               |                                                                                                                                                                                                                                                            |

## APPLICATION USAGE

Care should be taken when using characters in *pattern\_list* that may also be meaningful to the command interpreter. It is safest to enclose the entire *pattern\_list* argument in single-quotes:

'...'

The **-e** *pattern\_list* option has the same effect as the *pattern\_list* operand, but is useful when *pattern\_list* begins with the <hyphen> delimiter. It is also useful when it is more convenient to provide multiple patterns as separate arguments.

Multiple **-e** and **-f** options are accepted and *grep* uses all of the patterns it is given while matching input text lines. (Note that the order of evaluation is not specified. If an implementation finds a null string as a pattern, it is allowed to use that pattern first, matching every line, and effectively ignore any other patterns.)

The **-q** option provides a means of easily determining whether or not a pattern (or string) exists in a group of files. When searching several files, it provides a performance improvement (because it can quit as soon as it finds the first match) and requires less care by the user in choosing the set of files to supply as arguments (because it exits zero if it finds a match even if *grep* detected an access or read error on earlier *file* operands).

## EXAMPLES

1. To find all uses of the word "Posix" (in any case) in file **text.mm** and write with line numbers:

```
grep -i -n posix text.mm
```

2. To find all empty lines in the standard input:

```
grep ^$
```

or:

```
grep -v .
```

3. Both of the following commands print all lines containing strings "abc" or "def" or both:

```
grep -E 'abc|def'
```

```
grep -F 'abc
def'
```

4. Both of the following commands print all lines matching exactly "abc" or "def":

```
grep -E '^abc$|^def$'
```

```
grep -F -x 'abc
def'
```

## RATIONALE

This *grep* has been enhanced in an upwards-compatible way to provide the exact functionality of the historical *egrep* and *fgrep* commands as well. It was the clear intention of the standard developers to consolidate the three *greps* into a single command.

The old *egrep* and *fgrep* commands are likely to be supported for many years to come as implementation extensions, allowing historical applications to operate unmodified.

Historical implementations usually silently ignored all but one of multiply-specified **-e** and **-f** options, but were not consistent as to which specification was actually used.



- 92318 The **-b** option was omitted from the **OPTIONS** section because block numbers are  
92319 implementation-defined.
- 92320 The System V restriction on using **-** to mean standard input was omitted.
- 92321 A definition of action taken when given a null BRE or ERE is specified. This is an error  
92322 condition in some historical implementations.
- 92323 The **-l** option previously indicated that its use was undefined when no files were explicitly  
92324 named. This behavior was historical and placed an unnecessary restriction on future  
92325 implementations. It has been removed.
- 92326 The historical BSD *grep* **-s** option practice is easily duplicated by redirecting standard output to  
92327 **/dev/null**. The **-s** option required here is from System V.
- 92328 The **-x** option, historically available only with *fgrep*, is available here for all of the non-  
92329 obsolescent versions.
- 92330 **FUTURE DIRECTIONS**
- 92331 None.
- 92332 **SEE ALSO**
- 92333 *sed*
- 92334 XBD [Chapter 8](#) (on page 173), [Chapter 9](#) (on page 181), [Section 12.2](#) (on page 215)
- 92335 **CHANGE HISTORY**
- 92336 First released in Issue 2.
- 92337 **Issue 6**
- 92338 The Open Group Corrigendum U029/5 is applied, correcting the SYNOPSIS.
- 92339 The normative text is reworded to avoid use of the term “must” for application requirements.
- 92340 IEEE Std 1003.1-2001/Cor 1-2002, item XCU/TC1/D6/28 is applied, correcting the examples  
92341 using the *grep* **-F** option which did not match the normative description of the **-F** option.
- 92342 **Issue 7**
- 92343 Austin Group Interpretation 1003.1-2001 #092 is applied.
- 92344 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.
- 92345 SD5-XCU-ERN-98 is applied, updating the STDOUT section.

92346 **NAME**

92347 hash — remember or report utility locations

92348 **SYNOPSIS**92349 hash [*utility*...]

92350 hash -r

92351 **DESCRIPTION**

92352 The *hash* utility shall affect the way the current shell environment remembers the locations of  
 92353 utilities found as described in [Section 2.9.1.1](#) (on page 2339). Depending on the arguments  
 92354 specified, it shall add utility locations to its list of remembered locations or it shall purge the  
 92355 contents of the list. When no arguments are specified, it shall report on the contents of the list.

92356 Utilities provided as built-ins to the shell shall not be reported by *hash*.92357 **OPTIONS**92358 The *hash* utility shall conform to XBD [Section 12.2](#) (on page 215).

92359 The following option shall be supported:

92360 -r Forget all previously remembered utility locations.

92361 **OPERANDS**

92362 The following operand shall be supported:

92363 *utility* The name of a utility to be searched for and added to the list of remembered  
 92364 locations. If *utility* contains one or more <slash> characters, the results are  
 92365 unspecified.

92366 **STDIN**

92367 Not used.

92368 **INPUT FILES**

92369 None.

92370 **ENVIRONMENT VARIABLES**92371 The following environment variables shall affect the execution of *hash*:

92372 *LANG* Provide a default value for the internationalization variables that are unset or null.  
 92373 (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 92374 variables used to determine the values of locale categories.)

92375 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
 92376 internationalization variables.

92377 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
 92378 characters (for example, single-byte as opposed to multi-byte characters in  
 92379 arguments).

92380 *LC\_MESSAGES*

92381 Determine the locale that should be used to affect the format and contents of  
 92382 diagnostic messages written to standard error.

92383 XSI *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.92384 *PATH* Determine the location of *utility*, as described in XBD [Chapter 8](#) (on page 173).

**92385 ASYNCHRONOUS EVENTS**

92386 Default.

**92387 STDOUT**

92388 The standard output of *hash* shall be used when no arguments are specified. Its format is  
92389 unspecified, but includes the pathname of each utility in the list of remembered locations for the  
92390 current shell environment. This list shall consist of those utilities named in previous *hash*  
92391 invocations that have been invoked, and may contain those invoked and found through the  
92392 normal command search process.

**92393 STDERR**

92394 The standard error shall be used only for diagnostic messages.

**92395 OUTPUT FILES**

92396 None.

**92397 EXTENDED DESCRIPTION**

92398 None.

**92399 EXIT STATUS**

92400 The following exit values shall be returned:

92401 0 Successful completion.

92402 >0 An error occurred.

**92403 CONSEQUENCES OF ERRORS**

92404 Default.

**92405 APPLICATION USAGE**

92406 Since *hash* affects the current shell execution environment, it is always provided as a shell  
92407 regular built-in. If it is called in a separate utility execution environment, such as one of the  
92408 following:

92409 `nohup hash -r`  
92410 `find . -type f | xargs hash`

92411 it does not affect the command search process of the caller's environment.

92412 The *hash* utility may be implemented as an alias—for example, *alias -t -*, in which case utilities  
92413 found through normal command search are not listed by the *hash* command.

92414 The effects of *hash -r* can also be achieved portably by resetting the value of *PATH*; in the  
92415 simplest form, this can be:

92416 `PATH="$PATH"`

92417 The use of *hash* with *utility* names is unnecessary for most applications, but may provide a  
92418 performance improvement on a few implementations; normally, the hashing process is included  
92419 by default.

**92420 EXAMPLES**

92421 None.

**92422 RATIONALE**

92423 None.

92424 **FUTURE DIRECTIONS**

92425 None.

92426 **SEE ALSO**92427 [Section 2.9.1.1](#) (on page 2339)92428 XBD [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 215)92429 **CHANGE HISTORY**

92430 First released in Issue 2.

92431 **Issue 7**92432 The *hash* utility is moved from the XSI option to the Base.

92433 POSIX.1-2008, Technical Corrigendum 1, XCU/TC1-2008/0093 [241] is applied.

+

92434 **NAME**

92435 head — copy the first part of files

92436 **SYNOPSIS**92437 head [**-n** *number*] [*file*...]92438 **DESCRIPTION**92439 The *head* utility shall copy its input files to the standard output, ending the output for each file at  
92440 a designated point.92441 Copying shall end at the point in each input file indicated by the **-n** *number* option. The option-  
92442 argument *number* shall be counted in units of lines.92443 **OPTIONS**92444 The *head* utility shall conform to XBD [Section 12.2](#) (on page 215).

92445 The following option shall be supported:

92446 **-n** *number* The first *number* lines of each input file shall be copied to standard output. The  
92447 application shall ensure that the *number* option-argument is a positive decimal  
92448 integer.92449 When a file contains less than *number* lines, it shall be copied to standard output in its entirety.  
92450 This shall not be an error.92451 If no options are specified, *head* shall act as if **-n 10** had been specified.92452 **OPERANDS**

92453 The following operand shall be supported:

92454 *file* A pathname of an input file. If no *file* operands are specified, the standard input  
92455 shall be used.92456 **STDIN**92457 The standard input shall be used if no *file* operands are specified, and shall be used if a *file*  
92458 operand is '-' and the implementation treats the '-' as meaning standard input. Otherwise,  
92459 the standard input shall not be used. See the INPUT FILES section.92460 **INPUT FILES**

92461 Input files shall be text files, but the line length is not restricted to {LINE\_MAX} bytes.

92462 **ENVIRONMENT VARIABLES**92463 The following environment variables shall affect the execution of *head*:92464 **LANG** Provide a default value for the internationalization variables that are unset or null.  
92465 (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
92466 variables used to determine the values of locale categories.)92467 **LC\_ALL** If set to a non-empty string value, override the values of all the other  
92468 internationalization variables.92469 **LC\_CTYPE** Determine the locale for the interpretation of sequences of bytes of text data as  
92470 characters (for example, single-byte as opposed to multi-byte characters in  
92471 arguments and input files).92472 **LC\_MESSAGES**92473 Determine the locale that should be used to affect the format and contents of  
92474 diagnostic messages written to standard error.

92475 XSI **NLSPATH** Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

92476 **ASYNCHRONOUS EVENTS**

92477 Default.

92478 **STDOUT**

92479 The standard output shall contain designated portions of the input files.

92480 If multiple *file* operands are specified, *head* shall precede the output for each with the header:

92481 "*\n==> %s <==\n*", *<pathname>*

92482 except that the first header written shall not include the initial *<newline>*.

92483 **STDERR**

92484 The standard error shall be used only for diagnostic messages.

92485 **OUTPUT FILES**

92486 None.

92487 **EXTENDED DESCRIPTION**

92488 None.

92489 **EXIT STATUS**

92490 The following exit values shall be returned:

92491 0 Successful completion.

92492 >0 An error occurred.

92493 **CONSEQUENCES OF ERRORS**

92494 Default.

92495 **APPLICATION USAGE**

92496 None.

92497 **EXAMPLES**

92498 To write the first ten lines of all files (except those with a leading period) in the directory:

92499 *head -- \**

92500 **RATIONALE**

92501 Although it is possible to simulate *head* with *sed 10q* for a single file, the standard developers

92502 decided that the popularity of *head* on historical BSD systems warranted its inclusion alongside

92503 *tail*.

92504 POSIX.1-2008 version of *head* follows the Utility Syntax Guidelines. The *-n* option was added to

92505 this new interface so that *head* and *tail* would be more logically related. Earlier versions of this

92506 standard allowed a *-number* option. This form is no longer specified by POSIX.1-2008 but may

92507 be present in some implementations.

92508 There is no *-c* option (as there is in *tail*) because it is not historical practice and because other

92509 utilities in this volume of POSIX.1-2008 provide similar functionality.

92510 **FUTURE DIRECTIONS**

92511 None.

92512 **SEE ALSO**

92513 *sed*, *tail*

92514 XBD [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 215)

**92515 CHANGE HISTORY**

92516 First released in Issue 4.

**92517 Issue 6**

92518 The obsolescent **–number** form is removed.

92519 The normative text is reworded to avoid use of the term “must” for application requirements.

92520 The DESCRIPTION is updated to clarify that when a file contains less than the number of lines  
92521 requested, the entire file is copied to standard output.

**92522 Issue 7**

92523 Austin Group Interpretations 1003.1-2001 #027 and #092 are applied.

92524 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

92525 The APPLICATION USAGE section is removed and the EXAMPLES section is corrected.

**NAME**

iconv — codeset conversion

**SYNOPSIS**

```
iconv [-cs] -f frommap -t tomap [file...]
iconv -f fromcode [-cs] [-t tocode] [file...]
iconv -t tocode [-cs] [-f fromcode] [file...]
iconv -l
```

**DESCRIPTION**

The *iconv* utility shall convert the encoding of characters in *file* from one codeset to another and write the results to standard output.

When the options indicate that charmap files are used to specify the codesets (see **OPTIONS**), the codeset conversion shall be accomplished by performing a logical join on the symbolic character names in the two charmaps. The implementation need not support the use of charmap files for codeset conversion unless the POSIX2\_LOCALEDEF symbol is defined on the system.

**OPTIONS**

The *iconv* utility shall conform to XBD [Section 12.2](#) (on page 215).

The following options shall be supported:

**-c** Omit any characters that are invalid in the codeset of the input file from the output. When **-c** is not used, the results of encountering invalid characters in the input stream (either those that are not characters in the codeset of the input file or that have no corresponding character in the codeset of the output file) shall be specified in the system documentation. The presence or absence of **-c** shall not affect the exit status of *iconv*.

**-f** *fromcodeset*

Identify the codeset of the input file. The implementation shall recognize the following two forms of the *fromcodeset* option-argument:

*fromcode* The *fromcode* option-argument must not contain a <slash> character. It shall be interpreted as the name of one of the codeset descriptions provided by the implementation in an unspecified format. Valid values of *fromcode* are implementation-defined.

*frommap* The *frommap* option-argument must contain a <slash> character. It shall be interpreted as the pathname of a charmap file as defined in XBD [Section 6.4](#) (on page 129). If the pathname does not represent a valid, readable charmap file, the results are undefined.

If this option is omitted, the codeset of the current locale shall be used.

**-l** Write all supported *fromcode* and *tocode* values to standard output in an unspecified format.

**-s** Suppress any messages written to standard error concerning invalid characters. When **-s** is not used, the results of encountering invalid characters in the input stream (either those that are not valid characters in the codeset of the input file or that have no corresponding character in the codeset of the output file) shall be specified in the system documentation. The presence or absence of **-s** shall not affect the exit status of *iconv*.



|       |                            |                                                                               |
|-------|----------------------------|-------------------------------------------------------------------------------|
| 92569 | <b>-t <i>tocodeset</i></b> | Identify the codeset to be used for the output file. The implementation shall |
| 92570 |                            | recognize the following two forms of the <i>tocodeset</i> option-argument:    |

|       |                |                                                                           |
|-------|----------------|---------------------------------------------------------------------------|
| 92571 | <i>to</i> code | The semantics shall be equivalent to the <code>-f fromcode</code> option. |
|-------|----------------|---------------------------------------------------------------------------|

|       |              |                                                                          |
|-------|--------------|--------------------------------------------------------------------------|
| 92572 | <i>tomap</i> | The semantics shall be equivalent to the <code>-f frommap</code> option. |
|-------|--------------|--------------------------------------------------------------------------|

92573 If this option is omitted, the codeset of the current locale shall be used.

92574 If either `-f` or `-t` represents a charmap file, but the other does not (or is omitted), or both `-f` and  
92575 `-t` are omitted, the results are undefined.

## 92576 OPERANDS

92577 The following operand shall be supported:

|       |             |                                                                                                       |
|-------|-------------|-------------------------------------------------------------------------------------------------------|
| 92578 | <i>file</i> | A pathname of an input file. If no <i>file</i> operands are specified, or if a <i>file</i> operand is |
| 92579 |             | '-', the standard input shall be used.                                                                |

## 92580 STDIN

92581 The standard input shall be used only if no *file* operands are specified, or if a *file* operand is '–'.

## 92582 INPUT FILES

92583            The input file shall be a text file.

## 92584 ENVIRONMENT VARIABLES

92585      The following environment variables shall affect the execution of *iconv*:

|       |             |                                                                                               |
|-------|-------------|-----------------------------------------------------------------------------------------------|
| 92586 | <i>LANG</i> | Provide a default value for the internationalization variables that are unset or null.        |
| 92587 |             | (See XBD <a href="#">Section 8.2</a> (on page 174) for the precedence of internationalization |
| 92588 |             | variables used to determine the values of locale categories.)                                 |

|       |               |                                                                          |
|-------|---------------|--------------------------------------------------------------------------|
| 92589 | <i>LC_ALL</i> | If set to a non-empty string value, override the values of all the other |
| 92590 |               | internationalization variables.                                          |

92591 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
92592 characters (for example, single-byte as opposed to multi-byte characters in  
92593 arguments). During translation of the file, this variable is superseded by the use of  
92594 the *fromcode* option-argument.

```

92595 LC_MESSAGES
92596 Determine the locale that should be used to affect the format and contents of
92597 diagnostic messages written to standard error.

```

|       |     |                |                                                                                       |
|-------|-----|----------------|---------------------------------------------------------------------------------------|
| 92598 | XSI | <i>NLSPATH</i> | Determine the location of message catalogs for the processing of <i>LC_MESSAGES</i> . |
|-------|-----|----------------|---------------------------------------------------------------------------------------|

92599 **ASYNCHRONOUS EVENTS**

|       |          |
|-------|----------|
| 92600 | Default. |
|-------|----------|

92601 **STDOUT**

92602 When the `-I` option is used, the standard output shall contain all supported *fromcode* and *tocode*  
92603 values, written in an unspecified format.

92604 When the `-l` option is not used, the standard output shall contain the sequence of characters  
92605 read from the input files, translated to the specified codeset. Nothing else shall be written to the  
92606 standard output.

## 92607 STDERR

92608 The standard error shall be used only for diagnostic messages.

**92609 OUTPUT FILES**

92610 None.

**92611 EXTENDED DESCRIPTION**

92612 None.

**92613 EXIT STATUS**

92614 The following exit values shall be returned:

92615 0 Successful completion.

92616 >0 An error occurred.

**92617 CONSEQUENCES OF ERRORS**

92618 Default.

**92619 APPLICATION USAGE**

92620 The user must ensure that both charmap files use the same symbolic names for characters the  
92621 two codesets have in common.

**92622 EXAMPLES**

92623 The following example converts the contents of file **mail.x400** from the ISO/IEC 6937:2001  
92624 standard codeset to the ISO/IEC 8859-1:1998 standard codeset, and stores the results in file  
92625 **mail.local**:

92626 `iconv -f IS6937 -t IS8859 mail.x400 > mail.local`

**92627 RATIONALE**

92628 The *iconv* utility can be used portably only when the user provides two charmap files as option-  
92629 arguments. This is because a single charmap provided by the user cannot reliably be joined with  
92630 the names in a system-provided character set description. The valid values for *fromcode* and  
92631 *toctype* are implementation-defined and do not have to have any relation to the charmap  
92632 mechanisms. As an aid to interactive users, the `-l` option was adopted from the Plan 9 operating  
92633 system. It writes information concerning these implementation-defined values. The format is  
92634 unspecified because there are many possible useful formats that could be chosen, such as a  
92635 matrix of valid combinations of *fromcode* and *toctype*. The `-l` option is not intended for shell script  
92636 usage; conforming applications will have to use charmaps.

92637 The *iconv* utility may support the conversion between ASCII and EBCDIC-based encodings, but +  
92638 is not required to do so. In an XSI-compliant implementation, the *dd* utility is the only method +  
92639 guaranteed to support conversion between these two character sets.

**92640 FUTURE DIRECTIONS**

92641 None.

**92642 SEE ALSO**

92643 *dd*, *gencat*

92644 XBD [Section 6.4](#) (on page 129), [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 215)

**92645 CHANGE HISTORY**

92646 First released in Issue 3.

**92647 Issue 6**

92648 This utility has been rewritten to align with the IEEE P1003.2b draft standard. Specifically, the  
92649 ability to use charmap files for conversion has been added.

92650 IEEE Std 1003.1-2001/Cor 1-2002, item XCU/TC1/D6/29 is applied, making changes to address  
92651 inconsistencies with the *iconv()* function in the System Interfaces volume of POSIX.1-2008.

92652 **Issue 7**

92653 Austin Group Interpretation 1003.1-2001 #206 is applied, correcting the *tomap* option.

92654 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

92655 POSIX.1-2008, Technical Corrigendum 1, XCU/TC1-2008/0094 [291] and XCU/TC1-2008/0095 +  
92656 [291] are applied.

92657 **NAME**

92658       id — return user identity

92659 **SYNOPSIS**92660       id [*user*]92661       id -G [-n] [*user*]92662       id -g [-nr] [*user*]92663       id -u [-nr] [*user*]92664 **DESCRIPTION**

92665       If no *user* operand is provided, the *id* utility shall write the user and group IDs and the  
 92666       corresponding user and group names of the invoking process to standard output. If the effective  
 92667       and real IDs do not match, both shall be written. If multiple groups are supported by the  
 92668       underlying system (see the description of {NGROUPS\_MAX} in the System Interfaces volume of  
 92669       POSIX.1-2008), the supplementary group affiliations of the invoking process shall also be  
 92670       written.

92671       If a *user* operand is provided and the process has appropriate privileges, the user and group IDs  
 92672       of the selected user shall be written. In this case, effective IDs shall be assumed to be identical to  
 92673       real IDs. If the selected user has more than one allowable group membership listed in the group  
 92674       database, these shall be written in the same manner as the supplementary groups described in  
 92675       the preceding paragraph.

92676 **OPTIONS**92677       The *id* utility shall conform to XBD [Section 12.2](#) (on page 215).

92678       The following options shall be supported:

92679       **-G**       Output all different group IDs (effective, real, and supplementary) only, using the  
 92680       format "%u\n". If there is more than one distinct group affiliation, output each  
 92681       such affiliation, using the format " %u", before the <newline> is output.

92682       **-g**       Output only the effective group ID, using the format "%u\n".

92683       **-n**       Output the name in the format "%s" instead of the numeric ID using the format  
 92684       "%u".

92685       **-r**       Output the real ID instead of the effective ID.

92686       **-u**       Output only the effective user ID, using the format "%u\n".

92687 **OPERANDS**

92688       The following operand shall be supported:

92689       *user*       The login name for which information is to be written.

92690 **STDIN**

92691       Not used.

92692 **INPUT FILES**

92693       None.

92694 **ENVIRONMENT VARIABLES**92695       The following environment variables shall affect the execution of *id*:

92696       **LANG**       Provide a default value for the internationalization variables that are unset or null.  
 92697       (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 92698       variables used to determine the values of locale categories.)

92699 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
 92700 internationalization variables.

92701 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
 92702 characters (for example, single-byte as opposed to multi-byte characters in  
 92703 arguments).

92704 *LC\_MESSAGES*  
 92705 Determine the locale that should be used to affect the format and contents of  
 92706 diagnostic messages written to standard error and informative messages written to  
 92707 standard output.

92708 XSI *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

92709 **ASYNCHRONOUS EVENTS**  
 92710 Default.

92711 **STDOUT**  
 92712 The following formats shall be used when the *LC\_MESSAGES* locale category specifies the  
 92713 POSIX locale. In other locales, the strings *uid*, *gid*, *euid*, *egid*, and *groups* may be replaced with  
 92714 more appropriate strings corresponding to the locale.

92715 "uid=%u(%s) gid=%u(%s)\n", <real user ID>, <user-name>,  
 92716 <real group ID>, <group-name>

92717 If the effective and real user IDs do not match, the following shall be inserted immediately  
 92718 before the '\n' character in the previous format:

92719 " euid=%u(%s) "

92720 with the following arguments added at the end of the argument list:

92721 <effective user ID>, <effective user-name>

92722 If the effective and real group IDs do not match, the following shall be inserted directly before  
 92723 the '\n' character in the format string (and after any addition resulting from the effective and  
 92724 real user IDs not matching):

92725 " egid=%u(%s) "

92726 with the following arguments added at the end of the argument list:

92727 <effective group-ID>, <effective group name>

92728 If the process has supplementary group affiliations or the selected user is allowed to belong to  
 92729 multiple groups, the first shall be added directly before the <newline> in the format string:

92730 " groups=%u(%s) "

92731 with the following arguments added at the end of the argument list:

92732 <supplementary group ID>, <supplementary group name>

92733 and the necessary number of the following added after that for any remaining supplementary  
 92734 group IDs:

92735 ", %u(%s) "

92736 and the necessary number of the following arguments added at the end of the argument list:

92737 <supplementary group ID>, <supplementary group name>

92738 If any of the user ID, group ID, effective user ID, effective group ID, or supplementary/multiple

group IDs cannot be mapped by the system into printable user or group names, the corresponding "(%s)" and *name* argument shall be omitted from the corresponding format string.

When any of the options are specified, the output format shall be as described in the OPTIONS section.

#### STDERR

The standard error shall be used only for diagnostic messages.

#### OUTPUT FILES

None.

#### EXTENDED DESCRIPTION

None.

#### EXIT STATUS

The following exit values shall be returned:

0 Successful completion.

>0 An error occurred.

#### CONSEQUENCES OF ERRORS

Default.

#### APPLICATION USAGE

Output produced by the **-G** option and by the default case could potentially produce very long lines on systems that support large numbers of supplementary groups. (On systems with user and group IDs that are 32-bit integers and with group names with a maximum of 8 bytes per name, 93 supplementary groups plus distinct effective and real group and user IDs could theoretically overflow the 2048-byte {LINE\_MAX} text file line limit on the default output case. It would take about 186 supplementary groups to overflow the 2048-byte barrier using *id -G*). This is not expected to be a problem in practice, but in cases where it is a concern, applications should consider using *fold -s* before post-processing the output of *id*.

#### EXAMPLES

None.

#### RATIONALE

The functionality provided by the 4 BSD *groups* utility can be simulated using:

```
id -Gn [user]
```

The 4 BSD command *groups* was considered, but it was not included because it did not provide the functionality of the *id* utility of the SVID. Also, it was thought that it would be easier to modify *id* to provide the additional functionality necessary to systems with multiple groups than to invent another command.

The options **-u**, **-g**, **-n**, and **-r** were added to ease the use of *id* with shell commands substitution. Without these options it is necessary to use some preprocessor such as *sed* to select the desired piece of information. Since output such as that produced by:

```
id -u -n
```

is frequently wanted, it seemed desirable to add the options.

92779 **FUTURE DIRECTIONS**

92780 None.

92781 **SEE ALSO**92782 *fold, logname, who*92783 XBD [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 215)92784 XSH *getgid()*, *getgroups()*, *getuid()*92785 **CHANGE HISTORY**

92786 First released in Issue 2.

92787 **Issue 7**

92788 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

92789 **NAME**92790 `ipcrm` — remove an XSI message queue, semaphore set, or shared memory segment identifier92791 **SYNOPSIS**92792 XSI `ipcrm [-q msgid|-Q msgkey|-s semid|-S semkey|-m shmid|-M shmkey]...`92793 **DESCRIPTION**92794 The *ipcrm* utility shall remove zero or more message queues, semaphore sets, or shared memory  
92795 segments. The interprocess communication facilities to be removed are specified by the options.92796 Only a user with appropriate privileges shall be allowed to remove an interprocess  
92797 communication facility that was not created by or owned by the user invoking *ipcrm*.92798 **OPTIONS**92799 The *ipcrm* utility shall conform to XBD [Section 12.2](#) (on page 215).

92800 The following options shall be supported:

92801 `-q msgid` Remove the message queue identifier *msgid* from the system and destroy the  
92802 message queue and data structure associated with it.92803 `-m shmid` Remove the shared memory identifier *shmid* from the system. The shared memory  
92804 segment and data structure associated with it shall be destroyed after the last  
92805 detach.92806 `-s semid` Remove the semaphore identifier *semid* from the system and destroy the set of  
92807 semaphores and data structure associated with it.92808 `-Q msgkey` Remove the message queue identifier, created with key *msgkey*, from the system  
92809 and destroy the message queue and data structure associated with it.92810 `-M shmkey` Remove the shared memory identifier, created with key *shmkey*, from the system.  
92811 The shared memory segment and data structure associated with it shall be  
92812 destroyed after the last detach.92813 `-S semkey` Remove the semaphore identifier, created with key *semkey*, from the system and  
92814 destroy the set of semaphores and data structure associated with it.92815 **OPERANDS**

92816 None.

92817 **STDIN**

92818 Not used.

92819 **INPUT FILES**

92820 None.

92821 **ENVIRONMENT VARIABLES**92822 The following environment variables shall affect the execution of *ipcrm*:92823 `LANG` Provide a default value for the internationalization variables that are unset or null.  
92824 (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
92825 variables used to determine the values of locale categories.)92826 `LC_ALL` If set to a non-empty string value, override the values of all the other  
92827 internationalization variables.92828 `LC_CTYPE` Determine the locale for the interpretation of sequences of bytes of text data as  
92829 characters (for example, single-byte as opposed to multi-byte characters in  
92830 arguments).



|       |                               |                                                                                         |
|-------|-------------------------------|-----------------------------------------------------------------------------------------|
| 92831 | <i>LC_MESSAGES</i>            |                                                                                         |
| 92832 |                               | Determine the locale that should be used to affect the format and contents of           |
| 92833 |                               | diagnostic messages written to standard error.                                          |
| 92834 | <i>NLSPATH</i>                | Determine the location of message catalogs for the processing of <i>LC_MESSAGES</i> .   |
| 92835 | <b>ASYNCHRONOUS EVENTS</b>    |                                                                                         |
| 92836 |                               | Default.                                                                                |
| 92837 | <b>STDOUT</b>                 |                                                                                         |
| 92838 |                               | Not used.                                                                               |
| 92839 | <b>STDERR</b>                 |                                                                                         |
| 92840 |                               | The standard error shall be used only for diagnostic messages.                          |
| 92841 | <b>OUTPUT FILES</b>           |                                                                                         |
| 92842 |                               | None.                                                                                   |
| 92843 | <b>EXTENDED DESCRIPTION</b>   |                                                                                         |
| 92844 |                               | None.                                                                                   |
| 92845 | <b>EXIT STATUS</b>            |                                                                                         |
| 92846 |                               | The following exit values shall be returned:                                            |
| 92847 | 0                             | Successful completion.                                                                  |
| 92848 | >0                            | An error occurred.                                                                      |
| 92849 | <b>CONSEQUENCES OF ERRORS</b> |                                                                                         |
| 92850 |                               | Default.                                                                                |
| 92851 | <b>APPLICATION USAGE</b>      |                                                                                         |
| 92852 |                               | None.                                                                                   |
| 92853 | <b>EXAMPLES</b>               |                                                                                         |
| 92854 |                               | None.                                                                                   |
| 92855 | <b>RATIONALE</b>              |                                                                                         |
| 92856 |                               | None.                                                                                   |
| 92857 | <b>FUTURE DIRECTIONS</b>      |                                                                                         |
| 92858 |                               | None.                                                                                   |
| 92859 | <b>SEE ALSO</b>               |                                                                                         |
| 92860 |                               | <i>ipcs</i>                                                                             |
| 92861 |                               | XBD <a href="#">Chapter 8</a> (on page 173), <a href="#">Section 12.2</a> (on page 215) |
| 92862 |                               | XSH <i>msgctl()</i> , <i>semctl()</i> , <i>shmctl()</i>                                 |
| 92863 | <b>CHANGE HISTORY</b>         |                                                                                         |
| 92864 |                               | First released in Issue 5.                                                              |
| 92865 | <b>Issue 7</b>                |                                                                                         |
| 92866 |                               | SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.                                       |

92867 **NAME**

92868        *ipcs* — report XSI interprocess communication facilities status

92869 **SYNOPSIS**

92870 XSI        *ipcs* [-qms] [-a|-bcopt]

92871 **DESCRIPTION**

92872        The *ipcs* utility shall write information about active interprocess communication facilities.

92873        Without options, information shall be written in short format for message queues, shared  
92874        memory segments, and semaphore sets that are currently active in the system. Otherwise, the  
92875        information that is displayed is controlled by the options specified.

92876 **OPTIONS**

92877        The *ipcs* utility shall conform to XBD [Section 12.2](#) (on page 215).

92878        The *ipcs* utility accepts the following options:

92879        **-q**        Write information about active message queues.

92880        **-m**        Write information about active shared memory segments.

92881        **-s**        Write information about active semaphore sets.

92882        If **-q**, **-m**, or **-s** are specified, only information about those facilities shall be written. If none of  
92883        these three are specified, information about all three shall be written subject to the following  
92884        options:

92885        **-a**        Use all print options. (This is a shorthand notation for **-b**, **-c**, **-o**, **-p**, and **-t**.)

92886        **-b**        Write information on maximum allowable size. (Maximum number of bytes in  
92887        messages on queue for message queues, size of segments for shared memory, and  
92888        number of semaphores in each set for semaphores.)

92889        **-c**        Write creator's user name and group name; see below.

92890        **-o**        Write information on outstanding usage. (Number of messages on queue and total  
92891        number of bytes in messages on queue for message queues, and number of  
92892        processes attached to shared memory segments.)

92893        **-p**        Write process number information. (Process ID of the last process to send a  
92894        message and process ID of the last process to receive a message on message  
92895        queues, process ID of the creating process, and process ID of the last process to  
92896        attach or detach on shared memory segments.)

92897        **-t**        Write time information. (Time of the last control operation that changed the access  
92898        permissions for all facilities, time of the last *msgsnd()* and *msgrcv()* operations on  
92899        message queues, time of the last *shmat()* and *shmdt()* operations on shared  
92900        memory, and time of the last *semop()* operation on semaphores.)

92901 **OPERANDS**

92902        None.

92903 **STDIN**

92904        Not used.

92905 **INPUT FILES**

- 92906        • The group database

- The user database

## ENVIRONMENT VARIABLES

The following environment variables shall affect the execution of *ipcs*:

|             |                                                                                                                                                                                                                                                    |
|-------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| LANG        | Provide a default value for the internationalization variables that are unset or null. (See XBD <a href="#">Section 8.2</a> (on page 174) for the precedence of internationalization variables used to determine the values of locale categories.) |
| LC_ALL      | If set to a non-empty string value, override the values of all the other internationalization variables.                                                                                                                                           |
| LC_CTYPE    | Determine the locale for the interpretation of sequences of bytes of text data as characters (for example, single-byte as opposed to multi-byte characters in arguments).                                                                          |
| LC_MESSAGES | Determine the locale that should be used to affect the format and contents of diagnostic messages written to standard error.                                                                                                                       |
| NLSPATH     | Determine the location of message catalogs for the processing of <i>LC_MESSAGES</i> .                                                                                                                                                              |
| TZ          | Determine the timezone for the date and time strings written by <i>ipcs</i> . If <i>TZ</i> is unset or null, an unspecified default timezone shall be used.                                                                                        |

## ASYNCHRONOUS EVENTS

Default.

## STDOUT

An introductory line shall be written with the format:

```
"IPC status from %s as of %s\n", <source>, <date>
```

where *<source>* indicates the source used to gather the statistics and *<date>* is the information that would be produced by the *date* command when invoked in the POSIX locale.

The *ipcs* utility then shall create up to three reports depending upon the *-q*, *-m*, and *-s* options. The first report shall indicate the status of message queues, the second report shall indicate the status of shared memory segments, and the third report shall indicate the status of semaphore sets.

If the corresponding facility is not installed or has not been used since the last reboot, then the report shall be written out in the format:

```
"%s facility not in system.\n", <facility>
```

where *<facility>* is *Message Queue*, *Shared Memory*, or *Semaphore*, as appropriate. If the facility has been installed and has been used since the last reboot, column headings separated by one or more *<space>* characters and followed by a *<newline>* shall be written as indicated below followed by the facility name written out using the format:

```
"%s:\n", <facility>
```

where *<facility>* is *Message Queues*, *Shared Memory*, or *Semaphores*, as appropriate. On the second and third reports the column headings need not be written if the last column headings written already provide column headings for all information in that report.

The column headings provided in the first column below and the meaning of the information in those columns shall be given in order below; the letters in parentheses indicate the options that shall cause the corresponding column to appear; “all” means that the column shall always appear. Each column is separated by one or more *<space>* characters. Note that these options

|       |                                                                                          |                                                                                                     |
|-------|------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------|
| 92950 | only determine what information is provided for each report; they do not determine which |                                                                                                     |
| 92951 | reports are written.                                                                     |                                                                                                     |
| 92952 | T (all)                                                                                  | Type of facility:                                                                                   |
| 92953 | q                                                                                        | Message queue.                                                                                      |
| 92954 | m                                                                                        | Shared memory segment.                                                                              |
| 92955 | s                                                                                        | Semaphore.                                                                                          |
| 92956 | This field is a single character written using the format %c.                            |                                                                                                     |
| 92957 | ID (all)                                                                                 | The identifier for the facility entry. This field shall be written using the format                 |
| 92958 |                                                                                          | %d.                                                                                                 |
| 92959 | KEY (all)                                                                                | The key used as an argument to <i>msgget()</i> , <i>semget()</i> , or <i>shmget()</i> to create the |
| 92960 |                                                                                          | facility entry.                                                                                     |
| 92961 | MODE (all)                                                                               | <b>Note:</b> The key of a shared memory segment is changed to IPC_PRIVATE when the                  |
| 92962 |                                                                                          | segment has been removed until all processes attached to the segment                                |
| 92963 |                                                                                          | detach it.                                                                                          |
| 92964 |                                                                                          | This field shall be written using the format 0x%x.                                                  |
| 92965 | MODE (all)                                                                               | The facility access modes and flags. The mode shall consist of 11 characters                        |
| 92966 |                                                                                          | that are interpreted as follows.                                                                    |
| 92967 |                                                                                          | The first character shall be:                                                                       |
| 92968 |                                                                                          | S If a process is waiting on a <i>msgsnd()</i> operation.                                           |
| 92969 |                                                                                          | – If the above is not true.                                                                         |
| 92970 |                                                                                          | The second character shall be:                                                                      |
| 92971 |                                                                                          | R If a process is waiting on a <i>msgrcv()</i> operation.                                           |
| 92972 |                                                                                          | C or – If the associated shared memory segment is to be cleared when the                            |
| 92973 |                                                                                          | first attach operation is executed.                                                                 |
| 92974 |                                                                                          | – If none of the above is true.                                                                     |
| 92975 |                                                                                          | The next nine characters shall be interpreted as three sets of three bits each.                     |
| 92976 |                                                                                          | The first set refers to the owner's permissions; the next to permissions of                         |
| 92977 |                                                                                          | others in the usergroup of the facility entry; and the last to all others. Within                   |
| 92978 |                                                                                          | each set, the first character indicates permission to read, the second character                    |
| 92979 |                                                                                          | indicates permission to write or alter the facility entry, and the last character is                |
| 92980 |                                                                                          | a minus-sign ('-').                                                                                 |
| 92981 |                                                                                          | The permissions shall be indicated as follows:                                                      |
| 92982 |                                                                                          | r If read permission is granted.                                                                    |
| 92983 |                                                                                          | w If write permission is granted.                                                                   |
| 92984 |                                                                                          | a If alter permission is granted.                                                                   |
| 92985 |                                                                                          | – If the indicated permission is not granted.                                                       |
| 92986 |                                                                                          | The first character following the permissions specifies if there is an alternate or                 |
| 92987 |                                                                                          | additional access control method associated with the facility. If there is no                       |
| 92988 |                                                                                          | alternate or additional access control method associated with the facility, a                       |
| 92989 |                                                                                          | single <space> shall be written; otherwise, another printable character is                          |

|       |                                                                          |                                                                                                                                                                                                                                                                                                                |
|-------|--------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 92990 |                                                                          | written.                                                                                                                                                                                                                                                                                                       |
| 92991 | OWNER (all)                                                              | The user name of the owner of the facility entry. If the user name of the owner is found in the user database, at least the first eight column positions of the name shall be written using the format %s. Otherwise, the user ID of the owner shall be written using the format %d.                           |
| 92992 |                                                                          |                                                                                                                                                                                                                                                                                                                |
| 92993 |                                                                          |                                                                                                                                                                                                                                                                                                                |
| 92994 |                                                                          |                                                                                                                                                                                                                                                                                                                |
| 92995 | GROUP (all)                                                              | The group name of the owner of the facility entry. If the group name of the owner is found in the group database, at least the first eight column positions of the name shall be written using the format %s. Otherwise, the group ID of the owner shall be written using the format %d.                       |
| 92996 |                                                                          |                                                                                                                                                                                                                                                                                                                |
| 92997 |                                                                          |                                                                                                                                                                                                                                                                                                                |
| 92998 |                                                                          |                                                                                                                                                                                                                                                                                                                |
| 92999 | The following nine columns shall be only written out for message queues: |                                                                                                                                                                                                                                                                                                                |
| 93000 | CREATOR (a,c)                                                            | The user name of the creator of the facility entry. If the user name of the creator is found in the user database, at least the first eight column positions of the name shall be written using the format %s. Otherwise, the user ID of the creator shall be written using the format %d.                     |
| 93001 |                                                                          |                                                                                                                                                                                                                                                                                                                |
| 93002 |                                                                          |                                                                                                                                                                                                                                                                                                                |
| 93003 |                                                                          |                                                                                                                                                                                                                                                                                                                |
| 93004 | CGROUP (a,c)                                                             | The group name of the creator of the facility entry. If the group name of the creator is found in the group database, at least the first eight column positions of the name shall be written using the format %s. Otherwise, the group ID of the creator shall be written using the format %d.                 |
| 93005 |                                                                          |                                                                                                                                                                                                                                                                                                                |
| 93006 |                                                                          |                                                                                                                                                                                                                                                                                                                |
| 93007 |                                                                          |                                                                                                                                                                                                                                                                                                                |
| 93008 | CBYTES (a,o)                                                             | The number of bytes in messages currently outstanding on the associated message queue. This field shall be written using the format %d.                                                                                                                                                                        |
| 93009 |                                                                          |                                                                                                                                                                                                                                                                                                                |
| 93010 | QNUM (a,o)                                                               | The number of messages currently outstanding on the associated message queue. This field shall be written using the format %d.                                                                                                                                                                                 |
| 93011 |                                                                          |                                                                                                                                                                                                                                                                                                                |
| 93012 | QBYTES (a,b)                                                             | The maximum number of bytes allowed in messages outstanding on the associated message queue. This field shall be written using the format %d.                                                                                                                                                                  |
| 93013 |                                                                          |                                                                                                                                                                                                                                                                                                                |
| 93014 | LSPID (a,p)                                                              | The process ID of the last process to send a message to the associated queue. This field shall be written using the format:                                                                                                                                                                                    |
| 93015 |                                                                          |                                                                                                                                                                                                                                                                                                                |
| 93016 |                                                                          | "%d", <pid>                                                                                                                                                                                                                                                                                                    |
| 93017 |                                                                          | where <pid> is 0 if no message has been sent to the corresponding message queue; otherwise, <pid> shall be the process ID of the last process to send a message to the queue.                                                                                                                                  |
| 93018 |                                                                          |                                                                                                                                                                                                                                                                                                                |
| 93019 |                                                                          |                                                                                                                                                                                                                                                                                                                |
| 93020 | LRPID (a,p)                                                              | The process ID of the last process to receive a message from the associated queue. This field shall be written using the format:                                                                                                                                                                               |
| 93021 |                                                                          |                                                                                                                                                                                                                                                                                                                |
| 93022 |                                                                          | "%d", <pid>                                                                                                                                                                                                                                                                                                    |
| 93023 |                                                                          | where <pid> is 0 if no message has been received from the corresponding message queue; otherwise, <pid> shall be the process ID of the last process to receive a message from the queue.                                                                                                                       |
| 93024 |                                                                          |                                                                                                                                                                                                                                                                                                                |
| 93025 |                                                                          |                                                                                                                                                                                                                                                                                                                |
| 93026 | STIME (a,t)                                                              | The time the last message was sent to the associated queue. If a message has been sent to the corresponding message queue, the hour, minute, and second of the last time a message was sent to the queue shall be written using the format %d:%2.2d:%2.2d. Otherwise, the format " no-entry" shall be written. |
| 93027 |                                                                          |                                                                                                                                                                                                                                                                                                                |
| 93028 |                                                                          |                                                                                                                                                                                                                                                                                                                |
| 93029 |                                                                          |                                                                                                                                                                                                                                                                                                                |
| 93030 |                                                                          |                                                                                                                                                                                                                                                                                                                |
| 93031 | RTIME (a,t)                                                              | The time the last message was received from the associated queue. If a message has been received from the corresponding message queue, the hour, minute, and second of the last time a message was received from the queue                                                                                     |
| 93032 |                                                                          |                                                                                                                                                                                                                                                                                                                |
| 93033 |                                                                          |                                                                                                                                                                                                                                                                                                                |

93034 shall be written using the format %d:%2.2d:%2.2d. Otherwise, the format  
93035 " no-entry" shall be written.

93036 The following eight columns shall be only written out for shared memory segments.

93037 CREATOR (a,c) The user of the creator of the facility entry. If the user name of the creator is  
93038 found in the user database, at least the first eight column positions of the  
93039 name shall be written using the format %s. Otherwise, the user ID of the  
93040 creator shall be written using the format %d.

93041 CGROUP (a,c) The group name of the creator of the facility entry. If the group name of the  
93042 creator is found in the group database, at least the first eight column positions  
93043 of the name shall be written using the format %s. Otherwise, the group ID of  
93044 the creator shall be written using the format %d.

93045 NATTCH (a,o) The number of processes attached to the associated shared memory segment.  
93046 This field shall be written using the format %d.

93047 SEGSZ (a,b) The size of the associated shared memory segment. This field shall be written  
93048 using the format %d.

93049 CPID (a,p) The process ID of the creator of the shared memory entry. This field shall be  
93050 written using the format %d.

93051 LPID (a,p) The process ID of the last process to attach or detach the shared memory  
93052 segment. This field shall be written using the format:

93053 "%d", <pid>

93054 where <pid> is 0 if no process has attached the corresponding shared memory  
93055 segment; otherwise, <pid> shall be the process ID of the last process to attach  
93056 or detach the segment.

93057 ATIME (a,t) The time the last attach on the associated shared memory segment was  
93058 completed. If the corresponding shared memory segment has ever been  
93059 attached, the hour, minute, and second of the last time the segment was  
93060 attached shall be written using the format %d:%2.2d:%2.2d. Otherwise, the  
93061 format " no-entry" shall be written.

93062 DTIME (a,t) The time the last detach on the associated shared memory segment was  
93063 completed. If the corresponding shared memory segment has ever been  
93064 detached, the hour, minute, and second of the last time the segment was  
93065 detached shall be written using the format %d:%2.2d:%2.2d. Otherwise, the  
93066 format " no-entry" shall be written.

93067 The following four columns shall be only written out for semaphore sets:

93068 CREATOR (a,c) The user of the creator of the facility entry. If the user name of the creator is  
93069 found in the user database, at least the first eight column positions of the  
93070 name shall be written using the format %s. Otherwise, the user ID of the  
93071 creator shall be written using the format %d.

93072 CGROUP (a,c) The group name of the creator of the facility entry. If the group name of the  
93073 creator is found in the group database, at least the first eight column positions  
93074 of the name shall be written using the format %s. Otherwise, the group ID of  
93075 the creator shall be written using the format %d.

|       |                               |                                                                                                                                                                                                                                                                                                                                                                                  |
|-------|-------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 93076 | NSEMS (a,b)                   | The number of semaphores in the set associated with the semaphore entry.                                                                                                                                                                                                                                                                                                         |
| 93077 |                               | This field shall be written using the format %d.                                                                                                                                                                                                                                                                                                                                 |
| 93078 | OTIME (a,t)                   | The time the last semaphore operation on the set associated with the semaphore entry was completed. If a semaphore operation has ever been performed on the corresponding semaphore set, the hour, minute, and second of the last semaphore operation on the semaphore set shall be written using the format %d:%2.2d:%2.2d. Otherwise, the format " no-entry" shall be written. |
| 93079 |                               |                                                                                                                                                                                                                                                                                                                                                                                  |
| 93080 |                               |                                                                                                                                                                                                                                                                                                                                                                                  |
| 93081 |                               |                                                                                                                                                                                                                                                                                                                                                                                  |
| 93082 |                               |                                                                                                                                                                                                                                                                                                                                                                                  |
| 93083 |                               |                                                                                                                                                                                                                                                                                                                                                                                  |
| 93084 |                               | The following column shall be written for all three reports when it is requested:                                                                                                                                                                                                                                                                                                |
| 93085 | CTIME (a,t)                   | The time the associated entry was created or changed. The hour, minute, and second of the time when the associated entry was created shall be written using the format %d:%2.2d:%2.2d.                                                                                                                                                                                           |
| 93086 |                               |                                                                                                                                                                                                                                                                                                                                                                                  |
| 93087 |                               |                                                                                                                                                                                                                                                                                                                                                                                  |
| 93088 | <b>STDERR</b>                 |                                                                                                                                                                                                                                                                                                                                                                                  |
| 93089 |                               | The standard error shall be used only for diagnostic messages.                                                                                                                                                                                                                                                                                                                   |
| 93090 | <b>OUTPUT FILES</b>           |                                                                                                                                                                                                                                                                                                                                                                                  |
| 93091 |                               | None.                                                                                                                                                                                                                                                                                                                                                                            |
| 93092 | <b>EXTENDED DESCRIPTION</b>   |                                                                                                                                                                                                                                                                                                                                                                                  |
| 93093 |                               | None.                                                                                                                                                                                                                                                                                                                                                                            |
| 93094 | <b>EXIT STATUS</b>            |                                                                                                                                                                                                                                                                                                                                                                                  |
| 93095 |                               | The following exit values shall be returned:                                                                                                                                                                                                                                                                                                                                     |
| 93096 | 0                             | Successful completion.                                                                                                                                                                                                                                                                                                                                                           |
| 93097 | >0                            | An error occurred.                                                                                                                                                                                                                                                                                                                                                               |
| 93098 | <b>CONSEQUENCES OF ERRORS</b> |                                                                                                                                                                                                                                                                                                                                                                                  |
| 93099 |                               | Default.                                                                                                                                                                                                                                                                                                                                                                         |
| 93100 | <b>APPLICATION USAGE</b>      |                                                                                                                                                                                                                                                                                                                                                                                  |
| 93101 |                               | Things can change while <i>ipcs</i> is running; the information it gives is guaranteed to be accurate only when it was retrieved.                                                                                                                                                                                                                                                |
| 93102 |                               |                                                                                                                                                                                                                                                                                                                                                                                  |
| 93103 | <b>EXAMPLES</b>               |                                                                                                                                                                                                                                                                                                                                                                                  |
| 93104 |                               | None.                                                                                                                                                                                                                                                                                                                                                                            |
| 93105 | <b>RATIONALE</b>              |                                                                                                                                                                                                                                                                                                                                                                                  |
| 93106 |                               | None.                                                                                                                                                                                                                                                                                                                                                                            |
| 93107 | <b>FUTURE DIRECTIONS</b>      |                                                                                                                                                                                                                                                                                                                                                                                  |
| 93108 |                               | None.                                                                                                                                                                                                                                                                                                                                                                            |
| 93109 | <b>SEE ALSO</b>               |                                                                                                                                                                                                                                                                                                                                                                                  |
| 93110 |                               | <i>ipcrm</i>                                                                                                                                                                                                                                                                                                                                                                     |
| 93111 |                               | XBD <a href="#">Chapter 8</a> (on page 173), <a href="#">Section 12.2</a> (on page 215)                                                                                                                                                                                                                                                                                          |
| 93112 |                               | XSH <a href="#">msgrcv()</a> , <a href="#">msgsnd()</a> , <a href="#">semget()</a> , <a href="#">semop()</a> , <a href="#">shmat()</a> , <a href="#">shmdt()</a> , <a href="#">shmget()</a>                                                                                                                                                                                      |
| 93113 | <b>CHANGE HISTORY</b>         |                                                                                                                                                                                                                                                                                                                                                                                  |
| 93114 |                               | First released in Issue 5.                                                                                                                                                                                                                                                                                                                                                       |

93115 **Issue 6**

93116 The Open Group Corrigendum U020/1 is applied, correcting the SYNOPSIS.

93117 The Open Group Corrigenda U032/1 and U032/2 are applied, clarifying the output format.

93118 The Open Group Base Resolution bwg98-004 is applied.

93119 **Issue 7**

93120 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

93121 SD5-XCU-ERN-139 is applied, adding the *ipcrm* utility to the SEE ALSO section.



93122 **NAME**93123       *jobs* — display status of jobs in the current session93124 **SYNOPSIS**93125 UP       *jobs* **[-l|-p]** [*job\_id*...]93126 **DESCRIPTION**93127       The *jobs* utility shall display the status of jobs that were started in the current shell environment;  
93128       see [Section 2.12](#) (on page 2353).93129       When *jobs* reports the termination status of a job, the shell shall remove its process ID from the  
93130       list of those “known in the current shell execution environment”; see [Section 2.9.3.1](#) (on page  
93131       2342).93132 **OPTIONS**93133       The *jobs* utility shall conform to XBD [Section 12.2](#) (on page 215).

93134       The following options shall be supported:

93135       **-l**       (The letter ell.) Provide more information about each job listed. This information  
93136       shall include the job number, current job, process group ID, state, and the  
93137       command that formed the job.93138       **-p**       Display only the process IDs for the process group leaders of the selected jobs.93139       By default, the *jobs* utility shall display the status of all stopped jobs, running background jobs  
93140       and all jobs whose status has changed and have not been reported by the shell.93141 **OPERANDS**

93142       The following operand shall be supported:

93143       *job\_id*       Specifies the jobs for which the status is to be displayed. If no *job\_id* is given, the  
93144       status information for all jobs shall be displayed. The format of *job\_id* is described  
93145       in XBD [Section 3.204](#) (on page 65).93146 **STDIN**

93147       Not used.

93148 **INPUT FILES**

93149       None.

93150 **ENVIRONMENT VARIABLES**93151       The following environment variables shall affect the execution of *jobs*:93152       **LANG**       Provide a default value for the internationalization variables that are unset or null.  
93153       (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
93154       variables used to determine the values of locale categories.)93155       **LC\_ALL**       If set to a non-empty string value, override the values of all the other  
93156       internationalization variables.93157       **LC\_CTYPE**   Determine the locale for the interpretation of sequences of bytes of text data as  
93158       characters (for example, single-byte as opposed to multi-byte characters in  
93159       arguments).93160       **LC\_MESSAGES**93161       Determine the locale that should be used to affect the format and contents of  
93162       diagnostic messages written to standard error and informative messages written to  
93163       standard output.

93164 XSI **NLSPATH** Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

93165 **ASYNCHRONOUS EVENTS**

93166 Default.

93167 **STDOUT**

93168 If the **-p** option is specified, the output shall consist of one line for each process ID:

93169 "%d\n", <process ID>

93170 Otherwise, if the **-l** option is not specified, the output shall be a series of lines of the form:

93171 "[%d] %c %s %s\n", <job-number>, <current>, <state>, <command>

93172 where the fields shall be as follows:

93173 <current> The character '+' identifies the job that would be used as a default for the *fg* or *bg* utilities; this job can also be specified using the *job\_id* %+ or "%%". The character '-' identifies the job that would become the default if the current default job were to exit; this job can also be specified using the *job\_id* %-. For other jobs, this field is a <space>. At most one job can be identified with '+' and at most one job can be identified with '-'. If there is any suspended job, then the current job shall be a suspended job. If there are at least two suspended jobs, then the previous job also shall be a suspended job.

93181 <job-number> A number that can be used to identify the process group to the *wait*, *fg*, *bg*, and *kill* utilities. Using these utilities, the job can be identified by prefixing the job number with '% '.

93182

93183

93184 <state> One of the following strings (in the POSIX locale):

93185 **Running** Indicates that the job has not been suspended by a signal and has not exited.

93186

93187 **Done** Indicates that the job completed and returned exit status zero.

93188 **Done(code)** Indicates that the job completed normally and that it exited with the specified non-zero exit status, *code*, expressed as a decimal number.

93189

93190 **Stopped** Indicates that the job was suspended by the SIGTSTP signal.

93191 **Stopped (SIGTSTP)**

93192 Indicates that the job was suspended by the SIGTSTP signal.

93193 **Stopped (SIGSTOP)**

93194 Indicates that the job was suspended by the SIGSTOP signal.

93195 **Stopped (SIGTTIN)**

93196 Indicates that the job was suspended by the SIGTTIN signal.

93197 **Stopped (SIGTTOU)**

93198 Indicates that the job was suspended by the SIGTTOU signal.

93199 The implementation may substitute the string **Suspended** in place of **Stopped**. If the job was terminated by a signal, the format of <state> is unspecified, but it shall be visibly distinct from all of the other <state> formats shown here and shall indicate the name or description of the signal causing the termination.

93200

93201

93202

93203 <command> The associated command that was given to the shell.

93204 If the **-l** option is specified, a field containing the process group ID shall be inserted before the <state> field. Also, more processes in a process group may be output on separate lines, using

93205

93206 only the process ID and *<command>* fields.

#### 93207 **STDERR**

93208 The standard error shall be used only for diagnostic messages.

#### 93209 **OUTPUT FILES**

93210 None.

#### 93211 **EXTENDED DESCRIPTION**

93212 None.

#### 93213 **EXIT STATUS**

93214 The following exit values shall be returned:

93215 0 Successful completion.

93216 >0 An error occurred.

#### 93217 **CONSEQUENCES OF ERRORS**

93218 Default.

#### 93219 **APPLICATION USAGE**

93220 The **-p** option is the only portable way to find out the process group of a job because different  
 93221 implementations have different strategies for defining the process group of the job. Usage such  
 93222 as *\$(jobs -p)* provides a way of referring to the process group of the job in an implementation-  
 93223 independent way.

93224 The *jobs* utility does not work as expected when it is operating in its own utility execution  
 93225 environment because that environment has no applicable jobs to manipulate. See the  
 93226 APPLICATION USAGE section for *bg*. For this reason, *jobs* is generally implemented as a shell  
 93227 regular built-in.

#### 93228 **EXAMPLES**

93229 None.

#### 93230 **RATIONALE**

93231 Both "**%%**" and "**%+**" are used to refer to the current job. Both forms are of equal validity—the  
 93232 "**%%**" mirroring "**\$\$**" and "**%+**" mirroring the output of *jobs*. Both forms reflect historical  
 93233 practice of the KornShell and the C shell with job control.

93234 The job control features provided by *bg*, *fg*, and *jobs* are based on the KornShell. The standard  
 93235 developers examined the characteristics of the C shell versions of these utilities and found that  
 93236 differences exist. Despite widespread use of the C shell, the KornShell versions were selected for  
 93237 this volume of POSIX.1-2008 to maintain a degree of uniformity with the rest of the KornShell  
 93238 features selected (such as the very popular command line editing features).

93239 The *jobs* utility is not dependent on the job control option, as are the seemingly related *bg* and *fg*  
 93240 utilities because *jobs* is useful for examining background jobs, regardless of the condition of job  
 93241 control. When the user has invoked a *set +m* command and job control has been turned off, *jobs*  
 93242 can still be used to examine the background jobs associated with that current session. Similarly,  
 93243 *kill* can then be used to kill background jobs with *kill %<background job number>*.

93244 The output for terminated jobs is left unspecified to accommodate various historical systems.  
 93245 The following formats have been witnessed:

- 93246 1. **Killed**(*signal name*)
- 93247 2. *signal name*

93248 3. *signal name*(**coredump**)

93249 4. *signal description*– **core dumped**

93250 Most users should be able to understand these formats, although it means that applications have  
93251 trouble parsing them.

93252 The calculation of job IDs was not described since this would suggest an implementation, which  
93253 may impose unnecessary restrictions.

93254 In an early proposal, a **-n** option was included to “Display the status of jobs that have changed,  
93255 exited, or stopped since the last status report”. It was removed because the shell always writes  
93256 any changed status of jobs before each prompt.

#### 93257 **FUTURE DIRECTIONS**

93258 None.

#### 93259 **SEE ALSO**

93260 [Section 2.12](#) (on page 2353), *bg*, *fg*, *kill*, *wait*

93261 XBD [Section 3.204](#) (on page 65), [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 215)

#### 93262 **CHANGE HISTORY**

93263 First released in Issue 4.

#### 93264 **Issue 6**

93265 This utility is marked as part of the User Portability Utilities option.

93266 The JC shading is removed as job control is mandatory in this version.

#### 93267 **Issue 7**

93268 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

**NAME**

join — relational database operator

**SYNOPSIS**

```
join [-a file_number|-v file_number] [-e string] [-o list] [-t char]
 [-1 field] [-2 field] file1 file2
```

**DESCRIPTION**

The *join* utility shall perform an equality join on the files *file1* and *file2*. The joined files shall be written to the standard output.

The join field is a field in each file on which the files are compared. The *join* utility shall write one line in the output for each pair of lines in *file1* and *file2* that have identical join fields. The output line by default shall consist of the join field, then the remaining fields from *file1*, then the remaining fields from *file2*. This format can be changed by using the **-o** option (see below). The **-a** option can be used to add unmatched lines to the output. The **-v** option can be used to output only unmatched lines.

The files *file1* and *file2* shall be ordered in the collating sequence of *sort -b* on the fields on which they shall be joined, by default the first in each line. All selected output shall be written in the same collating sequence.

The default input field separators shall be <blank> characters. In this case, multiple separators shall count as one field separator, and leading separators shall be ignored. The default output field separator shall be a <space>.

The field separator and collating sequence can be changed by using the **-t** option (see below).

If the same key appears more than once in either file, all combinations of the set of remaining fields in *file1* and the set of remaining fields in *file2* are output in the order of the lines encountered.

If the input files are not in the appropriate collating sequence, the results are unspecified.

**OPTIONS**

The *join* utility shall conform to XBD [Section 12.2](#) (on page 215).

The following options shall be supported:

**-a file\_number**

Produce a line for each unpairable line in file *file\_number*, where *file\_number* is 1 or 2, in addition to the default output. If both **-a1** and **-a2** are specified, all unpairable lines shall be output.

**-e string**

Replace empty output fields in the list selected by **-o** with the string *string*.

**-o list**

Construct the output line to comprise the fields specified in *list*, each element of which shall have one of the following two forms:

1. *file\_number.field*, where *file\_number* is a file number and *field* is a decimal integer field number
2. 0 (zero), representing the join field

The elements of *list* shall be either <comma>-separated or <blank>-separated, as specified in Guideline 8 of XBD [Section 12.2](#) (on page 215). The fields specified by *list* shall be written for all selected output lines. Fields selected by *list* that do not appear in the input shall be treated as empty output fields. (See the **-e** option.) Only specifically requested fields shall be written. The application shall ensure that *list* is a single command line argument.

93313        **-t *char***        Use character *char* as a separator, for both input and output. Every appearance of  
 93314        *char* in a line shall be significant. When this option is specified, the collating  
 93315        sequence shall be the same as *sort* without the **-b** option.

93316        **-v *file\_number***        Instead of the default output, produce a line only for each unpairable line in  
 93317        *file\_number*, where *file\_number* is 1 or 2. If both **-v1** and **-v2** are specified, all  
 93318        unpairable lines shall be output.  
 93319

93320        **-1 *field***        Join on the *field*th field of file 1. Fields are decimal integers starting with 1.

93321        **-2 *field***        Join on the *field*th field of file 2. Fields are decimal integers starting with 1.

## 93322 OPERANDS

93323        The following operands shall be supported:

93324        *file1, file2*        A pathname of a file to be joined. If either of the *file1* or *file2* operands is **'-'**, the  
 93325        standard input shall be used in its place.

## 93326 STDIN

93327        The standard input shall be used only if the *file1* or *file2* operand is **'-'**. See the INPUT FILES  
 93328        section.

## 93329 INPUT FILES

93330        The input files shall be text files.

## 93331 ENVIRONMENT VARIABLES

93332        The following environment variables shall affect the execution of *join*:

93333        *LANG*        Provide a default value for the internationalization variables that are unset or null.  
 93334        (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 93335        variables used to determine the values of locale categories.)

93336        *LC\_ALL*        If set to a non-empty string value, override the values of all the other  
 93337        internationalization variables.

93338        *LC\_COLLATE*        Determine the locale of the collating sequence *join* expects to have been used when  
 93339        the input files were sorted.  
 93340

93341        *LC\_CTYPE*        Determine the locale for the interpretation of sequences of bytes of text data as  
 93342        characters (for example, single-byte as opposed to multi-byte characters in  
 93343        arguments and input files).

93344        *LC\_MESSAGES*        Determine the locale that should be used to affect the format and contents of  
 93345        diagnostic messages written to standard error.  
 93346

93347        XSI        *NLSPATH*        Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

## 93348 ASYNCHRONOUS EVENTS

93349        Default.

## 93350 STDOUT

93351        The *join* utility output shall be a concatenation of selected character fields. When the **-o** option  
 93352        is not specified, the output shall be:

93353        "%s%s%s\n", <*join field*>, <*other file1 fields*>,  
 93354        <*other file2 fields*>

93355        If the join field is not the first field in a file, the <*other file fields*> for that file shall be:

93356 <fields preceding join field>, <fields following join field>

93357 When the **-o** option is specified, the output format shall be:

93358 "%s\n", <concatenation of fields>

93359 where the concatenation of fields is described by the **-o** option, above.

93360 For either format, each field (except the last) shall be written with its trailing separator character.  
 93361 If the separator is the default (<blank> characters), a single <space> shall be written after each  
 93362 field (except the last).

## 93363 STDERR

93364 The standard error shall be used only for diagnostic messages.

## 93365 OUTPUT FILES

93366 None.

## 93367 EXTENDED DESCRIPTION

93368 None.

## 93369 EXIT STATUS

93370 The following exit values shall be returned:

93371 0 All input files were output successfully.

93372 >0 An error occurred.

## 93373 CONSEQUENCES OF ERRORS

93374 Default.

## 93375 APPLICATION USAGE

93376 Pathnames consisting of numeric digits or of the form *string.string* should not be specified  
 93377 directly following the **-o** list.

## 93378 EXAMPLES

93379 The **-o 0** field essentially selects the union of the join fields. For example, given file **phone**:

```
93380 !Name Phone Number
93381 Don +1 123-456-7890
93382 Hal +1 234-567-8901
93383 Yasushi +2 345-678-9012
```

93384 and file **fax**:

```
93385 !Name Fax Number
93386 Don +1 123-456-7899
93387 Keith +1 456-789-0122
93388 Yasushi +2 345-678-9011
```

93389 (where the large expanses of white space are meant to each represent a single <tab>), the  
 93390 command:

```
93391 join -t "<tab>" -a 1 -a 2 -e '(unknown)' -o 0,1.2,2.2 phone fax
```

93392 would produce:

```
93393 !Name Phone Number Fax Number
93394 Don +1 123-456-7890 +1 123-456-7899
93395 Hal +1 234-567-8901 (unknown)
93396 Keith (unknown) +1 456-789-0122
93397 Yasushi +2 345-678-9012 +2 345-678-9011
```

Multiple instances of the same key will produce combinatorial results. The following:

```
fa:
 a x
 a y
 a z
fb:
 a p
```

will produce:

```
a x p
a y p
a z p
```

And the following:

```
fa:
 a b c
 a d e
fb:
 a w x
 a y z
 a o p
```

will produce:

```
a b c w x
a b c y z
a b c o p
a d e w x
a d e y z
a d e o p
```

## RATIONALE

The **-e** option is only effective when used with **-o** because, unless specific fields are identified using **-o**, *join* is not aware of what fields might be empty. The exception to this is the join field, but identifying an empty join field with the **-e** string is not historical practice and some scripts might break if this were changed.

The 0 field in the **-o** list was adopted from the Tenth Edition version of *join* to satisfy international objections that the *join* in the base documents does not support the “full join” or “outer join” described in relational database literature. Although it has been possible to include a join field in the output (by default, or by field number using **-o**), the join field could not be included for an unpaired line selected by **-a**. The **-o 0** field essentially selects the union of the join fields.

This sort of outer join was not possible with the *join* commands in the base documents. The **-o 0** field was chosen because it is an upwards-compatible change for applications. An alternative was considered: have the join field represent the union of the fields in the files (where they are identical for matched lines, and one or both are null for unmatched lines). This was not adopted because it would break some historical applications.

The ability to specify *file2* as **-** is not historical practice; it was added for completeness.

The **-v** option is not historical practice, but was considered necessary because it permitted the writing of *only* those lines that do not match on the join field, as opposed to the **-a** option, which prints both lines that do and do not match. This additional facility is parallel with the **-v** option



93444 of *grep*.

93445 Some historical implementations have been encountered where a blank line in one of the input  
93446 files was considered to be the end of the file; the description in this volume of POSIX.1-2008 does  
93447 not cite this as an allowable case.

93448 Earlier versions of this standard allowed *-j*, *-j1*, *-j2* options, and a form of the *-o* option that  
93449 allowed the *list* option-argument to be multiple arguments. These forms are no longer specified  
93450 by POSIX.1-2008 but may be present in some implementations.

#### 93451 FUTURE DIRECTIONS

93452 None.

#### 93453 SEE ALSO

93454 *awk*, *comm*, *sort*, *uniq*

93455 XBD [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 215)

#### 93456 CHANGE HISTORY

93457 First released in Issue 2.

#### 93458 Issue 6

93459 The obsolescent *-j* options and the multi-argument *-o* option are removed in this version.

93460 The normative text is reworded to avoid use of the term “must” for application requirements.

#### 93461 Issue 7

93462 Austin Group Interpretation 1003.1-2001 #027 is applied.

93463 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

## 93464 NAME

93465 kill — terminate or signal processes

## 93466 SYNOPSIS

93467 kill -s *signal\_name* *pid*...93468 kill -l [*exit\_status*]93469 XSI kill [-*signal\_name*] *pid*...93470 kill [-*signal\_number*] *pid*...

## 93471 DESCRIPTION

93472 The *kill* utility shall send a signal to the process or processes specified by each *pid* operand.93473 For each *pid* operand, the *kill* utility shall perform actions equivalent to the *kill()* function  
93474 defined in the System Interfaces volume of POSIX.1-2008 called with the following arguments:

- 93475
- The value of the *pid* operand shall be used as the *pid* argument.
  - The *sig* argument is the value specified by the *-s* option, *-signal\_number* option, or the *-signal\_name* option, or by SIGTERM, if none of these options is specified.

## 93478 OPTIONS

93479 XSI The *kill* utility shall conform to XBD [Section 12.2](#) (on page 215), except that in the last two  
93480 SYNOPSIS forms, the *-signal\_number* and *-signal\_name* options are usually more than a single  
93481 character.

93482 The following options shall be supported:

93483 **-l** (The letter ell.) Write all values of *signal\_name* supported by the implementation, if  
93484 no operand is given. If an *exit\_status* operand is given and it is a value of the ' ? '   
93485 shell special parameter (see [Section 2.5.2](#) (on page 2324) and *wait*) corresponding to  
93486 a process that was terminated by a signal, the *signal\_name* corresponding to the  
93487 signal that terminated the process shall be written. If an *exit\_status* operand is  
93488 given and it is the unsigned decimal integer value of a signal number, the  
93489 *signal\_name* (the symbolic constant name without the **SIG** prefix defined in the  
93490 Base Definitions volume of POSIX.1-2008) corresponding to that signal shall be  
93491 written. Otherwise, the results are unspecified.93492 **-s *signal\_name***93493 Specify the signal to send, using one of the symbolic names defined in the  
93494 **<signal.h>** header. Values of *signal\_name* shall be recognized in a case-independent  
93495 fashion, without the **SIG** prefix. In addition, the symbolic name 0 shall be  
93496 recognized, representing the signal value zero. The corresponding signal shall be  
93497 sent instead of SIGTERM.93498 XSI **-*signal\_name***93499 Equivalent to **-s *signal\_name***.93500 XSI **-*signal\_number***93501 Specify a non-negative decimal integer, *signal\_number*, representing the signal to be  
93502 used instead of SIGTERM, as the *sig* argument in the effective call to *kill()*. The  
93503 correspondence between integer values and the *sig* value used is shown in the  
93504 following list.93505 The effects of specifying any *signal\_number* other than those listed below are  
93506 undefined.

|       |                                                                                                 |         |
|-------|-------------------------------------------------------------------------------------------------|---------|
| 93507 | 0                                                                                               | 0       |
| 93508 | 1                                                                                               | SIGHUP  |
| 93509 | 2                                                                                               | SIGINT  |
| 93510 | 3                                                                                               | SIGQUIT |
| 93511 | 6                                                                                               | SIGABRT |
| 93512 | 9                                                                                               | SIGKILL |
| 93513 | 14                                                                                              | SIGALRM |
| 93514 | 15                                                                                              | SIGTERM |
| 93515 | If the first argument is a negative integer, it shall be interpreted as a <i>-signal_number</i> |         |
| 93516 | option, not as a negative <i>pid</i> operand specifying a process group.                        |         |

## 93517 OPERANDS

93518 The following operands shall be supported:

|       |                    |                                                                                                |
|-------|--------------------|------------------------------------------------------------------------------------------------|
| 93519 | <i>pid</i>         | One of the following:                                                                          |
| 93520 |                    | 1. A decimal integer specifying a process or process group to be signaled. The                 |
| 93521 |                    | process or processes selected by positive, negative, and zero values of the                    |
| 93522 |                    | <i>pid</i> operand shall be as described for the <i>kill()</i> function. If process number 0   |
| 93523 |                    | is specified, all processes in the current process group shall be signaled. For                |
| 93524 |                    | the effects of negative <i>pid</i> numbers, see the <i>kill()</i> function defined in the      |
| 93525 |                    | System Interfaces volume of POSIX.1-2008. If the first <i>pid</i> operand is                   |
| 93526 |                    | negative, it should be preceded by "--" to keep it from being interpreted as                   |
| 93527 |                    | an option.                                                                                     |
| 93528 |                    | 2. A job control job ID (see XBD <a href="#">Section 3.204</a> , on page 65) that identifies a |
| 93529 |                    | background process group to be signaled. The job control job ID notation is                    |
| 93530 |                    | applicable only for invocations of <i>kill</i> in the current shell execution                  |
| 93531 |                    | environment; see <a href="#">Section 2.12</a> (on page 2353).                                  |
| 93532 | <i>exit_status</i> | A decimal integer specifying a signal number or the exit status of a process                   |
| 93533 |                    | terminated by a signal.                                                                        |

## 93534 STDIN

93535 Not used.

## 93536 INPUT FILES

93537 None.

## 93538 ENVIRONMENT VARIABLES

93539 The following environment variables shall affect the execution of *kill*:

|       |                 |                                                                                               |
|-------|-----------------|-----------------------------------------------------------------------------------------------|
| 93540 | <i>LANG</i>     | Provide a default value for the internationalization variables that are unset or null.        |
| 93541 |                 | (See XBD <a href="#">Section 8.2</a> (on page 174) for the precedence of internationalization |
| 93542 |                 | variables used to determine the values of locale categories.)                                 |
| 93543 | <i>LC_ALL</i>   | If set to a non-empty string value, override the values of all the other                      |
| 93544 |                 | internationalization variables.                                                               |
| 93545 | <i>LC_CTYPE</i> | Determine the locale for the interpretation of sequences of bytes of text data as             |
| 93546 |                 | characters (for example, single-byte as opposed to multi-byte characters in                   |
| 93547 |                 | arguments).                                                                                   |

93548 *LC\_MESSAGES*  
 93549 Determine the locale that should be used to affect the format and contents of  
 93550 diagnostic messages written to standard error.

93551 XSI *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

93552 **ASYNCHRONOUS EVENTS**  
 93553 Default.

93554 **STDOUT**  
 93555 When the *-l* option is not specified, the standard output shall not be used.  
 93556 When the *-l* option is specified, the symbolic name of each signal shall be written in the  
 93557 following format:  
 93558 "%s%c", <signal\_name>, <separator>  
 93559 where the <signal\_name> is in uppercase, without the **SIG** prefix, and the <separator> shall be  
 93560 either a <newline> or a <space>. For the last signal written, <separator> shall be a <newline>.  
 93561 When both the *-l* option and *exit\_status* operand are specified, the symbolic name of the  
 93562 corresponding signal shall be written in the following format:  
 93563 "%s\n", <signal\_name>

93564 **STDERR**  
 93565 The standard error shall be used only for diagnostic messages.

93566 **OUTPUT FILES**  
 93567 None.

93568 **EXTENDED DESCRIPTION**  
 93569 None.

93570 **EXIT STATUS**  
 93571 The following exit values shall be returned:  
 93572 0 At least one matching process was found for each *pid* operand, and the specified signal was  
 93573 successfully processed for at least one matching process.  
 93574 >0 An error occurred.

93575 **CONSEQUENCES OF ERRORS**  
 93576 Default.

93577 **APPLICATION USAGE**  
 93578 Process numbers can be found by using *ps*.  
 93579 The job control job ID notation is not required to work as expected when *kill* is operating in its  
 93580 own utility execution environment. In either of the following examples:  
 93581 nohup kill %1 &  
 93582 system("kill %1");  
 93583 the *kill* operates in a different environment and does not share the shell's understanding of job  
 93584 numbers.

93585 **EXAMPLES**  
 93586 Any of the commands:  
 93587 kill -9 100 -165  
 93588 kill -s kill 100 -165

```
93589 kill -s KILL 100 -165
```

```
93590 sends the SIGKILL signal to the process whose process ID is 100 and to all processes whose
93591 process group ID is 165, assuming the sending process has permission to send that signal to the
93592 specified processes, and that they exist.
```

```
93593 The System Interfaces volume of POSIX.1-2008 and this volume of POSIX.1-2008 do not require
93594 specific signal numbers for any signal_names. Even the -signal_number option provides symbolic
93595 (although numeric) names for signals. If a process is terminated by a signal, its exit status
93596 indicates the signal that killed it, but the exact values are not specified. The kill -l option,
93597 however, can be used to map decimal signal numbers and exit status values into the name of a
93598 signal. The following example reports the status of a terminated job:
```

```
93599 job
93600 stat=$?
93601 if [$stat -eq 0]
93602 then
93603 echo job completed successfully.
93604 elif [$stat -gt 128]
93605 then
93606 echo job terminated by signal SIG$(kill -l $stat).
93607 else
93608 echo job terminated with error code $stat.
93609 fi
```

```
93610 To send the default signal to a process group (say 123), an application should use a command
93611 similar to one of the following:
```

```
93612 kill -TERM -123
93613 kill -- -123
```

#### 93614 RATIONALE

```
93615 The -l option originated from the C shell, and is also implemented in the KornShell. The C shell
93616 output can consist of multiple output lines because the signal names do not always fit on a
93617 single line on some terminal screens. The KornShell output also included the implementation-
93618 defined signal numbers and was considered by the standard developers to be too difficult for
93619 scripts to parse conveniently. The specified output format is intended not only to accommodate
93620 the historical C shell output, but also to permit an entirely vertical or entirely horizontal listing
93621 on systems for which this is appropriate.
```

```
93622 An early proposal invented the name SIGNULL as a signal_name for signal 0 (used by the System
93623 Interfaces volume of POSIX.1-2008 to test for the existence of a process without sending it a
93624 signal). Since the signal_name 0 can be used in this case unambiguously, SIGNULL has been
93625 removed.
```

```
93626 An early proposal also required symbolic signal_names to be recognized with or without the SIG
93627 prefix. Historical versions of kill have not written the SIG prefix for the -l option and have not
93628 recognized the SIG prefix on signal_names. Since neither applications portability nor ease-of-use
93629 would be improved by requiring this extension, it is no longer required.
```

```
93630 To avoid an ambiguity of an initial negative number argument specifying either a signal number
93631 or a process group, POSIX.1-2008 mandates that it is always considered the former by
93632 implementations that support the XSI option. It also requires that conforming applications
93633 always use the "--" options terminator argument when specifying a process group, unless an
93634 option is also specified.
```

```
93635 The -s option was added in response to international interest in providing some form of kill that
```

93636 meets the Utility Syntax Guidelines.

93637 The job control job ID notation is not required to work as expected when *kill* is operating in its  
93638 own utility execution environment. In either of the following examples:

93639 `nohup kill %1 &`  
93640 `system("kill %1");`

93641 the *kill* operates in a different environment and does not understand how the shell has managed  
93642 its job numbers.

#### 93643 **FUTURE DIRECTIONS**

93644 None.

#### 93645 **SEE ALSO**

93646 [Chapter 2](#) (on page 2319), *ps*, *wait*

93647 XBD [Section 3.204](#) (on page 65), [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 215), [<signal.h>](#)

93648 XSH *kill()*

#### 93649 **CHANGE HISTORY**

93650 First released in Issue 2.

#### 93651 **Issue 6**

93652 The obsolescent versions of the SYNOPSIS are turned into non-obsolescent features of the XSI  
93653 option, corresponding to a similar change in the *trap* special built-in.

#### 93654 **Issue 7**

93655 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

93656 **NAME**

93657       lex — generate programs for lexical tasks (DEVELOPMENT)

93658 **SYNOPSIS**93659 CD       lex [-t] [-n|-v] [*file...*]93660 **DESCRIPTION**

93661       The *lex* utility shall generate C programs to be used in lexical processing of character input, and  
 93662       that can be used as an interface to *yacc*. The C programs shall be generated from *lex* source code  
 93663       and conform to the ISO C standard, without depending on any undefined, unspecified, or  
 93664       implementation-defined behavior, except in cases where the code is copied directly from the  
 93665       supplied source, or in cases that are documented by the implementation. Usually, the *lex* utility  
 93666       shall write the program it generates to the file **lex.yy.c**; the state of this file is unspecified if *lex*  
 93667       exits with a non-zero exit status. See the EXTENDED DESCRIPTION section for a complete  
 93668       description of the *lex* input language.

93669 **OPTIONS**93670       The *lex* utility shall conform to XBD [Section 12.2](#) (on page 215), except for Guideline 9.

93671       The following options shall be supported:

- 93672       **-n**       Suppress the summary of statistics usually written with the **-v** option. If no table  
 93673       sizes are specified in the *lex* source code and the **-v** option is not specified, then **-n**  
 93674       is implied.
- 93675       **-t**       Write the resulting program to standard output instead of **lex.yy.c**.
- 93676       **-v**       Write a summary of *lex* statistics to the standard output. (See the discussion of *lex*  
 93677       table sizes in [Definitions in lex](#) (on page 2852).) If the **-t** option is specified and **-n**  
 93678       is not specified, this report shall be written to standard error. If table sizes are  
 93679       specified in the *lex* source code, and if the **-n** option is not specified, the **-v** option  
 93680       may be enabled.

93681 **OPERANDS**

93682       The following operand shall be supported:

- 93683       *file*       A pathname of an input file. If more than one such *file* is specified, all files shall be  
 93684       concatenated to produce a single *lex* program. If no *file* operands are specified, or if  
 93685       a *file* operand is **'-'**, the standard input shall be used.

93686 **STDIN**

93687       The standard input shall be used if no *file* operands are specified, or if a *file* operand is **'-'**. See  
 93688       INPUT FILES.

93689 **INPUT FILES**

93690       The input files shall be text files containing *lex* source code, as described in the EXTENDED  
 93691       DESCRIPTION section.

93692 **ENVIRONMENT VARIABLES**93693       The following environment variables shall affect the execution of *lex*:

- 93694       **LANG**       Provide a default value for the internationalization variables that are unset or null.  
 93695       (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 93696       variables used to determine the values of locale categories.)
- 93697       **LC\_ALL**      If set to a non-empty string value, override the values of all the other  
 93698       internationalization variables.

- 93699 *LC\_COLLATE*
- 93700 Determine the locale for the behavior of ranges, equivalence classes, and multi-
- 93701 character collating elements within regular expressions. If this variable is not set to
- 93702 the POSIX locale, the results are unspecified.
- 93703 *LC\_CTYPE*
- 93704 Determine the locale for the interpretation of sequences of bytes of text data as
- 93705 characters (for example, single-byte as opposed to multi-byte characters in
- 93706 arguments and input files), and the behavior of character classes within regular
- 93707 expressions. If this variable is not set to the POSIX locale, the results are
- unspecified.
- 93708 *LC\_MESSAGES*
- 93709 Determine the locale that should be used to affect the format and contents of
- 93710 diagnostic messages written to standard error.
- 93711 XSI *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.
- 93712 **ASYNCHRONOUS EVENTS**
- 93713 Default.
- 93714 **STDOUT**
- 93715 If the **-t** option is specified, the text file of C source code output of *lex* shall be written to
- 93716 standard output.
- 93717 If the **-t** option is not specified:
- 93718 • Implementation-defined informational, error, and warning messages concerning the
  - 93719 contents of *lex* source code input shall be written to either the standard output or standard
  - 93720 error.
  - 93721 • If the **-v** option is specified and the **-n** option is not specified, *lex* statistics shall also be
  - 93722 written to either the standard output or standard error, in an implementation-defined
  - 93723 format. These statistics may also be generated if table sizes are specified with a '**%**'
  - 93724 operator in the *Definitions* section, as long as the **-n** option is not specified.
- 93725 **STDERR**
- 93726 If the **-t** option is specified, implementation-defined informational, error, and warning messages
- 93727 concerning the contents of *lex* source code input shall be written to the standard error.
- 93728 If the **-t** option is not specified:
- 93729 1. Implementation-defined informational, error, and warning messages concerning the
  - 93730 contents of *lex* source code input shall be written to either the standard output or
  - 93731 standard error.
  - 93732 2. If the **-v** option is specified and the **-n** option is not specified, *lex* statistics shall also be
  - 93733 written to either the standard output or standard error, in an implementation-defined
  - 93734 format. These statistics may also be generated if table sizes are specified with a '**%**'
  - 93735 operator in the *Definitions* section, as long as the **-n** option is not specified.
- 93736 **OUTPUT FILES**
- 93737 A text file containing C source code shall be written to **lex.yy.c**, or to the standard output if the **-t**
- 93738 option is present.
- 93739 **EXTENDED DESCRIPTION**
- 93740 Each input file shall contain *lex* source code, which is a table of regular expressions with
- 93741 corresponding actions in the form of C program fragments.
- 93742 When **lex.yy.c** is compiled and linked with the *lex* library (using the **-ll** operand with *c99*), the



resulting program shall read character input from the standard input and shall partition it into strings that match the given expressions.

When an expression is matched, these actions shall occur:

- The input string that was matched shall be left in *yytext* as a null-terminated string; *yytext* shall either be an external character array or a pointer to a character string. As explained in [Definitions in lex](#) (on page 2852), the type can be explicitly selected using the **%array** or **%pointer** declarations, but the default is implementation-defined.
- The external **int** *yylen* shall be set to the length of the matching string.
- The expression's corresponding program fragment, or action, shall be executed.

During pattern matching, *lex* shall search the set of patterns for the single longest possible match. Among rules that match the same number of characters, the rule given first shall be chosen.

The general format of *lex* source shall be:

```

Definitions
%%
Rules
%%
UserSubroutines
```

The first "%%" is required to mark the beginning of the rules (regular expressions and actions); the second "%%" is required only if user subroutines follow.

Any line in the *Definitions* section beginning with a <blank> shall be assumed to be a C program fragment and shall be copied to the external definition area of the **lex.yy.c** file. Similarly, anything in the *Definitions* section included between delimiter lines containing only "%{" and "%}" shall also be copied unchanged to the external definition area of the **lex.yy.c** file.

Any such input (beginning with a <blank> or within "%{" and "%}" delimiter lines) appearing at the beginning of the *Rules* section before any rules are specified shall be written to **lex.yy.c** after the declarations of variables for the *yylex()* function and before the first line of code in *yylex()*. Thus, user variables local to *yylex()* can be declared here, as well as application code to execute upon entry to *yylex()*.

The action taken by *lex* when encountering any input beginning with a <blank> or within "%{" and "%}" delimiter lines appearing in the *Rules* section but coming after one or more rules is undefined. The presence of such input may result in an erroneous definition of the *yylex()* function.

C-language code in the input shall not contain C-language trigraphs. The C-language code within "%{" and "%}" delimiter lines shall not contain any lines consisting only of "%}", or only of "%%".

## Definitions in lex

*Definitions* appear before the first "%" delimiter. Any line in this section not contained between "%{" and "%}" lines and not beginning with a <blank> shall be assumed to define a *lex* substitution string. The format of these lines shall be:

*name substitute*

If a *name* does not meet the requirements for identifiers in the ISO C standard, the result is undefined. The string *substitute* shall replace the string {*name*} when it is used in a rule. The *name* string shall be recognized in this context only when the braces are provided and when it does not appear within a bracket expression or within double-quotes.

In the *Definitions* section, any line beginning with a <percent-sign> ('%') character and followed by an alphanumeric word beginning with either 's' or 'S' shall define a set of start conditions. Any line beginning with a '%' followed by a word beginning with either 'x' or 'X' shall define a set of exclusive start conditions. When the generated scanner is in a %s state, patterns with no state specified shall be also active; in a %x state, such patterns shall not be active. The rest of the line, after the first word, shall be considered to be one or more <blank>-separated names of start conditions. Start condition names shall be constructed in the same way as definition names. Start conditions can be used to restrict the matching of regular expressions to one or more states as described in [Regular Expressions in lex](#) (on page 2853).

Implementations shall accept either of the following two mutually-exclusive declarations in the *Definitions* section:

**%array**      Declare the type of *yytext* to be a null-terminated character array.

**%pointer**    Declare the type of *yytext* to be a pointer to a null-terminated character string.

The default type of *yytext* is implementation-defined. If an application refers to *yytext* outside of the scanner source file (that is, via an **extern**), the application shall include the appropriate **%array** or **%pointer** declaration in the scanner source file.

Implementations shall accept declarations in the *Definitions* section for setting certain internal table sizes. The declarations are shown in the following table.

**Table 4-10** Table Size Declarations in *lex*

| Declaration | Description                        | Minimum Value |
|-------------|------------------------------------|---------------|
| %p <i>n</i> | Number of positions                | 2 500         |
| %n <i>n</i> | Number of states                   | 500           |
| %a <i>n</i> | Number of transitions              | 2 000         |
| %e <i>n</i> | Number of parse tree nodes         | 1 000         |
| %k <i>n</i> | Number of packed character classes | 1 000         |
| %o <i>n</i> | Size of the output array           | 3 000         |

In the table, *n* represents a positive decimal integer, preceded by one or more <blank> characters. The exact meaning of these table size numbers is implementation-defined. The implementation shall document how these numbers affect the *lex* utility and how they are related to any output that may be generated by the implementation should limitations be encountered during the execution of *lex*. It shall be possible to determine from this output which of the table size values needs to be modified to permit *lex* to successfully generate tables for the input language. The values in the column Minimum Value represent the lowest values conforming implementations shall provide.

**Rules in lex**

The rules in *lex* source files are a table in which the left column contains regular expressions and the right column contains actions (C program fragments) to be executed when the expressions are recognized.

```
ERE action
ERE action
...
```

The extended regular expression (ERE) portion of a row shall be separated from *action* by one or more <blank> characters. A regular expression containing <blank> characters shall be recognized under one of the following conditions:

- The entire expression appears within double-quotes.
- The <blank> characters appear within double-quotes or square brackets.
- Each <blank> is preceded by a <backslash> character.

**User Subroutines in lex**

Anything in the user subroutines section shall be copied to **lex.yy.c** following *yylex()*.

**Regular Expressions in lex**

The *lex* utility shall support the set of extended regular expressions (see XBD [Section 9.4](#), on page 188), with the following additions and exceptions to the syntax:

"..." Any string enclosed in double-quotes shall represent the characters within the double-quotes as themselves, except that <backslash>-escapes (which appear in the following table) shall be recognized. Any <backslash>-escape sequence shall be terminated by the closing quote. For example, "\01"1" represents a single string: the octal value 1 followed by the character '1'.

<state>*r*, <state1, state2, ...>*r*

The regular expression *r* shall be matched only when the program is in one of the start conditions indicated by *state*, *state1*, and so on; see [Actions in lex](#) (on page 2855). (As an exception to the typographical conventions of the rest of this volume of POSIX.1-2008, in this case <state> does not represent a metavariable, but the literal angle-bracket characters surrounding a symbol.) The start condition shall be recognized as such only at the beginning of a regular expression.

*r*/*x*

The regular expression *r* shall be matched only if it is followed by an occurrence of regular expression *x* (*x* is the instance of trailing context, further defined below). The token returned in *yytext* shall only match *r*. If the trailing portion of *r* matches the beginning of *x*, the result is unspecified. The *r* expression cannot include further trailing context or the '\$' (match-end-of-line) operator; *x* cannot include the '^' (match-beginning-of-line) operator, nor trailing context, nor the '\$' operator. That is, only one occurrence of trailing context is allowed in a *lex* regular expression, and the '^' operator only can be used at the beginning of such an expression.

{*name*}

When *name* is one of the substitution symbols from the *Definitions* section, the string, including the enclosing braces, shall be replaced by the *substitute* value. The *substitute* value shall be treated in the extended regular expression as if it were enclosed in parentheses. No substitution shall occur if {*name*} occurs within a bracket expression or within double-quotes.

Within an ERE, a <backslash> character shall be considered to begin an escape sequence as specified in the table in XBD [Chapter 5](#) (on page 121) ('\\', '\a', '\b', '\f', '\n', '\r', '\t', '\v'). In addition, the escape sequences in the following table shall be recognized.

A literal <newline> cannot occur within an ERE; the escape sequence '\n' can be used to represent a <newline>. A <newline> shall not be matched by a period operator.

**Table 4-11** Escape Sequences in *lex*

| Escape Sequence | Description                                                                                                                                                                                                             | Meaning                                                                                                                                                                                                                         |
|-----------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| \digits         | A <backslash> character followed by the longest sequence of one, two, or three octal-digit characters (01234567). If all of the digits are 0 (that is, representation of the NUL character), the behavior is undefined. | The character whose encoding is represented by the one, two, or three-digit octal integer. Multi-byte characters require multiple, concatenated escape sequences of this type, including the leading <backslash> for each byte. |
| \xdigits        | A <backslash> character followed by the longest sequence of hexadecimal-digit characters (01234567abcdefABCDEF). If all of the digits are 0 (that is, representation of the NUL character), the behavior is undefined.  | The character whose encoding is represented by the hexadecimal integer.                                                                                                                                                         |
| \c              | A <backslash> character followed by any character not described in this table or in the table in XBD <a href="#">Chapter 5</a> (on page 121) ('\\', '\a', '\b', '\f', '\n', '\r', '\t', '\v').                          | The character 'c', unchanged.                                                                                                                                                                                                   |

**Note:** If a '\x' sequence needs to be immediately followed by a hexadecimal digit character, a sequence such as "\x1"1" can be used, which represents a character containing the value 1, followed by the character '1'.

The order of precedence given to extended regular expressions for *lex* differs from that specified in XBD [Section 9.4](#) (on page 188). The order of precedence for *lex* shall be as shown in the following table, from high to low.

**Note:** The escaped characters entry is not meant to imply that these are operators, but they are included in the table to show their relationships to the true operators. The start condition, trailing context, and anchoring notations have been omitted from the table because of the placement restrictions described in this section; they can only appear at the beginning or ending of an ERE.

Table 4-12 ERE Precedence in *lex*

| Extended Regular Expression       | Precedence            |
|-----------------------------------|-----------------------|
| collation-related bracket symbols | [ = ] [ : : ] [ . . ] |
| escaped characters                | \<special character>  |
| bracket expression                | [ ]                   |
| quoting                           | " . . . "             |
| grouping                          | ( )                   |
| definition                        | { name }              |
| single-character RE duplication   | * + ?                 |
| concatenation                     |                       |
| interval expression               | { m , n }             |
| alternation                       |                       |

The ERE anchoring operators '*^*' and '*\$*' do not appear in the table. With *lex* regular expressions, these operators are restricted in their use: the '*^*' operator can only be used at the beginning of an entire regular expression, and the '*\$*' operator only at the end. The operators apply to the entire regular expression. Thus, for example, the pattern "*(^abc) | (def\$)*" is undefined; it can instead be written as two separate rules, one with the regular expression "*^abc*" and one with "*def\$*", which share a common action via the special '*|*' action (see below). If the pattern were written "*^abc|def\$*", it would match either "*abc*" or "*def*" on a line by itself.

Unlike the general ERE rules, embedded anchoring is not allowed by most historical *lex* implementations. An example of embedded anchoring would be for patterns such as "*(^| )foo( |\$)*" to match "*foo*" when it exists as a complete word. This functionality can be obtained using existing *lex* features:

```
^foo/[\n] |
" foo"/[\n] /* Found foo as a separate word. */
```

Note also that '*\$*' is a form of trailing context (it is equivalent to "*/\n*") and as such cannot be used with regular expressions containing another instance of the operator (see the preceding discussion of trailing context).

The additional regular expressions trailing-context operator '*/'*' can be used as an ordinary character if presented within double-quotes, "*/"*"; preceded by a <backslash>, "*\"*"; or within a bracket expression, "*[/]*". The start-condition '*<*' and '*>*' operators shall be special only in a start condition at the beginning of a regular expression; elsewhere in the regular expression they shall be treated as ordinary characters.

### Actions in *lex*

The action to be taken when an ERE is matched can be a C program fragment or the special actions described below; the program fragment can contain one or more C statements, and can also include special actions. The empty C statement '*;*' shall be a valid action; any string in the *lex.yy.c* input that matches the pattern portion of such a rule is effectively ignored or skipped. However, the absence of an action shall not be valid, and the action *lex* takes in such a condition is undefined.

The specification for an action, including C statements and special actions, can extend across several lines if enclosed in braces:

```
ERE <one or more blanks> { program statement
 program statement }
```

93949 The program statements shall not contain unbalanced curly brace preprocessing tokens.

93950 The default action when a string in the input to a **lex.yy.c** program is not matched by any  
 93951 expression shall be to copy the string to the output. Because the default behavior of a program  
 93952 generated by *lex* is to read the input and copy it to the output, a minimal *lex* source program that  
 93953 has just "%%" shall generate a C program that simply copies the input to the output unchanged.

93954 Four special actions shall be available:

93955 | ECHO; REJECT; BEGIN

93956 | The action ' | ' means that the action for the next rule is the action for this rule.  
 93957 Unlike the other three actions, ' | ' cannot be enclosed in braces or be  
 93958 <semicolon>-terminated; the application shall ensure that it is specified alone, with  
 93959 no other actions.

93960 **ECHO;** Write the contents of the string *yytext* on the output.

93961 **REJECT;** Usually only a single expression is matched by a given string in the input.  
 93962 **REJECT** means "continue to the next expression that matches the current input",  
 93963 and shall cause whatever rule was the second choice after the current rule to be  
 93964 executed for the same input. Thus, multiple rules can be matched and executed for  
 93965 one input string or overlapping input strings. For example, given the regular  
 93966 expressions "xyz" and "xy" and the input "xyz", usually only the regular  
 93967 expression "xyz" would match. The next attempted match would start after **z**. If  
 93968 the last action in the "xyz" rule is **REJECT**, both this rule and the "xy" rule  
 93969 would be executed. The **REJECT** action may be implemented in such a fashion that  
 93970 flow of control does not continue after it, as if it were equivalent to a **goto** to  
 93971 another part of *yylex()*. The use of **REJECT** may result in somewhat larger and  
 93972 slower scanners.

93973 **BEGIN** The action:

93974 BEGIN *newstate*;

93975 switches the state (start condition) to *newstate*. If the string *newstate* has not been  
 93976 declared previously as a start condition in the *Definitions* section, the results are  
 93977 unspecified. The initial state is indicated by the digit '0' or the token **INITIAL**.

93978 The functions or macros described below are accessible to user code included in the *lex* input. It  
 93979 is unspecified whether they appear in the C code output of *lex*, or are accessible only through the  
 93980 **-ll** operand to *c99* (the *lex* library).

93981 **int yylex(void)**  
 93982 Performs lexical analysis on the input; this is the primary function generated by the *lex*  
 93983 utility. The function shall return zero when the end of input is reached; otherwise, it shall  
 93984 return non-zero values (tokens) determined by the actions that are selected.

93985 **int yymore(void)**  
 93986 When called, indicates that when the next input string is recognized, it is to be appended to  
 93987 the current value of *yytext* rather than replacing it; the value in *yyleng* shall be adjusted  
 93988 accordingly.

93989 **int yyless(int n)**  
 93990 Retains *n* initial characters in *yytext*, NUL-terminated, and treats the remaining characters as  
 93991 if they had not been read; the value in *yyleng* shall be adjusted accordingly.

**int input(void)**

Returns the next character from the input, or zero on end-of-file. It shall obtain input from the stream pointer *yyin*, although possibly via an intermediate buffer. Thus, once scanning has begun, the effect of altering the value of *yyin* is undefined. The character read shall be removed from the input stream of the scanner without any processing by the scanner.

**int unput(int c)**

Returns the character 'c' to the input; *yytext* and *yylen* are undefined until the next expression is matched. The result of using *unput()* for more characters than have been input is unspecified.

The following functions shall appear only in the *lex* library accessible through the **-ll** operand; they can therefore be redefined by a conforming application:

**int yywrap(void)**

Called by *yylex()* at end-of-file; the default *yywrap()* shall always return 1. If the application requires *yylex()* to continue processing with another source of input, then the application can include a function *yywrap()*, which associates another file with the external variable **FILE \* yyin** and shall return a value of zero.

**int main(int argc, char \*argv[])**

Calls *yylex()* to perform lexical analysis, then exits. The user code can contain *main()* to perform application-specific operations, calling *yylex()* as applicable.

Except for *input()*, *unput()*, and *main()*, all external and static names generated by *lex* shall begin with the prefix **yy** or **YY**.

**EXIT STATUS**

The following exit values shall be returned:

- 0 Successful completion.
- >0 An error occurred.

**CONSEQUENCES OF ERRORS**

Default.

**APPLICATION USAGE**

Conforming applications are warned that in the *Rules* section, an ERE without an action is not acceptable, but need not be detected as erroneous by *lex*. This may result in compilation or runtime errors.

The purpose of *input()* is to take characters off the input stream and discard them as far as the lexical analysis is concerned. A common use is to discard the body of a comment once the beginning of a comment is recognized.

The *lex* utility is not fully internationalized in its treatment of regular expressions in the *lex* source code or generated lexical analyzer. It would seem desirable to have the lexical analyzer interpret the regular expressions given in the *lex* source according to the environment specified when the lexical analyzer is executed, but this is not possible with the current *lex* technology. Furthermore, the very nature of the lexical analyzers produced by *lex* must be closely tied to the lexical requirements of the input language being described, which is frequently locale-specific anyway. (For example, writing an analyzer that is used for French text is not automatically useful for processing other languages.)

**EXAMPLES**

The following is an example of a *lex* program that implements a rudimentary scanner for a Pascal-like syntax:

```
%{
/* Need this for the call to atof() below. */
#include <math.h>
/* Need this for printf(), fopen(), and stdin below. */
#include <stdio.h>
}%

DIGIT [0-9]
ID [a-z][a-z0-9]*

%%

{DIGIT}+ {
 printf("An integer: %s (%d)\n", yytext,
 atoi(yytext));
}

{DIGIT}+"."{DIGIT}* {
 printf("A float: %s (%g)\n", yytext,
 atof(yytext));
}

if|then|begin|end|procedure|function {
 printf("A keyword: %s\n", yytext);
}

{ID} printf("An identifier: %s\n", yytext);

"+"|"-"|"*"|"/" printf("An operator: %s\n", yytext);

"{[^]\n}*" /* Eat up one-line comments. */

[\t\n]+ /* Eat up white space. */

. printf("Unrecognized character: %s\n", yytext);

%%

int main(int argc, char *argv[])
{
 ++argv, --argc; /* Skip over program name. */
 if (argc > 0)
 yyin = fopen(argv[0], "r");
 else
 yyin = stdin;

 yylex();
}
```

**RATIONALE**

Even though the `-c` option and references to the C language are retained in this description, *lex* may be generalized to other languages, as was done at one time for EFL, the Extended FORTRAN Language. Since the *lex* input specification is essentially language-independent, versions of this utility could be written to produce Ada, Modula-2, or Pascal code, and there are known historical implementations that do so.



The current description of *lex* bypasses the issue of dealing with internationalized EREs in the *lex* source code or generated lexical analyzer. If it follows the model used by *awk* (the source code is assumed to be presented in the POSIX locale, but input and output are in the locale specified by the environment variables), then the tables in the lexical analyzer produced by *lex* would interpret EREs specified in the *lex* source in terms of the environment variables specified when *lex* was executed. The desired effect would be to have the lexical analyzer interpret the EREs given in the *lex* source according to the environment specified when the lexical analyzer is executed, but this is not possible with the current *lex* technology.

The description of octal and hexadecimal-digit escape sequences agrees with the ISO C standard use of escape sequences.

Earlier versions of this standard allowed for implementations with bytes other than eight bits, but this has been modified in this version.

There is no detailed output format specification. The observed behavior of *lex* under four different historical implementations was that none of these implementations consistently reported the line numbers for error and warning messages. Furthermore, there was a desire that *lex* be allowed to output additional diagnostic messages. Leaving message formats unspecified avoids these formatting questions and problems with internationalization.

Although the `%x` specifier for *exclusive* start conditions is not historical practice, it is believed to be a minor change to historical implementations and greatly enhances the usability of *lex* programs since it permits an application to obtain the expected functionality with fewer statements.

The `%array` and `%pointer` declarations were added as a compromise between historical systems. The System V-based *lex* copies the matched text to a *yytext* array. The *flex* program, supported in BSD and GNU systems, uses a pointer. In the latter case, significant performance improvements are available for some scanners. Most historical programs should require no change in porting from one system to another because the string being referenced is null-terminated in both cases. (The method used by *flex* in its case is to null-terminate the token in place by remembering the character that used to come right after the token and replacing it before continuing on to the next scan.) Multi-file programs with external references to *yytext* outside the scanner source file should continue to operate on their historical systems, but would require one of the new declarations to be considered strictly portable.

The description of EREs avoids unnecessary duplication of ERE details because their meanings within a *lex* ERE are the same as that for the ERE in this volume of POSIX.1-2008.

The reason for the undefined condition associated with text beginning with a <blank> or within "`%{`" and "`%}`" delimiter lines appearing in the *Rules* section is historical practice. Both the BSD and System V *lex* copy the indented (or enclosed) input in the *Rules* section (except at the beginning) to unreachable areas of the *yylex()* function (the code is written directly after a *break* statement). In some cases, the System V *lex* generates an error message or a syntax error, depending on the form of indented input.

The intention in breaking the list of functions into those that may appear in `lex.yy.c` *versus* those that only appear in `libl.a` is that only those functions in `libl.a` can be reliably redefined by a conforming application.

The descriptions of standard output and standard error are somewhat complicated because historical *lex* implementations chose to issue diagnostic messages to standard output (unless `-t` was given). POSIX.1-2008 allows this behavior, but leaves an opening for the more expected behavior of using standard error for diagnostics. Also, the System V behavior of writing the statistics when any table sizes are given is allowed, while BSD-derived systems can avoid it. The

programmer can always precisely obtain the desired results by using either the `-t` or `-n` options.

The OPERANDS section does not mention the use of `-` as a synonym for standard input; not all historical implementations support such usage for any of the *file* operands.

A description of the *translation table* was deleted from early proposals because of its relatively low usage in historical applications.

The change to the definition of the *input()* function that allows buffering of input presents the opportunity for major performance gains in some applications.

The following examples clarify the differences between *lex* regular expressions and regular expressions appearing elsewhere in this volume of POSIX.1-2008. For regular expressions of the form `"r/x"`, the string matching *r* is always returned; confusion may arise when the beginning of *x* matches the trailing portion of *r*. For example, given the regular expression `"a*b/cc"` and the input `"aaabcc"`, *yytext* would contain the string `"aaab"` on this match. But given the regular expression `"x*/xy"` and the input `"xxxxy"`, the token `xxx`, not `xx`, is returned by some implementations because `xxx` matches `"x*"`.

In the rule `"ab*/bc"`, the `"b*"` at the end of *r* extends *r*'s match into the beginning of the trailing context, so the result is unspecified. If this rule were `"ab/bc"`, however, the rule matches the text `"ab"` when it is followed by the text `"bc"`. In this latter case, the matching of *r* cannot extend into the beginning of *x*, so the result is specified.

#### FUTURE DIRECTIONS

None.

#### SEE ALSO

*c99*, *ed*, *yacc*

XBD [Chapter 5](#) (on page 121), [Chapter 8](#) (on page 173), [Chapter 9](#) (on page 181), [Section 12.2](#) (on page 215)

#### CHANGE HISTORY

First released in Issue 2.

#### Issue 6

This utility is marked as part of the C-Language Development Utilities option.

The obsolescent `-c` option is removed.

The normative text is reworded to avoid use of the term “must” for application requirements.

IEEE Std 1003.1-2001/Cor 2-2004, item XCU/TC2/D6/14 is applied, removing text describing behavior on systems with bytes consisting of more than eight bits.

#### Issue 7

Austin Group Interpretation 1003.1-2001 #190 is applied, clarifying the requirements for generated code to conform to the ISO C standard.

Austin Group Interpretation 1003.1-2001 #191 is applied, clarifying the handling of C-language trigraphs and curly brace preprocessing tokens.

SD5-XCU-ERN-6 is applied, clarifying that Guideline 9 of the Utility Syntax Guidelines does not apply.

SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

94165 **NAME**94166 link — call *link()* function94167 **SYNOPSIS**94168 XSI `link file1 file2`94169 **DESCRIPTION**94170 The *link* utility shall perform the function call:94171 `link(file1, file2);`94172 A user may need appropriate privileges to invoke the *link* utility.94173 **OPTIONS**

94174 None.

94175 **OPERANDS**

94176 The following operands shall be supported:

94177 *file1* The pathname of an existing file.94178 *file2* The pathname of the new directory entry to be created.94179 **STDIN**

94180 Not used.

94181 **INPUT FILES**

94182 Not used.

94183 **ENVIRONMENT VARIABLES**94184 The following environment variables shall affect the execution of *link*:94185 *LANG* Provide a default value for the internationalization variables that are unset or null.  
94186 (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
94187 variables used to determine the values of locale categories.)94188 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
94189 internationalization variables.94190 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
94191 characters (for example, single-byte as opposed to multi-byte characters in  
94192 arguments).94193 *LC\_MESSAGES*  
94194 Determine the locale that should be used to affect the format and contents of  
94195 diagnostic messages written to standard error.94196 *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.94197 **ASYNCHRONOUS EVENTS**

94198 Default.

94199 **STDOUT**

94200 None.

94201 **STDERR**

94202 The standard error shall be used only for diagnostic messages.

**94203 OUTPUT FILES**

94204       None.

**94205 EXTENDED DESCRIPTION**

94206       None.

**94207 EXIT STATUS**

94208       The following exit values shall be returned:

94209       0   Successful completion.

94210       >0  An error occurred.

**94211 CONSEQUENCES OF ERRORS**

94212       Default.

**94213 APPLICATION USAGE**

94214       None.

**94215 EXAMPLES**

94216       None.

**94217 RATIONALE**

94218       None.

**94219 FUTURE DIRECTIONS**

94220       None.

**94221 SEE ALSO**

94222       *ln*, *unlink*

94223       XBD [Chapter 8](#) (on page 173)

94224       XSH [link\(\)](#)

**94225 CHANGE HISTORY**

94226       First released in Issue 5.

**NAME**

ln — link files

**SYNOPSIS**

ln [-fs] [-L|-P] *source\_file target\_file*

ln [-fs] [-L|-P] *source\_file... target\_dir*

**DESCRIPTION**

In the first synopsis form, the *ln* utility shall create a new directory entry (link) at the destination path specified by the *target\_file* operand. If the *-s* option is specified, a symbolic link shall be created for the file specified by the *source\_file* operand. This first synopsis form shall be assumed when the final operand does not name an existing directory; if more than two operands are specified and the final is not an existing directory, an error shall result.

In the second synopsis form, the *ln* utility shall create a new directory entry (link), or if the *-s* option is specified a symbolic link, for each file specified by a *source\_file* operand, at a destination path in the existing directory named by *target\_dir*.

If the last operand specifies an existing file of a type not specified by the System Interfaces volume of POSIX.1-2008, the behavior is implementation-defined.

The corresponding destination path for each *source\_file* shall be the concatenation of the target directory pathname, a <slash> character if the target directory pathname did not end in a <slash>, and the last pathname component of the *source\_file*. The second synopsis form shall be assumed when the final operand names an existing directory.

For each *source\_file*:

1. If the destination path exists and was created by a previous step, it is unspecified whether *ln* shall write a diagnostic message to standard error, do nothing more with the current *source\_file*, and go on to any remaining *source\_files*; or will continue processing the current *source\_file*. If the destination path exists:

- a. If the *-f* option is not specified, *ln* shall write a diagnostic message to standard error, do nothing more with the current *source\_file*, and go on to any remaining *source\_files*.
- b. If *destination* names the same directory entry as the current *source\_file* *ln* shall write a diagnostic message to standard error, do nothing more with the current *source\_file*, and go on to any remaining *source\_files*.
- c. Actions shall be performed equivalent to the *unlink()* function defined in the System Interfaces volume of POSIX.1-2008, called using *destination* as the *path* argument. If this fails for any reason, *ln* shall write a diagnostic message to standard error, do nothing more with the current *source\_file*, and go on to any remaining *source\_files*.

2. If the *-s* option is specified, actions shall be performed equivalent to the *symlink()* function with *source\_file* as the *path1* argument and the destination path as the *path2* argument. The *ln* utility shall do nothing more with *source\_file* and shall go on to any remaining files.

3. If *source\_file* is a symbolic link:

- a. If the *-P* option is in effect, actions shall be performed equivalent to the *linkat()* function with *source\_file* as the *path1* argument, the destination path as the *path2* argument, *AT\_FDCWD* as the *fd1* and *fd2* arguments, and zero as the *flag* argument.

- b. If the **-L** option is in effect, actions shall be performed equivalent to the *linkat()* function with *source\_file* as the *path1* argument, the destination path as the *path2* argument, **AT\_FDCWD** as the *fd1* and *fd2* arguments, and **AT\_SYMLINK\_FOLLOW** as the *flag* argument.

The *ln* utility shall do nothing more with *source\_file* and shall go on to any remaining files.

4. Actions shall be performed equivalent to the *link()* function defined in the System Interfaces volume of POSIX.1-2008 using *source\_file* as the *path1* argument, and the destination path as the *path2* argument.

## OPTIONS

The *ln* utility shall conform to XBD [Section 12.2](#) (on page 215).

The following options shall be supported:

- f** Force existing destination pathnames to be removed to allow the link.
- L** For each *source\_file* operand that names a file of type symbolic link, create a (hard) link to the file referenced by the symbolic link.
- P** For each *source\_file* operand that names a file of type symbolic link, create a (hard) link to the symbolic link itself.
- s** Create symbolic links instead of hard links. If the **-s** option is specified, the **-L** and **-P** options shall be silently ignored.

Specifying more than one of the mutually-exclusive options **-L** and **-P** shall not be considered an error. The last option specified shall determine the behavior of the utility (unless the **-s** option causes it to be ignored).

If the **-s** option is not specified and neither a **-L** nor a **-P** option is specified, it is implementation-defined which of the **-L** and **-P** options will be used as the default.

## OPERANDS

The following operands shall be supported:

- source\_file* A pathname of a file to be linked. If the **-s** option is specified, no restrictions on the type of file or on its existence shall be made. If the **-s** option is not specified, whether a directory can be linked is implementation-defined.
- target\_file* The pathname of the new directory entry to be created.
- target\_dir* A pathname of an existing directory in which the new directory entries are created.

## STDIN

Not used.

## INPUT FILES

None.

## ENVIRONMENT VARIABLES

The following environment variables shall affect the execution of *ln*:

- LANG** Provide a default value for the internationalization variables that are unset or null. (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization variables used to determine the values of locale categories.)
- LC\_ALL** If set to a non-empty string value, override the values of all the other internationalization variables.

|       |                               |                                                                                                                                                                                                                                                                                                                                                   |
|-------|-------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 94313 | <i>LC_CTYPE</i>               | Determine the locale for the interpretation of sequences of bytes of text data as characters (for example, single-byte as opposed to multi-byte characters in arguments).                                                                                                                                                                         |
| 94314 |                               |                                                                                                                                                                                                                                                                                                                                                   |
| 94315 |                               |                                                                                                                                                                                                                                                                                                                                                   |
| 94316 | <i>LC_MESSAGES</i>            |                                                                                                                                                                                                                                                                                                                                                   |
| 94317 |                               | Determine the locale that should be used to affect the format and contents of diagnostic messages written to standard error.                                                                                                                                                                                                                      |
| 94318 |                               |                                                                                                                                                                                                                                                                                                                                                   |
| 94319 | XSI <i>NLSPATH</i>            | Determine the location of message catalogs for the processing of <i>LC_MESSAGES</i> .                                                                                                                                                                                                                                                             |
| 94320 | <b>ASYNCHRONOUS EVENTS</b>    |                                                                                                                                                                                                                                                                                                                                                   |
| 94321 |                               | Default.                                                                                                                                                                                                                                                                                                                                          |
| 94322 | <b>STDOUT</b>                 |                                                                                                                                                                                                                                                                                                                                                   |
| 94323 |                               | Not used.                                                                                                                                                                                                                                                                                                                                         |
| 94324 | <b>STDERR</b>                 |                                                                                                                                                                                                                                                                                                                                                   |
| 94325 |                               | The standard error shall be used only for diagnostic messages.                                                                                                                                                                                                                                                                                    |
| 94326 | <b>OUTPUT FILES</b>           |                                                                                                                                                                                                                                                                                                                                                   |
| 94327 |                               | None.                                                                                                                                                                                                                                                                                                                                             |
| 94328 | <b>EXTENDED DESCRIPTION</b>   |                                                                                                                                                                                                                                                                                                                                                   |
| 94329 |                               | None.                                                                                                                                                                                                                                                                                                                                             |
| 94330 | <b>EXIT STATUS</b>            |                                                                                                                                                                                                                                                                                                                                                   |
| 94331 |                               | The following exit values shall be returned:                                                                                                                                                                                                                                                                                                      |
| 94332 | 0                             | All the specified files were linked successfully.                                                                                                                                                                                                                                                                                                 |
| 94333 | >0                            | An error occurred.                                                                                                                                                                                                                                                                                                                                |
| 94334 | <b>CONSEQUENCES OF ERRORS</b> |                                                                                                                                                                                                                                                                                                                                                   |
| 94335 |                               | Default.                                                                                                                                                                                                                                                                                                                                          |
| 94336 | <b>APPLICATION USAGE</b>      |                                                                                                                                                                                                                                                                                                                                                   |
| 94337 |                               | None.                                                                                                                                                                                                                                                                                                                                             |
| 94338 | <b>EXAMPLES</b>               |                                                                                                                                                                                                                                                                                                                                                   |
| 94339 |                               | None.                                                                                                                                                                                                                                                                                                                                             |
| 94340 | <b>RATIONALE</b>              |                                                                                                                                                                                                                                                                                                                                                   |
| 94341 |                               | The CONSEQUENCES OF ERRORS section does not require <i>ln -f a b</i> to remove <i>b</i> if a subsequent link operation would fail.                                                                                                                                                                                                                |
| 94342 |                               |                                                                                                                                                                                                                                                                                                                                                   |
| 94343 |                               | Some historic versions of <i>ln</i> (including the one specified by the SVID) unlink the destination file, if it exists, by default. If the mode does not permit writing, these versions prompt for confirmation before attempting the unlink. In these versions the <i>-f</i> option causes <i>ln</i> not to attempt to prompt for confirmation. |
| 94344 |                               |                                                                                                                                                                                                                                                                                                                                                   |
| 94345 |                               |                                                                                                                                                                                                                                                                                                                                                   |
| 94346 |                               |                                                                                                                                                                                                                                                                                                                                                   |
| 94347 |                               | This allows <i>ln</i> to succeed in creating links when the target file already exists, even if the file itself is not writable (although the directory must be). Early proposals specified this functionality.                                                                                                                                   |
| 94348 |                               |                                                                                                                                                                                                                                                                                                                                                   |
| 94349 |                               | This volume of POSIX.1-2008 does not allow the <i>ln</i> utility to unlink existing destination paths by default for the following reasons:                                                                                                                                                                                                       |
| 94350 |                               |                                                                                                                                                                                                                                                                                                                                                   |
| 94351 |                               | • The <i>ln</i> utility has historically been used to provide locking for shell applications, a usage that is incompatible with <i>ln</i> unlinking the destination path by default. There was no corresponding technical advantage to adding this functionality.                                                                                 |
| 94352 |                               |                                                                                                                                                                                                                                                                                                                                                   |
| 94353 |                               |                                                                                                                                                                                                                                                                                                                                                   |

- This functionality gave *ln* the ability to destroy the link structure of files, which changes the historical behavior of *ln*.
- This functionality is easily replicated with a combination of *rm* and *ln*.
- It is not historical practice in many systems; BSD and BSD-derived systems do not support this behavior. Unfortunately, whichever behavior is selected can cause scripts written expecting the other behavior to fail.
- It is preferable that *ln* perform in the same manner as the *link()* function, which does not permit the target to exist already.

This volume of POSIX.1-2008 retains the *-f* option to provide support for shell scripts depending on the SVID semantics. It seems likely that shell scripts would not be written to handle prompting by *ln* and would therefore have specified the *-f* option.

The *-f* option is an undocumented feature of many historical versions of the *ln* utility, allowing linking to directories. These versions require modification.

Early proposals of this volume of POSIX.1-2008 also required a *-i* option, which behaved like the *-i* options in *cp* and *mv*, prompting for confirmation before unlinking existing files. This was not historical practice for the *ln* utility and has been omitted.

The *-L* and *-P* options allow for implementing both common behaviors of the *ln* utility. Earlier versions of this standard did not specify these options and required the behavior now described for the *-L* option. Many systems by default or as an alternative provided a non-conforming *ln* utility with the behavior now described for the *-P* option. Since applications could not rely on *ln* following links in practice, the *-L* and *-P* options were added to specify the desired behavior for the application.

The *-L* and *-P* options are ignored when *-s* is specified in order to allow an alias to be created to alter the default behavior when creating hard links (for example, *alias ln='ln -L'*). They serve no purpose when *-s* is specified, since *source\_file* is then just a string to be used as the contents of the created symbolic link and need not exist as a file.

The specification ensures that *ln a a* with or without the *-f* option will not unlink the file *a*. Earlier versions of this standard were unclear in this case.

## FUTURE DIRECTIONS

None.

## SEE ALSO

*chmod*, *find*, *pax*, *rm*

XBD Chapter 8 (on page 173), Section 12.2 (on page 215)

XSH *link()*, *unlink()*

## CHANGE HISTORY

First released in Issue 2.

### Issue 6

The *ln* utility is updated to include symbolic link processing as defined in the IEEE P1003.2b draft standard.

### Issue 7

Austin Group Interpretations 1003.1-2001 #164, #168, and #169 are applied.

SD5-XCU-ERN-27 is applied, adding a new paragraph to the RATIONALE.

SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.



94397 The **-L** and **-P** options are added to make it implementation-defined whether the *ln* utility  
94398 follows symbolic links.  
94399 POSIX.1-2008, Technical Corrigendum 1, XCU/TC1-2008/0096 [136] is applied. +

94400 **NAME**

94401 locale — get locale-specific information

94402 **SYNOPSIS**

94403 locale [-a|-m]

94404 locale [-ck] name...

94405 **DESCRIPTION**

94406 The *locale* utility shall write information about the current locale environment, or all public  
 94407 locales, to the standard output. For the purposes of this section, a *public locale* is one provided by  
 94408 the implementation that is accessible to the application.

94409 When *locale* is invoked without any arguments, it shall summarize the current locale  
 94410 environment for each locale category as determined by the settings of the environment variables  
 94411 defined in XBD [Chapter 7](#) (on page 135).

94412 When invoked with operands, it shall write values that have been assigned to the keywords in  
 94413 the locale categories, as follows:

- 94414 • Specifying a keyword name shall select the named keyword and the category containing  
 94415 that keyword.
- 94416 • Specifying a category name shall select the named category and all keywords in that  
 94417 category.

94418 **OPTIONS**94419 The *locale* utility shall conform to XBD [Section 12.2](#) (on page 215).

94420 The following options shall be supported:

- 94421 **-a** Write information about all available public locales. The available locales shall  
 94422 include **POSIX**, representing the POSIX locale. The manner in which the  
 94423 implementation determines what other locales are available is implementation-  
 94424 defined.
- 94425 **-c** Write the names of selected locale categories; see the STDOUT section. The **-c**  
 94426 option increases readability when more than one category is selected (for example,  
 94427 via more than one keyword name or via a category name). It is valid both with  
 94428 and without the **-k** option.
- 94429 **-k** Write the names and values of selected keywords. The implementation may omit  
 94430 values for some keywords; see the OPERANDS section.
- 94431 **-m** Write names of available charmaps; see XBD [Section 6.1](#) (on page 125).

94432 **OPERANDS**

94433 The following operand shall be supported:

- 94434 *name* The name of a locale category as defined in XBD [Chapter 7](#) (on page 135), the name  
 94435 of a keyword in a locale category, or the reserved name **charmap**. The named  
 94436 category or keyword shall be selected for output. If a single *name* represents both a  
 94437 locale category name and a keyword name in the current locale, the results are  
 94438 unspecified. Otherwise, both category and keyword names can be specified as  
 94439 *name* operands, in any sequence. It is implementation-defined whether any  
 94440 keyword values are written for the categories *LC\_CTYPE* and *LC\_COLLATE*.

94441 **STDIN**  
 94442 Not used.

94443 **INPUT FILES**  
 94444 None.

94445 **ENVIRONMENT VARIABLES**  
 94446 The following environment variables shall affect the execution of *locale*:

94447 *LANG* Provide a default value for the internationalization variables that are unset or null.  
 94448 (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 94449 variables used to determine the values of locale categories.)

94450 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
 94451 internationalization variables.

94452 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
 94453 characters (for example, single-byte as opposed to multi-byte characters in  
 94454 arguments and input files).

94455 *LC\_MESSAGES*  
 94456 Determine the locale that should be used to affect the format and contents of  
 94457 diagnostic messages written to standard error.

94458 XSI *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

94459 XSI The application shall ensure that the *LANG*, *LC\_\**, and *NLSPATH* environment variables specify  
 94460 the current locale environment to be written out; they shall be used if the *-a* option is not  
 94461 specified.

94462 **ASYNCHRONOUS EVENTS**  
 94463 Default.

94464 **STDOUT**  
 94465 The *LANG* variable shall be written first using the format:  
 94466 "LANG=%s\n", <value>

94467 If *LANG* is not set or is an empty string, the value is the empty string.

94468 If *locale* is invoked without any options or operands, the names and values of the *LC\_\**  
 94469 environment variables described in this volume of POSIX.1-2008 shall be written to the standard  
 94470 output, one variable per line, and each line using the following format. Only those variables set  
 94471 in the environment and not overridden by *LC\_ALL* shall be written using this format:  
 94472 "%s=%s\n", <variable\_name>, <value>

94473 The names of those *LC\_\** variables associated with locale categories defined in this volume of  
 94474 POSIX.1-2008 that are not set in the environment or are overridden by *LC\_ALL* shall be written  
 94475 in the following format:  
 94476 "%s=\"%s\"\\n", <variable\_name>, <implied value>

94477 The <implied value> shall be the name of the locale that has been selected for that category by the  
 94478 implementation, based on the values in *LANG* and *LC\_ALL*, as described in XBD [Chapter 8](#) (on  
 94479 page 173).

94480 The <value> and <implied value> shown above shall be properly quoted for possible later reentry  
 94481 to the shell. The <value> shall not be quoted using double-quotes (so that it can be distinguished  
 94482 by the user from the <implied value> case, which always requires double-quotes).

The `LC_ALL` variable shall be written last, using the first format shown above. If it is not set, it shall be written as:

```
"LC_ALL=\n"
```

If any arguments are specified:

1. If the `-a` option is specified, the names of all the public locales shall be written, each in the following format:

```
"%s\n", <locale name>
```

2. If the `-c` option is specified, the names of all selected categories shall be written, each in the following format:

```
"%s\n", <category name>
```

If keywords are also selected for writing (see following items), the category name output shall precede the keyword output for that category.

If the `-c` option is not specified, the names of the categories shall not be written; only the keywords, as selected by the `<name>` operand, shall be written.

3. If the `-k` option is specified, the names and values of selected keywords shall be written. If a value is non-numeric and is not a compound keyword value, it shall be written in the following format:

```
"%s=\"%s\"\n", <keyword name>, <keyword value>
```

If a value is a non-numeric compound keyword value, it shall either be written in the format:

```
"%s=\"%s\"\n", <keyword name>, <keyword value>
```

where the `<keyword value>` is a single string of values separated by `<semicolon>` characters, or it shall be written in the format:

```
"%s=%s\n", <keyword name>, <keyword value>
```

where the `<keyword value>` is encoded as a set of strings, each enclosed in double-quotation-marks, separated by `<semicolon>` characters.

If the keyword was **charmap**, the name of the charmap (if any) that was specified via the `localedef -f` option when the locale was created shall be written, with the word **charmap** as `<keyword name>`.

If a value is numeric, it shall be written in one of the following formats:

```
"%s=%d\n", <keyword name>, <keyword value>
```

```
"%s=%c%o\n", <keyword name>, <escape character>, <keyword value>
```

```
"%s=%cx%x\n", <keyword name>, <escape character>, <keyword value>
```

where the `<escape character>` is that identified by the **escape\_char** keyword in the current locale; see XBD [Section 7.3](#) (on page 136).

Compound keyword values (list entries) shall be separated in the output by `<semicolon>` characters. When included in keyword values, the `<semicolon>`, `<backslash>`, double-quote, and any control character shall be preceded (escaped) with the escape character.

4. If the **-k** option is not specified, selected keyword values shall be written, each in the following format:

```
"%s\n", <keyword value>
```

If the keyword was **charmap**, the name of the charmap (if any) that was specified via the *localedef -f* option when the locale was created shall be written.

5. If the **-m** option is specified, then a list of all available charmaps shall be written, each in the format:

```
"%s\n", <charmap>
```

where *<charmap>* is in a format suitable for use as the option-argument to the *localedef -f* option.

## STDERR

The standard error shall be used only for diagnostic messages.

## OUTPUT FILES

None.

## EXTENDED DESCRIPTION

None.

## EXIT STATUS

The following exit values shall be returned:

- 0 All the requested information was found and output successfully.
- >0 An error occurred.

## CONSEQUENCES OF ERRORS

Default.

## APPLICATION USAGE

If the *LANG* environment variable is not set or set to an empty value, or one of the *LC\_\** environment variables is set to an unrecognized value, the actual locales assumed (if any) are implementation-defined as described in XBD [Chapter 8](#) (on page 173).

Implementations are not required to write out the actual values for keywords in the categories *LC\_CTYPE* and *LC\_COLLATE*; however, they must write out the categories (allowing an application to determine, for example, which character classes are available).

## EXAMPLES

In the following examples, the assumption is that locale environment variables are set as follows:

```
LANG=locale_x
LC_COLLATE=locale_y
```

The command *locale* would result in the following output:

```
LANG=locale_x
LC_CTYPE="locale_x"
LC_COLLATE=locale_y
LC_TIME="locale_x"
LC_NUMERIC="locale_x"
LC_MONETARY="locale_x"
LC_MESSAGES="locale_x"
LC_ALL=
```

The order of presentation of the categories is not specified by this volume of POSIX.1-2008.

The command:

```
LC_ALL=POSIX locale -ck decimal_point
```

would produce:

```
LC_NUMERIC
decimal_point="."
```

The following command shows an application of *locale* to determine whether a user-supplied response is affirmative:

```
if printf "%s\n" "$response" | grep -Eq "${locale yesexpr}"
then
 affirmative processing goes here
else
 non-affirmative processing goes here
fi
```

## RATIONALE

The output for categories *LC\_CTYPE* and *LC\_COLLATE* has been made implementation-defined because there is a questionable value in having a shell script receive an entire array of characters. It is also difficult to return a logical collation description, short of returning a complete *localedef* source.

The **-m** option was included to allow applications to query for the existence of charmaps. The output is a list of the charmaps (implementation-supplied and user-supplied, if any) on the system.

The **-c** option was included for readability when more than one category is selected (for example, via more than one keyword name or via a category name). It is valid both with and without the **-k** option.

The **charmap** keyword, which returns the name of the charmap (if any) that was used when the current locale was created, was included to allow applications needing the information to retrieve it.

According to XBD [Section 6.1](#) (on page 125), the standard requires that all supported locales must have the same encoding for <period> and <slash>, because these two characters are used within the locale-independent pathname resolution sequence. Therefore, it would be an error if *locale -a* listed both ASCII and EBCDIC-based locales, since those two encodings do not share the same representation for either <period> or <slash>. Any system that supports both environments would be expected to provide two POSIX locales, one in either codeset, where only the locales appropriate to the current environment can be visible at a time. In an XSI-compliant implementation, the *dd* utility is the only portable means for performing conversions between the two character sets.

## FUTURE DIRECTIONS

None.

## SEE ALSO

*localedef*

XBD [Section 6.1](#) (on page 125), [Chapter 7](#) (on page 135), [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 215)

94607 **CHANGE HISTORY**

94608 First released in Issue 4.

94609 **Issue 5**

94610 The FUTURE DIRECTIONS section is added.

94611 **Issue 6**

94612 The normative text is reworded to avoid use of the term “must” for application requirements.

94613 IEEE Std 1003.1-2001/Cor 1-2002, item XCU/TC1/D6/30 is applied, correcting an editorial error  
94614 in the STDOUT section.94615 **Issue 7**94616 Austin Group Interpretations 1003.1-2001 #017, #021, and #088 are applied, clarifying the  
94617 standard output for the **-k** option when *LANG* is not set or is an empty string.

94618 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

94619 POSIX.1-2008, Technical Corrigendum 1, XCU/TC1-2008/0097 [291] is applied.

+

94620 **NAME**

94621 localedef — define locale environment

94622 **SYNOPSIS**94623 localedef **[-c]** **[-f charmap]** **[-i sourcefile]** **[-u code\_set\_name]** *name*94624 **DESCRIPTION**

94625 The *localedef* utility shall convert source definitions for locale categories into a format usable by  
 94626 the functions and utilities whose operational behavior is determined by the setting of the locale  
 94627 environment variables defined in XBD [Chapter 7](#) (on page 135). It is implementation-defined  
 94628 whether users have the capability to create new locales, in addition to those supplied by the  
 94629 implementation. If the symbolic constant POSIX2\_LOCALEDEF is defined, the system supports  
 94630 XSI the creation of new locales. On XSI-conformant systems, the symbolic constant  
 94631 POSIX2\_LOCALEDEF shall be defined.

94632 The utility shall read source definitions for one or more locale categories belonging to the same  
 94633 locale from the file named in the **-i** option (if specified) or from standard input.

94634 The *name* operand identifies the target locale. The utility shall support the creation of *public*, or  
 94635 generally accessible locales, as well as *private*, or restricted-access locales. Implementations may  
 94636 restrict the capability to create or modify public locales to users with appropriate privileges.

94637 Each category source definition shall be identified by the corresponding environment variable  
 94638 name and terminated by an **END category-name** statement. The following categories shall be  
 94639 supported. In addition, the input may contain source for implementation-defined categories.

94640 *LC\_CTYPE* Defines character classification and case conversion.

94641 *LC\_COLLATE*  
 94642 Defines collation rules.

94643 *LC\_MONETARY*  
 94644 Defines the format and symbols used in formatting of monetary information.

94645 *LC\_NUMERIC*  
 94646 Defines the decimal delimiter, grouping, and grouping symbol for non-monetary  
 94647 numeric editing.

94648 *LC\_TIME* Defines the format and content of date and time information.

94649 *LC\_MESSAGES*  
 94650 Defines the format and values of affirmative and negative responses.

94651 **OPTIONS**

94652 The *localedef* utility shall conform to XBD [Section 12.2](#) (on page 215).

94653 The following options shall be supported:

94654 **-c** Create permanent output even if warning messages have been issued.

94655 **-f charmap** Specify the pathname of a file containing a mapping of character symbols and  
 94656 collating element symbols to actual character encodings. The format of the  
 94657 *charmap* is described in XBD [Section 6.4](#) (on page 129). The application shall ensure  
 94658 that this option is specified if symbolic names (other than collating symbols  
 94659 defined in a **collating-symbol** keyword) are used. If the **-f** option is not present, an  
 94660 implementation-defined character mapping shall be used.

94661 **-i inputfile** The pathname of a file containing the source definitions. If this option is not  
 94662 present, source definitions shall be read from standard input. The format of the  
 94663 *inputfile* is described in XBD [Section 7.3](#) (on page 136).



94664        **-u** *code\_set\_name*  
 94665               Specify the name of a codeset used as the target mapping of character symbols and  
 94666               collating element symbols whose encoding values are defined in terms of the  
 94667               ISO/IEC 10646-1:2000 standard position constant values.

## 94668 OPERANDS

94669        The following operand shall be supported:

94670        *name*           Identifies the locale; see XBD [Chapter 7](#) (on page 135) for a description of the use of  
 94671               this name. If the name contains one or more <slash> characters, *name* shall be  
 94672               interpreted as a pathname where the created locale definitions shall be stored. If  
 94673               *name* does not contain any <slash> characters, the interpretation of the name is  
 94674               implementation-defined and the locale shall be public. The ability to create public  
 94675               locales in this way may be restricted to users with appropriate privileges. (As a  
 94676               consequence of specifying one *name*, although several categories can be processed  
 94677               in one execution, only categories belonging to the same locale can be processed.)

## 94678 STDIN

94679        Unless the **-i** option is specified, the standard input shall be a text file containing one or more  
 94680        locale category source definitions, as described in XBD [Section 7.3](#) (on page 136). When lines are  
 94681        continued using the escape character mechanism, there is no limit to the length of the  
 94682        accumulated continued line.

## 94683 INPUT FILES

94684        The character set mapping file specified as the *charmap* option-argument is described in XBD  
 94685        [Section 6.4](#) (on page 129). If a locale category source definition contains a **copy** statement, as  
 94686        defined in XBD [Chapter 7](#) (on page 135), and the **copy** statement names a valid, existing locale,  
 94687        then *localedef* shall behave as if the source definition had contained a valid category source  
 94688        definition for the named locale.

## 94689 ENVIRONMENT VARIABLES

94690        The following environment variables shall affect the execution of *localedef*:

94691        **LANG**           Provide a default value for the internationalization variables that are unset or null.  
 94692               (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 94693               variables used to determine the values of locale categories.)

94694        **LC\_ALL**        If set to a non-empty string value, override the values of all the other  
 94695               internationalization variables.

94696        **LC\_COLLATE**  
 94697               (This variable has no affect on *localedef*; the POSIX locale is used for this category.)

94698        **LC\_CTYPE**       Determine the locale for the interpretation of sequences of bytes of text data as  
 94699               characters (for example, single-byte as opposed to multi-byte characters in  
 94700               arguments and input files). This variable has no affect on the processing of *localedef*  
 94701               input data; the POSIX locale is used for this purpose, regardless of the value of this  
 94702               variable.

94703        **LC\_MESSAGES**  
 94704               Determine the locale that should be used to affect the format and contents of  
 94705               diagnostic messages written to standard error.

94706        **XS1 NLSPATH**   Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

94707 **ASYNCHRONOUS EVENTS**

94708 Default.

94709 **STDOUT**

94710 The utility shall report all categories successfully processed, in an unspecified format.

94711 **STDERR**

94712 The standard error shall be used only for diagnostic messages.

94713 **OUTPUT FILES**94714 The format of the created output is unspecified. If the *name* operand does not contain a <slash>,  
94715 the existence of an output file for the locale is unspecified.94716 **EXTENDED DESCRIPTION**94717 When the **-u** option is used, the *code\_set\_name* option-argument shall be interpreted as an  
94718 implementation-defined name of a codeset to which the ISO/IEC 10646-1:2000 standard  
94719 position constant values shall be converted via an implementation-defined method. Both the  
94720 ISO/IEC 10646-1:2000 standard position constant values and other formats (decimal,  
94721 hexadecimal, or octal) shall be valid as encoding values within the *charmap* file. The codeset  
94722 represented by the implementation-defined name can be any codeset that is supported by the  
94723 implementation.94724 When conflicts occur between the *charmap* specification of <*code\_set\_name*>, <*mb\_cur\_max*>, or  
94725 <*mb\_cur\_min*> and the implementation-defined interpretation of these respective items for the  
94726 codeset represented by the **-u** option-argument *code\_set\_name*, the result is unspecified.94727 When conflicts occur between the *charmap* encoding values specified for symbolic names of  
94728 characters of the portable character set and the implementation-defined assignment of character  
94729 encoding values, the result is unspecified.94730 If a non-printable character in the *charmap* has a width specified that is not **-1**, the result will be  
94731 undefined.94732 **EXIT STATUS**

94733 The following exit values shall be returned:

- 94734 0 No errors occurred and the locales were successfully created.
- 94735 1 Warnings occurred and the locales were successfully created.
- 94736 2 The locale specification exceeded implementation limits or the coded character set or sets  
94737 used were not supported by the implementation, and no locale was created.
- 94738 3 The capability to create new locales is not supported by the implementation.
- 94739 >3 Warnings or errors occurred and no output was created.

94740 **CONSEQUENCES OF ERRORS**

94741 If an error is detected, no permanent output shall be created.

94742 If warnings occur, permanent output shall be created if the **-c** option was specified. The  
94743 following conditions shall cause warning messages to be issued:

- 94744 • If a symbolic name not found in the *charmap* file is used for the descriptions of the  
94745 *LC\_CTYPE* or *LC\_COLLATE* categories (for other categories, this shall be an error  
94746 condition).
- 94747 • If the number of operands to the **order** keyword exceeds the {*COLL\_WEIGHTS\_MAX*}  
94748 limit.

- If optional keywords not supported by the implementation are present in the source.

Other implementation-defined conditions may also cause warnings.

## APPLICATION USAGE

The *charmap* definition is optional, and is contained outside the locale definition. This allows both completely self-defined source files, and generic sources (applicable to more than one codeset). To aid portability, all *charmap* definitions must use the same symbolic names for the portable character set. As explained in XBD [Section 6.4](#) (on page 129), it is implementation-defined whether or not users or applications can provide additional character set description files. Therefore, the `-f` option might be operable only when an implementation-defined *charmap* is named.

## EXAMPLES

None.

## RATIONALE

The output produced by the *localedef* utility is implementation-defined. The *name* operand is used to identify the specific locale. (As a consequence, although several categories can be processed in one execution, only categories belonging to the same locale can be processed.)

## FUTURE DIRECTIONS

None.

## SEE ALSO

[locale](#)

XBD [Section 6.4](#) (on page 129), [Chapter 7](#) (on page 135), [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 215)

## CHANGE HISTORY

First released in Issue 4.

### Issue 6

The `-u` option is added, as specified in the IEEE P1003.2b draft standard.

The normative text is reworded to avoid use of the term “must” for application requirements.

IEEE Std 1003.1-2001/Cor 2-2004, item XCU/TC2/D6/15 is applied, rewording text in the OPERANDS section describing the ability to create public locales.

IEEE Std 1003.1-2001/Cor 2-2004, item XCU/TC2/D6/16 is applied, making the text consistent with the descriptions of **WIDTH** and **WIDTH\_DEFAULT** in the Base Definitions volume of POSIX.1-2008.

### Issue 7

SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

94783 **NAME**

94784       logger — log messages

94785 **SYNOPSIS**94786       logger *string*...94787 **DESCRIPTION**

94788       The *logger* utility saves a message, in an unspecified manner and format, containing the *string*  
 94789       operands provided by the user. The messages are expected to be evaluated later by personnel  
 94790       performing system administration tasks.

94791       It is implementation-defined whether messages written in locales other than the POSIX locale  
 94792       are effective.

94793 **OPTIONS**

94794       None.

94795 **OPERANDS**

94796       The following operand shall be supported:

94797       *string*       One of the string arguments whose contents are concatenated together, in the order  
 94798       specified, separated by single <space> characters.

94799 **STDIN**

94800       Not used.

94801 **INPUT FILES**

94802       None.

94803 **ENVIRONMENT VARIABLES**94804       The following environment variables shall affect the execution of *logger*:

94805       *LANG*       Provide a default value for the internationalization variables that are unset or null.  
 94806       (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 94807       variables used to determine the values of locale categories.)

94808       *LC\_ALL*      If set to a non-empty string value, override the values of all the other  
 94809       internationalization variables.

94810       *LC\_CTYPE*   Determine the locale for the interpretation of sequences of bytes of text data as  
 94811       characters (for example, single-byte as opposed to multi-byte characters in  
 94812       arguments).

94813       *LC\_MESSAGES*

94814       Determine the locale that should be used to affect the format and contents of  
 94815       diagnostic messages written to standard error. (This means diagnostics from *logger*  
 94816       to the user or application, not diagnostic messages that the user is sending to the  
 94817       system administrator.)

94818       XSI       *NLSPATH*   Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

94819 **ASYNCHRONOUS EVENTS**

94820       Default.

94821 **STDOUT**

94822       Not used.

**94823 STDERR**

94824 The standard error shall be used only for diagnostic messages.

**94825 OUTPUT FILES**

94826 Unspecified.

**94827 EXTENDED DESCRIPTION**

94828 None.

**94829 EXIT STATUS**

94830 The following exit values shall be returned:

94831 0 Successful completion.

94832 >0 An error occurred.

**94833 CONSEQUENCES OF ERRORS**

94834 Default.

**94835 APPLICATION USAGE**

94836 This utility allows logging of information for later use by a system administrator or programmer  
94837 in determining why non-interactive utilities have failed. The locations of the saved messages,  
94838 their format, and retention period are all unspecified. There is no method for a conforming  
94839 application to read messages, once written.

**94840 EXAMPLES**

94841 A batch application, running non-interactively, tries to read a configuration file and fails; it may  
94842 attempt to notify the system administrator with:

94843 `logger myname: unable to read file foo. [timestamp]`

**94844 RATIONALE**

94845 The standard developers believed strongly that some method of alerting administrators to errors  
94846 was necessary. The obvious example is a batch utility, running non-interactively, that is unable to  
94847 read its configuration files or that is unable to create or write its results file. However, the  
94848 standard developers did not wish to define the format or delivery mechanisms as they have  
94849 historically been (and will probably continue to be) very system-specific, as well as involving  
94850 functionality clearly outside the scope of this volume of POSIX.1-2008.

94851 The text with *LC\_MESSAGES* about diagnostic messages means diagnostics from *logger* to the  
94852 user or application, not diagnostic messages that the user is sending to the system administrator.

94853 Multiple *string* arguments are allowed, similar to *echo*, for ease-of-use.

94854 Like the utilities *mailx* and *lp*, *logger* is admittedly difficult to test. This was not deemed sufficient  
94855 justification to exclude these utilities from this volume of POSIX.1-2008. It is also arguable that  
94856 they are, in fact, testable, but that the tests themselves are not portable.

**94857 FUTURE DIRECTIONS**

94858 None.

**94859 SEE ALSO**

94860 *lp*, *mailx*, *write*

94861 XBD Chapter 8 (on page 173)

**94862 CHANGE HISTORY**

94863 First released in Issue 4.

94864

94865

Issue 7

SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

94866 **NAME**

94867       logname — return the user's login name

94868 **SYNOPSIS**

94869       logname

94870 **DESCRIPTION**

94871       The *logname* utility shall write the user's login name to standard output. The login name shall be  
 94872       the string that would be returned by the *getlogin()* function defined in the System Interfaces  
 94873       volume of POSIX.1-2008. Under the conditions where the *getlogin()* function would fail, the  
 94874       *logname* utility shall write a diagnostic message to standard error and exit with a non-zero exit  
 94875       status.

94876 **OPTIONS**

94877       None.

94878 **OPERANDS**

94879       None.

94880 **STDIN**

94881       Not used.

94882 **INPUT FILES**

94883       None.

94884 **ENVIRONMENT VARIABLES**94885       The following environment variables shall affect the execution of *logname*:

94886       *LANG*       Provide a default value for the internationalization variables that are unset or null.  
 94887       (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 94888       variables used to determine the values of locale categories.)

94889       *LC\_ALL*     If set to a non-empty string value, override the values of all the other  
 94890       internationalization variables.

94891       *LC\_CTYPE*   Determine the locale for the interpretation of sequences of bytes of text data as  
 94892       characters (for example, single-byte as opposed to multi-byte characters in  
 94893       arguments).

94894       *LC\_MESSAGES*  
 94895       Determine the locale that should be used to affect the format and contents of  
 94896       diagnostic messages written to standard error.

94897   XSI       *NLSPATH*   Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

94898 **ASYNCHRONOUS EVENTS**

94899       Default.

94900 **STDOUT**94901       The *logname* utility output shall be a single line consisting of the user's login name:

94902       "%s\n", &lt;login name&gt;

94903 **STDERR**

94904       The standard error shall be used only for diagnostic messages.

94905 **OUTPUT FILES**

94906       None.

94907 **EXTENDED DESCRIPTION**

94908 None.

94909 **EXIT STATUS**

94910 The following exit values shall be returned:

94911 0 Successful completion.

94912 &gt;0 An error occurred.

94913 **CONSEQUENCES OF ERRORS**

94914 Default.

94915 **APPLICATION USAGE**94916 The *logname* utility explicitly ignores the *LOGNAME* environment variable because environment  
94917 changes could produce erroneous results.94918 **EXAMPLES**

94919 None.

94920 **RATIONALE**94921 The **passwd** file is not listed as required because the implementation may have other means of  
94922 mapping login names.94923 **FUTURE DIRECTIONS**

94924 None.

94925 **SEE ALSO**94926 *id*, *who*94927 XBD [Chapter 8](#) (on page 173)94928 XSH [getlogin\(\)](#)94929 **CHANGE HISTORY**

94930 First released in Issue 2.



94931 **NAME**94932 `lp` — send files to a printer94933 **SYNOPSIS**94934 `lp [-c] [-d dest] [-n copies] [-msw] [-o option]... [-t title] [file...]`94935 **DESCRIPTION**

94936 The *lp* utility shall copy the input files to an output destination in an unspecified manner. The  
 94937 default output destination should be to a hardcopy device, such as a printer or microfilm  
 94938 recorder, that produces non-volatile, human-readable documents. If such a device is not  
 94939 available to the application, or if the system provides no such device, the *lp* utility shall exit with  
 94940 a non-zero exit status.

94941 The actual writing to the output device may occur some time after the *lp* utility successfully  
 94942 exits. During the portion of the writing that corresponds to each input file, the implementation  
 94943 shall guarantee exclusive access to the device.

94944 The *lp* utility shall associate a unique *request ID* with each request.

94945 Normally, a banner page is produced to separate and identify each print job. This page may be  
 94946 suppressed by implementation-defined conditions, such as an operator command or one of the  
 94947 `-o option` values.

94948 **OPTIONS**94949 The *lp* utility shall conform to XBD [Section 12.2](#) (on page 215).

94950 The following options shall be supported:

94951 `-c` Exit only after further access to any of the input files is no longer required. The  
 94952 application can then safely delete or modify the files without affecting the output  
 94953 operation. Normally, files are not copied, but are linked whenever possible. If the  
 94954 `-c` option is not given, then the user should be careful not to remove any of the  
 94955 files before the request has been printed in its entirety. It should also be noted that  
 94956 in the absence of the `-c` option, any changes made to the named files after the  
 94957 request is made but before it is printed may be reflected in the printed output. On  
 94958 some implementations, `-c` may be on by default.

94959 `-d dest` Specify a string that names the destination (*dest*). If *dest* is a printer, the request  
 94960 shall be printed only on that specific printer. If *dest* is a class of printers, the request  
 94961 shall be printed on the first available printer that is a member of the class. Under  
 94962 certain conditions (printer unavailability, file space limitation, and so on), requests  
 94963 for specific destinations need not be accepted. Destination names vary between  
 94964 systems.

94965 If `-d` is not specified, and neither the *LPDEST* nor *PRINTER* environment variable  
 94966 is set, an unspecified destination is used. The `-d dest` option shall take precedence  
 94967 over *LPDEST*, which in turn shall take precedence over *PRINTER*. Results are  
 94968 undefined when *dest* contains a value that is not a valid destination name.

94969 `-m` Send mail (see [mailx](#)) after the files have been printed. By default, no mail is sent  
 94970 upon normal completion of the print request.

94971 `-n copies` Write *copies* number of copies of the files, where *copies* is a positive decimal integer.  
 94972 The methods for producing multiple copies and for arranging the multiple copies  
 94973 when multiple *file* operands are used are unspecified, except that each file shall be  
 94974 output as an integral whole, not interleaved with portions of other files.

- 94975        **-o option**        Specify printer-dependent or class-dependent *options*. Several such *options* may be  
 94976        collected by specifying the **-o** option more than once.
- 94977        **-s**                Suppress messages from *lp*.
- 94978        **-t title**            Write *title* on the banner page of the output.
- 94979        **-w**                Write a message on the user's terminal after the files have been printed. If the user  
 94980        is not logged in, then mail shall be sent instead.

## 94981 OPERANDS

94982        The following operand shall be supported:

- 94983        *file*            A pathname of a file to be output. If no *file* operands are specified, or if a *file*  
 94984        operand is '-', the standard input shall be used. If a *file* operand is used, but the  
 94985        **-c** option is not specified, the process performing the writing to the output device  
 94986        may have user and group permissions that differ from that of the process invoking  
 94987        *lp*.

## 94988 STDIN

94989        The standard input shall be used only if no *file* operands are specified, or if a *file* operand is '-'.  
 94990        See the INPUT FILES section.

## 94991 INPUT FILES

94992        The input files shall be text files.

## 94993 ENVIRONMENT VARIABLES

94994        The following environment variables shall affect the execution of *lp*:

- 94995        *LANG*            Provide a default value for the internationalization variables that are unset or null.  
 94996        (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 94997        variables used to determine the values of locale categories.)
- 94998        *LC\_ALL*            If set to a non-empty string value, override the values of all the other  
 94999        internationalization variables.
- 95000        *LC\_CTYPE*        Determine the locale for the interpretation of sequences of bytes of text data as  
 95001        characters (for example, single-byte as opposed to multi-byte characters in  
 95002        arguments and input files).
- 95003        *LC\_MESSAGES*     Determine the locale that should be used to affect the format and contents of  
 95004        diagnostic messages written to standard error and informative messages written to  
 95005        standard output.  
 95006
- 95007        *LC\_TIME*        Determine the format and contents of date and time strings displayed in the *lp*  
 95008        banner page, if any.
- 95009        *LPDEST*            Determine the destination. If the *LPDEST* environment variable is not set, the  
 95010        *PRINTER* environment variable shall be used. The **-d dest** option takes precedence  
 95011        over *LPDEST*. Results are undefined when **-d** is not specified and *LPDEST*  
 95012        contains a value that is not a valid destination name.
- 95013        *NLSPATH*        Determine the location of message catalogs for the processing of *LC\_MESSAGES*.
- 95014        *PRINTER*        Determine the output device or destination. If the *LPDEST* and *PRINTER*  
 95015        environment variables are not set, an unspecified output device is used. The **-d**  
 95016        *dest* option and the *LPDEST* environment variable shall take precedence over  
 95017        *PRINTER*. Results are undefined when **-d** is not specified, *LPDEST* is unset, and  
 95018        *PRINTER* contains a value that is not a valid device or destination name.

95019            *TZ*            Determine the timezone used to calculate date and time strings displayed in the *lp*  
 95020 banner page, if any. If *TZ* is unset or null, an unspecified default timezone shall be  
 95021 used.

## 95022 **ASYNCHRONOUS EVENTS**

95023            Default.

## 95024 **STDOUT**

95025            The *lp* utility shall write a *request ID* to the standard output, unless *-s* is specified. The format of  
 95026 the message is unspecified. The request ID can be used on systems supporting the historical  
 95027 *cancel* and *lpstat* utilities.

## 95028 **STDERR**

95029            The standard error shall be used only for diagnostic messages.

## 95030 **OUTPUT FILES**

95031            None.

## 95032 **EXTENDED DESCRIPTION**

95033            None.

## 95034 **EXIT STATUS**

95035            The following exit values shall be returned:

95036            0    All input files were processed successfully.

95037            >0   No output device was available, or an error occurred.

## 95038 **CONSEQUENCES OF ERRORS**

95039            Default.

## 95040 **APPLICATION USAGE**

95041            The *pr* and *fold* utilities can be used to achieve reasonable formatting for the implementation's  
 95042 default page size.

95043            A conforming application can use one of the *file* operands only with the *-c* option or if the file is  
 95044 publicly readable and guaranteed to be available at the time of printing. This is because  
 95045 POSIX.1-2008 gives the implementation the freedom to queue up the request for printing at  
 95046 some later time by a different process that might not be able to access the file.

## 95047 **EXAMPLES**

95048            1. To print file *file*:

95049            `lp -c file`

95050            2. To print multiple files with headers:

95051            `pr file1 file2 | lp`

## 95052 **RATIONALE**

95053            The *lp* utility was designed to be a basic version of a utility that is already available in many  
 95054 historical implementations. The standard developers considered that it should be implementable  
 95055 simply as:

95056            `cat "$@" > /dev/lp`

95057            after appropriate processing of options, if that is how the implementation chose to do it and if  
 95058 exclusive access could be granted (so that two users did not write to the device simultaneously).  
 95059 Although in the future the standard developers may add other options to this utility, it should  
 95060 always be able to execute with no options or operands and send the standard input to an

unspecified output device.

This volume of POSIX.1-2008 makes no representations concerning the format of the printed output, except that it must be “human-readable” and “non-volatile”. Thus, writing by default to a disk or tape drive or a display terminal would not qualify. (Such destinations are not prohibited when `-d dest`, `LPDEST`, or `PRINTER` are used, however.)

This volume of POSIX.1-2008 is worded such that a “print job” consisting of multiple input files, possibly in multiple copies, is guaranteed to print so that any one file is not intermixed with another, but there is no statement that all the files or copies have to print out together.

The `-c` option may imply a spooling operation, but this is not required. The utility can be implemented to wait until the printer is ready and then wait until it is finished. Because of that, there is no attempt to define a queuing mechanism (priorities, classes of output, and so on).

On some historical systems, the request ID reported on the `STDOUT` can be used to later cancel or find the status of a request using utilities not defined in this volume of POSIX.1-2008.

Although the historical System V `lp` and BSD `lpr` utilities have provided similar functionality, they used different names for the environment variable specifying the destination printer. Since the name of the utility here is `lp`, `LPDEST` (used by the System V `lp` utility) was given precedence over `PRINTER` (used by the BSD `lpr` utility). Since environments of users frequently contain one or the other environment variable, the `lp` utility is required to recognize both. If this was not done, many applications would send output to unexpected output devices when users moved from system to system.

Some have commented that `lp` has far too little functionality to make it worthwhile. Requests have proposed additional options or operands or both that added functionality. The requests included:

- Wording *requiring* the output to be “hardcopy”
- A requirement for multiple printers
- Options for supporting various page-description languages

Given that a compliant system is not required to even have a printer, placing further restrictions upon the behavior of the printer is not useful. Since hardcopy format is so application-dependent, it is difficult, if not impossible, to select a reasonable subset of functionality that should be required on all compliant systems.

The term *unspecified* is used in this section in lieu of *implementation-defined* as most known implementations would not be able to make definitive statements in their conformance documents; the existence and usage of printers is very dependent on how the system administrator configures each individual system.

Since the default destination, device type, queuing mechanisms, and acceptable forms of input are all unspecified, usage guidelines for what a conforming application can do are as follows:

- Use the command in a pipeline, or with `-c`, so that there are no permission problems and the files can be safely deleted or modified.
- Limit output to text files of reasonable line lengths and printable characters and include no device-specific formatting information, such as a page description language. The meaning of “reasonable” in this context can only be answered as a quality-of-implementation issue, but it should be apparent from historical usage patterns in the industry and the locale. The `pr` and `fold` utilities can be used to achieve reasonable formatting for the default page size of the implementation.

Alternatively, the application can arrange its installation in such a way that it requires the system administrator or operator to provide the appropriate information on *lp* options and environment variable values.

At a minimum, having this utility in this volume of POSIX.1-2008 tells the industry that conforming applications require a means to print output and provides at least a command name and *LPDEST* routing mechanism that can be used for discussions between vendors, application developers, and users. The use of “should” in the DESCRIPTION of *lp* clearly shows the intent of the standard developers, even if they cannot mandate that all systems (such as laptops) have printers.

This volume of POSIX.1-2008 does not specify what the ownership of the process performing the writing to the output device may be. If *-c* is not used, it is unspecified whether the process performing the writing to the output device has permission to read *file* if there are any restrictions in place on who may read *file* until after it is printed. Also, if *-c* is not used, the results of deleting *file* before it is printed are unspecified.

#### FUTURE DIRECTIONS

None.

#### SEE ALSO

*mailx*

XBD [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 215)

#### CHANGE HISTORY

First released in Issue 2.

#### Issue 6

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- In the DESCRIPTION, the requirement to associate a unique request ID, and the normal generation of a banner page is added.
- In the OPTIONS section:
  - The *-d dest* description is expanded, but references to *lpstat* are removed.
  - The *-m*, *-o*, *-s*, *-t*, and *-w* options are added.
- In the ENVIRONMENT VARIABLES section, *LC\_TIME* may now affect the execution.
- The STDOUT section is added.

The normative text is reworded to avoid use of the term “must” for application requirements.

The *TZ* entry is added to the ENVIRONMENT VARIABLES section.

#### Issue 7

SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

**NAME**

ls — list directory contents

**SYNOPSIS**

```
ls [-ikqrs] [-g lno] [-A|-a] [-C|-m|-x|-l] \
 [-F|-p] [-H|-L] [-R|-d] [-S|-f|-t] [-c|-u] [file...]
```

**DESCRIPTION**

For each operand that names a file of a type other than directory or symbolic link to a directory, *ls* shall write the name of the file as well as any requested, associated information. For each operand that names a file of type directory, *ls* shall write the names of files contained within the directory as well as any requested, associated information. Filenames beginning with a <period> ('.') and any associated information shall not be written out unless explicitly referenced, the *-A* or *-a* option is supplied, or an implementation-defined condition causes them to be written. If one or more of the *-d*, *-F*, or *-l* options are specified, and neither the *-H* nor the *-L* option is specified, for each operand that names a file of type symbolic link to a directory, *ls* shall write the name of the file as well as any requested, associated information. If none of the *-d*, *-F*, or *-l* options are specified, or the *-H* or *-L* options are specified, for each operand that names a file of type symbolic link to a directory, *ls* shall write the names of files contained within the directory as well as any requested, associated information. In each case where the names of files contained within a directory are written, if the directory contains any symbolic links then *ls* shall evaluate the file information and file type to be those of the symbolic link itself, unless the *-L* option is specified.

If no operands are specified, *ls* shall behave as if a single operand of dot ('.') had been specified. If more than one operand is specified, *ls* shall write non-directory operands first; it shall sort directory and non-directory operands separately according to the collating sequence in the current locale.

The *ls* utility shall detect infinite loops; that is, entering a previously visited directory that is an ancestor of the last file encountered. When it detects an infinite loop, *ls* shall write a diagnostic message to standard error and shall either recover its position in the hierarchy or terminate.

**OPTIONS**

The *ls* utility shall conform to XBD [Section 12.2](#) (on page 215).

The following options shall be supported:

- A** Write out all directory entries, including those whose names begin with a <period> ('.') but excluding the entries dot and dot-dot (if they exist).
- C** Write multi-text-column output with entries sorted down the columns, according to the collating sequence. The number of text columns and the column separator characters are unspecified, but should be adapted to the nature of the output device. This option disables long format output.
- F** Do not follow symbolic links named as operands unless the *-H* or *-L* options are specified. Write a <slash> ('/') immediately after each pathname that is a directory, an <asterisk> ('\*') after each that is executable, a <vertical-line> ('|') after each that is a FIFO, and an at-sign('@') after each that is a symbolic link. For other file types, other symbols may be written.
- H** Evaluate the file information and file type for symbolic links specified on the command line to be those of the file referenced by the link, and not the link itself; however, *ls* shall write the name of the link itself and not the file referenced by the link.

|       |     |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |   |
|-------|-----|-----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---|
| 95186 |     | <b>-L</b> | Evaluate the file information and file type for all symbolic links (whether named on the command line or encountered in a file hierarchy) to be those of the file referenced by the link, and not the link itself; however, <i>ls</i> shall write the name of the link itself and not the file referenced by the link. When <b>-L</b> is used with <b>-l</b> , write the contents of symbolic links in the long format (see the STDOUT section).                                                               |   |
| 95187 |     |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |   |
| 95188 |     |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |   |
| 95189 |     |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |   |
| 95190 |     |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |   |
| 95191 |     | <b>-R</b> | Recursively list subdirectories encountered. When a symbolic link to a directory is encountered, the directory shall not be recursively listed unless the <b>-L</b> option is specified. The use of <b>-R</b> with <b>-d</b> or <b>-f</b> produces unspecified results.                                                                                                                                                                                                                                        | + |
| 95192 |     |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |   |
| 95193 |     |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |   |
| 95194 |     | <b>-S</b> | Sort with the primary key being file size (in decreasing order) and the secondary key being filename in the collating sequence (in increasing order).                                                                                                                                                                                                                                                                                                                                                          |   |
| 95195 |     |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |   |
| 95196 |     | <b>-a</b> | Write out all directory entries, including those whose names begin with a <period> ('.').                                                                                                                                                                                                                                                                                                                                                                                                                      |   |
| 95197 |     |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |   |
| 95198 |     | <b>-c</b> | Use time of last modification of the file status information (see XBD <sys/stat.h>) instead of last modification of the file itself for sorting ( <b>-t</b> ) or writing ( <b>-l</b> ).                                                                                                                                                                                                                                                                                                                        |   |
| 95199 |     |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |   |
| 95200 |     | <b>-d</b> | Do not follow symbolic links named as operands unless the <b>-H</b> or <b>-L</b> options are specified. Do not treat directories differently than other types of files. The use of <b>-d</b> with <b>-R</b> or <b>-f</b> produces unspecified results.                                                                                                                                                                                                                                                         | + |
| 95201 |     |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |   |
| 95202 |     |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |   |
| 95203 |     | <b>-f</b> | List the entries in directory operands in the order they appear in the directory. The behavior for non-directory operands is unspecified. This option shall turn on <b>-a</b> . When <b>-f</b> is specified, any occurrences of the <b>-r</b> , <b>-S</b> , and <b>-t</b> options shall be ignored and any occurrences of the <b>-A</b> , <b>-g</b> , <b>-l</b> , <b>-n</b> , <b>-o</b> , and <b>-s</b> options may be ignored. The use of <b>-f</b> with <b>-R</b> or <b>-d</b> produces unspecified results. |   |
| 95204 |     |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |   |
| 95205 |     |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |   |
| 95206 | XSI |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |   |
| 95207 |     |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |   |
| 95208 | XSI | <b>-g</b> | Turn on the <b>-l</b> (ell) option, but disable writing the file's owner name or number. Disable the <b>-C</b> , <b>-m</b> , and <b>-x</b> options.                                                                                                                                                                                                                                                                                                                                                            |   |
| 95209 |     |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |   |
| 95210 |     | <b>-i</b> | For each file, write the file's file serial number (see <i>stat()</i> in the System Interfaces volume of POSIX.1-2008).                                                                                                                                                                                                                                                                                                                                                                                        |   |
| 95211 |     |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |   |
| 95212 |     | <b>-k</b> | Set the block size for the <b>-s</b> option and the per-directory block count written for the <b>-l</b> , <b>-n</b> , <b>-s</b> , <b>-g</b> , and <b>-o</b> options (see the STDOUT section) to 1 024 bytes.                                                                                                                                                                                                                                                                                                   |   |
| 95213 | XSI |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |   |
| 95214 |     | <b>-l</b> | (The letter ell.) Do not follow symbolic links named as operands unless the <b>-H</b> or <b>-L</b> options are specified. Write out in long format (see the STDOUT section). Disable the <b>-C</b> , <b>-m</b> , and <b>-x</b> options.                                                                                                                                                                                                                                                                        |   |
| 95215 |     |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |   |
| 95216 |     |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |   |
| 95217 |     | <b>-m</b> | Stream output format; list pathnames across the page, separated by a <comma> character followed by a <space> character. Use a <newline> character as the list terminator and after the separator sequence when there is not room on a line for the next list entry. This option disables long format output.                                                                                                                                                                                                   |   |
| 95218 |     |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |   |
| 95219 |     |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |   |
| 95220 |     |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |   |
| 95221 |     | <b>-n</b> | Turn on the <b>-l</b> (ell) option, but when writing the file's owner or group, write the file's numeric UID or GID rather than the user or group name, respectively. Disable the <b>-C</b> , <b>-m</b> , and <b>-x</b> options.                                                                                                                                                                                                                                                                               |   |
| 95222 |     |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |   |
| 95223 |     |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |   |
| 95224 | XSI | <b>-o</b> | Turn on the <b>-l</b> (ell) option, but disable writing the file's group name or number. Disable the <b>-C</b> , <b>-m</b> , and <b>-x</b> options.                                                                                                                                                                                                                                                                                                                                                            |   |
| 95225 |     |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |   |
| 95226 |     | <b>-p</b> | Write a <slash> ('/') after each filename if that file is a directory.                                                                                                                                                                                                                                                                                                                                                                                                                                         |   |
| 95227 |     | <b>-q</b> | Force each instance of non-printable filename characters and <tab> characters to be written as the <question-mark> ('?') character. Implementations may provide this option by default if the output is to a terminal device.                                                                                                                                                                                                                                                                                  |   |
| 95228 |     |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |   |
| 95229 |     |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |   |

|       |           |                                                                                                                                                                                                                                                                                                                                                                                                           |  |
|-------|-----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--|
| 95230 | <b>-r</b> | Reverse the order of the sort to get reverse collating sequence oldest first, or smallest file size first depending on the other options given.                                                                                                                                                                                                                                                           |  |
| 95231 |           |                                                                                                                                                                                                                                                                                                                                                                                                           |  |
| 95232 | <b>-s</b> | Indicate the total number of file system blocks consumed by each file displayed. If the <b>-k</b> option is also specified, the block size shall be 1 024 bytes; otherwise, the block size is implementation-defined.                                                                                                                                                                                     |  |
| 95233 |           |                                                                                                                                                                                                                                                                                                                                                                                                           |  |
| 95234 |           |                                                                                                                                                                                                                                                                                                                                                                                                           |  |
| 95235 | <b>-t</b> | Sort with the primary key being time modified (most recently modified first) and the secondary key being filename in the collating sequence. For a symbolic link, the time used as the sort key is that of the symbolic link itself, unless <i>ls</i> is evaluating its file information to be that of the file referenced by the link (see the <b>-H</b> and <b>-L</b> options).                         |  |
| 95236 |           |                                                                                                                                                                                                                                                                                                                                                                                                           |  |
| 95237 |           |                                                                                                                                                                                                                                                                                                                                                                                                           |  |
| 95238 |           |                                                                                                                                                                                                                                                                                                                                                                                                           |  |
| 95239 |           |                                                                                                                                                                                                                                                                                                                                                                                                           |  |
| 95240 | <b>-u</b> | Use time of last access (see XBD <a href="#">&lt;sys/stat.h&gt;</a> ) instead of last modification of the file for sorting ( <b>-t</b> ) or writing ( <b>-l</b> ).                                                                                                                                                                                                                                        |  |
| 95241 |           |                                                                                                                                                                                                                                                                                                                                                                                                           |  |
| 95242 | <b>-x</b> | The same as <b>-C</b> , except that the multi-text-column output is produced with entries sorted across, rather than down, the columns. This option disables long format output.                                                                                                                                                                                                                          |  |
| 95243 |           |                                                                                                                                                                                                                                                                                                                                                                                                           |  |
| 95244 |           |                                                                                                                                                                                                                                                                                                                                                                                                           |  |
| 95245 | <b>-1</b> | (The numeric digit one.) Force output to be one entry per line. This option does not disable long format output. (Long format output is enabled by <b>-g</b> , <b>-l</b> (ell), <b>-n</b> , and <b>-o</b> ; and disabled by <b>-C</b> , <b>-m</b> , and <b>-x</b> .)                                                                                                                                      |  |
| 95246 | XSI       |                                                                                                                                                                                                                                                                                                                                                                                                           |  |
| 95247 | XSI       |                                                                                                                                                                                                                                                                                                                                                                                                           |  |
| 95248 | XSI       | If an option that enables long format output ( <b>-g</b> , <b>-l</b> (ell), <b>-n</b> , and <b>-o</b> ) is given with an option that disables long format output ( <b>-C</b> , <b>-m</b> , and <b>-x</b> ), this shall not be considered an error. The last of these options specified shall determine whether long format output is written.                                                             |  |
| 95249 |           |                                                                                                                                                                                                                                                                                                                                                                                                           |  |
| 95250 |           |                                                                                                                                                                                                                                                                                                                                                                                                           |  |
| 95251 |           | If <b>-R</b> , <b>-d</b> , or <b>-f</b> are specified, the results of specifying these mutually-exclusive options are specified by the descriptions of these options above. If more than one of any of the other options shown in the SYNOPSIS section in mutually-exclusive sets are given, this shall not be considered an error; the last option specified in each set shall determine the output.     |  |
| 95252 |           |                                                                                                                                                                                                                                                                                                                                                                                                           |  |
| 95253 |           |                                                                                                                                                                                                                                                                                                                                                                                                           |  |
| 95254 |           |                                                                                                                                                                                                                                                                                                                                                                                                           |  |
| 95255 |           | Note that if <b>-t</b> is specified, <b>-c</b> and <b>-u</b> are not only mutually-exclusive with each other, they are also mutually-exclusive with <b>-S</b> when determining sort order. But even if <b>-S</b> is specified after all occurrences of <b>-c</b> , <b>-t</b> , and <b>-u</b> , the last use of <b>-c</b> or <b>-u</b> determines the timestamp printed when producing long format output. |  |
| 95256 |           |                                                                                                                                                                                                                                                                                                                                                                                                           |  |
| 95257 |           |                                                                                                                                                                                                                                                                                                                                                                                                           |  |
| 95258 |           |                                                                                                                                                                                                                                                                                                                                                                                                           |  |

## OPERANDS

The following operand shall be supported:

|       |             |                                                                                                                                 |
|-------|-------------|---------------------------------------------------------------------------------------------------------------------------------|
| 95261 | <i>file</i> | A pathname of a file to be written. If the file specified is not found, a diagnostic message shall be output on standard error. |
| 95262 |             |                                                                                                                                 |

## STDIN

Not used.

## INPUT FILES

None.



**ENVIRONMENT VARIABLES**

The following environment variables shall affect the execution of *ls*:

**COLUMNS** Determine the user's preferred column position width for writing multiple text-column output. If this variable contains a string representing a decimal integer, the *ls* utility shall calculate how many pathname text columns to write (see **-C**) based on the width provided. If **COLUMNS** is not set or invalid, an implementation-defined number of column positions shall be assumed, based on the implementation's knowledge of the output device. The column width chosen to write the names of files in any given directory shall be constant. Filenames shall not be truncated to fit into the multiple text-column output.

**LANG** Provide a default value for the internationalization variables that are unset or null. (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization variables used to determine the values of locale categories.)

**LC\_ALL** If set to a non-empty string value, override the values of all the other internationalization variables.

**LC\_COLLATE** Determine the locale for character collation information in determining the pathname collation sequence.

**LC\_CTYPE** Determine the locale for the interpretation of sequences of bytes of text data as characters (for example, single-byte as opposed to multi-byte characters in arguments) and which characters are defined as printable (character class **print**).

**LC\_MESSAGES** Determine the locale that should be used to affect the format and contents of diagnostic messages written to standard error.

**LC\_TIME** Determine the format and contents for date and time strings written by *ls*.

**NLSPATH** Determine the location of message catalogs for the processing of **LC\_MESSAGES**.

**TZ** Determine the timezone for date and time strings written by *ls*. If **TZ** is unset or null, an unspecified default timezone shall be used.

**ASYNCHRONOUS EVENTS**

Default.

**STDOUT**

The default format shall be to list one entry per line to standard output; the exceptions are to terminals or when one of the **-C**, **-m**, or **-x** options is specified. If the output is to a terminal, the format is implementation-defined.

When **-m** is specified, the format used for the last element of the list shall be:

"%s\n", <filename>

The format used for each other element of the list shall be:

"%s,%s", <filename>, <separator>

where, if there is not room for the next element of the list to fit within the current line length, <separator> is a string containing an optional <space> character and a mandatory <newline> character; otherwise it is a single <space> character.

If the **-i** option is specified, the file's file serial number (see XBD [<sys/stat.h>](#)) shall be written in the following format before any other output for the corresponding entry:

95310        %u ", <file serial number>

95311        If the **-l** option is specified, the following information shall be written for files other than  
95312        character special and block special files:

95313        "%s %u %s %s %u %s %s\n", <file mode>, <number of links>,  
95314               <owner name>, <group name>, <size>, <date and time>,  
95315               <pathname>

95316        If the **-l** option is specified, the following information shall be written for character special and  
95317        block special files:

95318        "%s %u %s %s %s %s %s\n", <file mode>, <number of links>,  
95319               <owner name>, <group name>, <device info>, <date and time>,  
95320               <pathname>

95321        In both cases if the file is a symbolic link and the **-L** option is also specified, this information  
95322        shall be for the file resolved from the symbolic link, except that the <pathname> field shall  
95323        contain the pathname of the symbolic link itself. If the file is a symbolic link and the **-L** option is  
95324        not specified, this information shall be about the link itself and the <pathname> field shall be of  
95325        the form:

95326        "%s -> %s", <pathname of link>, <contents of link>

95327        XSI        The **-n**, **-g**, and **-o** options use the same format as **-l**, but with omitted items and their  
95328        associated <blank> characters. See the OPTIONS section.

95329        In both the preceding **-l** forms, if <owner name> or <group name> cannot be determined, or if **-n**  
95330        is given, they shall be replaced with their associated numeric values using the format %u.

95331        The <size> field shall contain the value that would be returned for the file in the *st\_size* field of  
95332        **struct stat** (see XBD <sys/stat.h>). Note that for some file types this value is unspecified.

95333        The <device info> field shall contain implementation-defined information associated with the  
95334        device in question.

95335        The <date and time> field shall contain the appropriate date and timestamp of when the file was  
95336        last modified. In the POSIX locale, the field shall be the equivalent of the output of the following  
95337        *date* command:

95338        date "+%b %e %H:%M"

95339        if the file has been modified in the last six months, or:

95340        date "+%b %e %Y"

95341        (where two <space> characters are used between %e and %Y) if the file has not been modified in  
95342        the last six months or if the modification date is in the future, except that, in both cases, the final  
95343        <newline> produced by *date* shall not be included and the output shall be as if the *date*  
95344        command were executed at the time of the last modification date of the file rather than the  
95345        current time. When the *LC\_TIME* locale category is not set to the POSIX locale, a different format  
95346        and order of presentation of this field may be used.

95347        If the pathname was specified as a *file* operand, it shall be written as specified.

95348        XSI        The file mode written under the **-l**, **-n**, **-g**, and **-o** options shall consist of the following format:

95349        "%c%s%s%s", <entry type>, <owner permissions>,  
95350               <group permissions>, <other permissions>,  
95351               <optional alternate access method flag>

95352 The *<optional alternate access method flag>* shall be the empty string if there is no alternate or  
 95353 additional access control method associated with the file; otherwise, it shall be a string  
 95354 containing a single printable character that is not a *<blank>*.

95355 The *<entry type>* character shall describe the type of file, as follows:

- 95356 d Directory.
- 95357 b Block special file.
- 95358 c Character special file.
- 95359 l (ell) Symbolic link.
- 95360 p FIFO.
- 95361 – Regular file.

95362 Implementations may add other characters to this list to represent other implementation-defined  
 95363 file types.

95364 The next three fields shall be three characters each:

95365 *<owner permissions>*

95366 Permissions for the file owner class (see XBD [Section 4.4](#), on page 108).

95367 *<group permissions>*

95368 Permissions for the file group class.

95369 *<other permissions>*

95370 Permissions for the file other class.

95371 Each field shall have three character positions:

- 95372 1. If 'r', the file is readable; if '–', the file is not readable.
- 95373 2. If 'w', the file is writable; if '–', the file is not writable.
- 95374 3. The first of the following that applies:
  - 95375 S If in *<owner permissions>*, the file is not executable and set-user-ID mode is set. If in
  - 95376 *<group permissions>*, the file is not executable and set-group-ID mode is set.
  - 95377 s If in *<owner permissions>*, the file is executable and set-user-ID mode is set. If in
  - 95378 *<group permissions>*, the file is executable and set-group-ID mode is set.
  - 95379 XSI T If in *<other permissions>* and the file is a directory, search permission is not granted to
  - 95380 others, and the restricted deletion flag is set.
  - 95381 XSI t If in *<other permissions>* and the file is a directory, search permission is granted to
  - 95382 others, and the restricted deletion flag is set.
  - 95383 x The file is executable or the directory is searchable.
  - 95384 – None of the attributes of 'S', 's', 'T', 't', or 'x' applies.

95385 Implementations may add other characters to this list for the third character position.  
 95386 Such additions shall, however, be written in lowercase if the file is executable or  
 95387 searchable, and in uppercase if it is not.

95388 XSI If any of the *–l*, *–n*, *–s*, *–g*, or *–o* options is specified, each list of files within the directory shall be  
 95389 preceded by a status line indicating the number of file system blocks occupied by files in the  
 95390 directory in 512-byte units if the *–k* option is not specified, or 1024-byte units if the *–k* option is  
 95391 specified, rounded up to the next integral number of units, if necessary. In the POSIX locale, the

95392 format shall be:

95393 "total %u\n", <number of units in the directory>

95394 If more than one directory, or a combination of non-directory files and directories are written,  
 95395 either as a result of specifying multiple operands, or the **-R** option, each list of files within a  
 95396 directory shall be preceded by:

95397 "\n%s:\n", <directory name>

95398 If this string is the first thing to be written, the first <newline> shall not be written. This output  
 95399 shall precede the number of units in the directory.

95400 If the **-s** option is given, each file shall be written with the number of blocks used by the file.  
 95401 XSI Along with **-C**, **-l**, **-m**, or **-x**, the number and a <space> shall precede the filename; with **-l**, **-n**,  
 95402 **-g**, or **-o**, they shall precede each line describing a file.

95403 **STDERR**

95404 The standard error shall be used only for diagnostic messages.

95405 **OUTPUT FILES**

95406 None.

95407 **EXTENDED DESCRIPTION**

95408 None.

95409 **EXIT STATUS**

95410 The following exit values shall be returned:

95411 0 Successful completion.

95412 >0 An error occurred.

95413 **CONSEQUENCES OF ERRORS**

95414 Default.

95415 **APPLICATION USAGE**

95416 Many implementations use the <equals-sign> ('=') to denote sockets bound to the file system  
 95417 for the **-F** option. Similarly, many historical implementations use the 's' character to denote  
 95418 sockets as the entry type characters for the **-l** option.

95419 It is difficult for an application to use every part of the file modes field of **ls -l** in a portable  
 95420 manner. Certain file types and executable bits are not guaranteed to be exactly as shown, as  
 95421 implementations may have extensions. Applications can use this field to pass directly to a user  
 95422 printout or prompt, but actions based on its contents should generally be deferred, instead, to  
 95423 the *test* utility.

95424 The output of **ls** (with the **-l** and related options) contains information that logically could be  
 95425 used by utilities such as *chmod* and *touch* to restore files to a known state. However, this  
 95426 information is presented in a format that cannot be used directly by those utilities or be easily  
 95427 translated into a format that can be used. A character has been added to the end of the  
 95428 permissions string so that applications at least have an indication that they may be working in  
 95429 an area they do not understand instead of assuming that they can translate the permissions  
 95430 string into something that can be used. Future versions or related documents may define one or  
 95431 more specific characters to be used based on different standard additional or alternative access  
 95432 control mechanisms.

95433 As with many of the utilities that deal with filenames, the output of **ls** for multiple files or in one  
 95434 of the long listing formats must be used carefully on systems where filenames can contain  
 95435 embedded white space. Systems and system administrators should institute policies and user

training to limit the use of such filenames.

The number of disk blocks occupied by the file that it reports varies depending on underlying file system type, block size units reported, and the method of calculating the number of blocks. On some file system types, the number is the actual number of blocks occupied by the file (counting indirect blocks and ignoring holes in the file); on others it is calculated based on the file size (usually making an allowance for indirect blocks, but ignoring holes).

## EXAMPLES

An example of a small directory tree being fully listed with *ls -laRF a* in the POSIX locale:

```
total 11
drwxr-xr-x 3 fox prog 64 Jul 4 12:07 ./
drwxrwxrwx 4 fox prog 3264 Jul 4 12:09 ../
drwxr-xr-x 2 fox prog 48 Jul 4 12:07 b/
-rwxr--r-- 1 fox prog 572 Jul 4 12:07 foo*

a/b:
total 4
drwxr-xr-x 2 fox prog 48 Jul 4 12:07 ./
drwxr-xr-x 3 fox prog 64 Jul 4 12:07 ../
-rw-r--r-- 1 fox prog 700 Jul 4 12:07 bar
```

## RATIONALE

Some historical implementations of the *ls* utility show all entries in a directory except dot and dot-dot when a superuser invokes *ls* without specifying the *-a* option. When “normal” users invoke *ls* without specifying *-a*, they should not see information about any files with names beginning with a <period> unless they were named as *file* operands.

Implementations are expected to traverse arbitrary depths when processing the *-R* option. The only limitation on depth should be based on running out of physical storage for keeping track of untraversed directories.

The *-1* (one) option was historically found in BSD and BSD-derived implementations only. It is required in this volume of POSIX.1-2008 so that conforming applications might ensure that output is one entry per line, even if the output is to a terminal.

The *-S* option was added in Issue 7, but had been provided by several implementations for many years. The description given in the standard documents historic practice, but does not match much of the documentation that described its behavior. Historical documentation typically described it as something like:

*-S* Sort by size (largest size first) instead of by name. Special character devices (listed last) are sorted by name.

even though the file type was never considered when sorting the output. Character special files do typically sort close to the end of the list because their file size on most implementations is zero. But they are sorted alphabetically with any other files that happen to have the same file size (zero), not sorted separately and added to the end.

This volume of POSIX.1-2008 is frequently silent about what happens when mutually-exclusive options are specified. Except for *-R*, *-d*, and *-f*, the *ls* utility is required to accept multiple options from each mutually-exclusive option set without treating them as errors and to use the behavior specified by the last option given in each mutually-exclusive set. Since *ls* is one of the most aliased commands, it is important that the implementation perform intuitively. For example, if the alias were:

```
alias ls="ls -C"
```

and the user typed `ls -1` (one), single-text-column output should result, not an error.

The `-g`, `-l` (ell), `-n`, and `-o` options are not mutually-exclusive options. They all enable long format output. They work together to determine whether the file's owner is written (no if `-g` is present), file's group is written (no if `-o` is present), and if the file's group or owner is written whether it is written as the name (default) or a string representation of the UID or GID number (if `-n` is present). The `-C`, `-m`, `-x`, and `-1` (one) are mutually-exclusive options and the first three of these disable long format output. The `-1` (one) option does not directly change whether or not long format output is enabled, but by overriding `-C`, `-m`, and `-x`, it can re-enable long format output that had been disabled by one of these options.

Earlier versions of this standard did not describe the BSD `-A` option (like `-a`, but dot and dot-dot are not written out). It has been added due to widespread implementation.

Implementations may make `-q` the default for terminals to prevent trojan horse attacks on terminals with special escape sequences. This is not required because:

- Some control characters may be useful on some terminals; for example, a system might write them as `"\001"` or `"^A"`.
- Special behavior for terminals is not relevant to applications portability.

An early proposal specified that the *<optional alternate access method flag>* had to be `'+'` if there was an alternate access method used on the file or `<space>` if there was not. This was changed to be `<space>` if there is not and a single printable character if there is. This was done for three reasons:

1. There are historical implementations using characters other than `'+'`.
2. There are implementations that vary this character used in that position to distinguish between various alternate access methods in use.
3. The standard developers did not want to preclude future specifications that might need a way to specify more than one alternate access method.

Nonetheless, implementations providing a single alternate access method are encouraged to use `'+'`.

Earlier versions of this standard did not have the `-k` option, which meant that the `-s` option could not be used portably as its block size was implementation-defined, and the units used to specify the number of blocks occupied by files in a directory in an `ls -l` listing were fixed as 512-byte units. The `-k` option has been added to provide a way for the `-s` option to be used portably, and for consistency it also changes the aforementioned units from 512-byte to 1024-byte.

The *<date and time>* field in the `-l` format is specified only for the POSIX locale. As noted, the format can be different in other locales. No mechanism for defining this is present in this volume of POSIX.1-2008, as the appropriate vehicle is a messaging system; that is, the format should be specified as a "message".

## FUTURE DIRECTIONS

Allowing `-f` to ignore the `-A`, `-g`, `-l`, `-n`, `-o`, and `-s` options may be removed in a future version.

## SEE ALSO

[\*chmod\*](#), [\*find\*](#)

XBD [Section 4.4](#) (on page 108), [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 215), [`<sys/stat.h>`](#)

XSH [\*fstatat\(\)\*](#)

95525 **CHANGE HISTORY**

95526 First released in Issue 2.

95527 **Issue 5**

95528 A second FUTURE DIRECTION is added.

95529 **Issue 6**95530 The following new requirements on POSIX implementations derive from alignment with the  
95531 Single UNIX Specification:

- 95532
- In the **-F** option, other symbols are allowed for other file types.

95533 Treatment of symbolic links is added, as defined in the IEEE P1003.2b draft standard.

95534 The Open Group Base Resolution bwg2001-010 is applied, adding the **T** and **t** fields as part of  
95535 the XSI option.95536 **Issue 7**95537 Austin Group Interpretation 1003.1-2001 #101 is applied, clarifying the optional alternate access  
95538 method flag in the STDOUT section.95539 Austin Group Interpretation 1003.1-2001 #128 is applied, clarifying the DESCRIPTION and the  
95540 definition of the **-R** option.95541 Austin Group Interpretation 1003.1-2001 #129 is applied, clarifying the behavior of *ls* when no  
95542 operands are specified.95543 Austin Group Interpretation 1003.1-2001 #198 is applied, clarifying the requirements for the **-H**  
95544 option.95545 SD5-XCU-ERN-50 is applied, adding the **-A** option.

95546 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

95547 The **-S** option is added from The Open Group Technical Standard, 2006, Extended API Set  
95548 Part 1.95549 The **-f**, **-m**, **-n**, **-p**, **-s**, and **-x** options are moved from the XSI option to the Base.95550 The description of the **-f**, **-s**, and **-t** options are revised and the **-k** option is added.95551 POSIX.1-2008, Technical Corrigendum 1, XCU/TC1-2008/0098 [424], XCU/TC1-2008/0099 +  
95552 [424], XCU/TC1-2008/0100 [424], XCU/TC1-2008/0101 [424], XCU/TC1-2008/0102 [424], +  
95553 XCU/TC1-2008/0103 [424], XCU/TC1-2008/0104 [424], XCU/TC1-2008/0105 [423,424], +  
95554 XCU/TC1-2008/0106 [424], XCU/TC1-2008/0107 [424], XCU/TC1-2008/0108 [424], +  
95555 XCU/TC1-2008/0109 [424], XCU/TC1-2008/0110 [424], XCU/TC1-2008/0111 [423], +  
95556 XCU/TC1-2008/0112 [117], XCU/TC1-2008/0113 [117], XCU/TC1-2008/0114 [117], +  
95557 XCU/TC1-2008/0115 [424], and XCU/TC1-2008/0116 [424] are applied.

95558 **NAME**

95559 m4 — macro processor

95560 **SYNOPSIS**95561 m4 [-s] [-D *name*[=*val*]]... [-U *name*]... *file*...95562 **DESCRIPTION**

95563 The *m4* utility is a macro processor that shall read one or more text files, process them according  
 95564 to their included macro statements, and write the results to standard output.

95565 **OPTIONS**

95566 The *m4* utility shall conform to XBD [Section 12.2](#) (on page 215), except that the order of the **-D**  
 95567 and **-U** options shall be significant, and options can be interspersed with operands.

95568 The following options shall be supported:

95569 **-s** Enable line synchronization output for the *c99* preprocessor phase (that is, **#line**  
 95570 directives).

95571 **-D *name*[=*val*]**

95572 Define *name* to *val* or to null if *=val* is omitted.

95573 **-U *name*** Undefine *name*.

95574 **OPERANDS**

95575 The following operand shall be supported:

95576 *file* A pathname of a text file to be processed. If no *file* is given, or if it is '-', the  
 95577 standard input shall be read.

95578 **STDIN**

95579 The standard input shall be a text file that is used if no *file* operand is given, or if it is '-'.

95580 **INPUT FILES**

95581 The input file named by the *file* operand shall be a text file.

95582 **ENVIRONMENT VARIABLES**

95583 The following environment variables shall affect the execution of *m4*:

95584 **LANG** Provide a default value for the internationalization variables that are unset or null.  
 95585 (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 95586 variables used to determine the values of locale categories.)

95587 **LC\_ALL** If set to a non-empty string value, override the values of all the other  
 95588 internationalization variables.

95589 **LC\_CTYPE** Determine the locale for the interpretation of sequences of bytes of text data as  
 95590 characters (for example, single-byte as opposed to multi-byte characters in  
 95591 arguments and input files).

95592 **LC\_MESSAGES**

95593 Determine the locale that should be used to affect the format and contents of  
 95594 diagnostic messages written to standard error.

95595 **XSI NLSPATH** Determine the location of message catalogs for the processing of **LC\_MESSAGES**.

95596 **ASYNCHRONOUS EVENTS**

95597 Default.



**STDOUT**

The standard output shall be the same as the input files, after being processed for macro expansion.

**STDERR**

The standard error shall be used to display strings with the **errprint** macro, macro tracing enabled by the **traceon** macro, the defined text for macros written by the **dumpdef** macro, or for diagnostic messages.

**OUTPUT FILES**

None.

**EXTENDED DESCRIPTION**

The *m4* utility shall compare each token from the input against the set of built-in and user-defined macros. If the token matches the name of a macro, then the token shall be replaced by the macro's defining text, if any, and rescanned for matching macro names. Once no portion of the token matches the name of a macro, it shall be written to standard output. Macros may have arguments, in which case the arguments shall be substituted into the defining text before it is rescanned.

Macro calls have the form:

```
name(arg1, arg2, ..., argn)
```

Macro names shall consist of letters, digits, and underscores, where the first character is not a digit. Tokens not of this form shall not be treated as macros.

The application shall ensure that the <left-parenthesis> immediately follows the name of the macro. If a token matching the name of a macro is not followed by a <left-parenthesis>, it is handled as a use of that macro without arguments.

If a macro name is followed by a <left-parenthesis>, its arguments are the <comma>-separated tokens between the <left-parenthesis> and the matching <right-parenthesis>. Unquoted white-space characters preceding each argument shall be ignored. All other characters, including trailing white-space characters, are retained. <comma> characters enclosed between <left-parenthesis> and <right-parenthesis> characters do not delimit arguments.

Arguments are positionally defined and referenced. The string "\$1" in the defining text shall be replaced by the first argument. Systems shall support at least nine arguments; only the first nine can be referenced, using the strings "\$1" to "\$9", inclusive. The string "\$0" is replaced with the name of the macro. The string "\$#" is replaced by the number of arguments as a string. The string "\$\*" is replaced by a list of all of the arguments, separated by <comma> characters. The string "\$@" is replaced by a list of all of the arguments separated by <comma> characters, and each argument is quoted using the current left and right quoting strings. The string "\${" produces unspecified behavior.

If fewer arguments are supplied than are in the macro definition, the omitted arguments are taken to be null. It is not an error if more arguments are supplied than are in the macro definition.

No special meaning is given to any characters enclosed between matching left and right quoting strings, but the quoting strings are themselves discarded. By default, the left quoting string consists of a grave accent (backquote) and the right quoting string consists of an acute accent (single-quote); see also the **changequote** macro.

Comments are written but not scanned for matching macro names; by default, the begin-comment string consists of the <number-sign> character and the end-comment string consists of a <newline>. See also the **changecom** and **dnl** macros.

95644 The *m4* utility shall make available the following built-in macros. They can be redefined, but  
 95645 once this is done the original meaning is lost. Their values shall be null unless otherwise stated.  
 95646 In the descriptions below, the term *defining text* refers to the value of the macro: the second  
 95647 argument to the **define** macro, among other things. Except for the first argument to the **eval**  
 95648 macro, all numeric arguments to built-in macros shall be interpreted as decimal values. The  
 95649 string values produced as the defining text of the **decr**, **divnum**, **incr**, **index**, **len**, and **sysval**  
 95650 built-in macros shall be in the form of a decimal-constant as defined in the C language.

95651 **changecom** The **changecom** macro shall set the begin-comment and end-comment strings.  
 95652 With no arguments, the comment mechanism shall be disabled. With a single non-  
 95653 null argument, that argument shall become the begin-comment and the <newline>  
 95654 shall become the end-comment string. With two non-null arguments, the first  
 95655 argument shall become the begin-comment string and the second argument shall  
 95656 become the end-comment string. The behavior is unspecified if either argument is  
 95657 provided but null. Systems shall support comment strings of at least five  
 95658 characters.

95659 **changequote** The **changequote** macro shall set the begin-quote and end-quote strings. With no  
 95660 arguments, the quote strings shall be set to the default values (that is, ` `'). The  
 95661 behavior is unspecified if there is a single argument or either argument is null.  
 95662 With two non-null arguments, the first argument shall become the begin-quote  
 95663 string and the second argument shall become the end-quote string. Systems shall  
 95664 support quote strings of at least five characters.

95665 **decr** The defining text of the **decr** macro shall be its first argument decremented by 1. It  
 95666 shall be an error to specify an argument containing any non-numeric characters.  
 95667 The behavior is unspecified if **decr** is not immediately followed by a <left-  
 95668 parenthesis>.

95669 **define** The second argument shall become the defining text of the macro whose name is  
 95670 the first argument. It is unspecified whether the **define** macro deletes all prior  
 95671 definitions of the macro named by its first argument or preserves all but the  
 95672 current definition of the macro. The behavior is unspecified if **define** is not  
 95673 immediately followed by a <left-parenthesis>.

95674 **defn** The defining text of the **defn** macro shall be the quoted definition (using the  
 95675 current quoting strings) of its arguments. The behavior is unspecified if **defn** is not  
 95676 immediately followed by a <left-parenthesis>.

95677 **divert** The *m4* utility maintains nine temporary buffers, numbered 1 to 9, inclusive.  
 95678 When the last of the input has been processed, any output that has been placed in  
 95679 these buffers shall be written to standard output in buffer-numerical order. The  
 95680 **divert** macro shall divert future output to the buffer specified by its argument.  
 95681 Specifying no argument or an argument of 0 shall resume the normal output  
 95682 process. Output diverted to a stream with a negative number shall be discarded.  
 95683 Behavior is implementation-defined if a stream number larger than 9 is specified. It  
 95684 shall be an error to specify an argument containing any non-numeric characters.

95685 **divnum** The defining text of the **divnum** macro shall be the number of the current output  
 95686 stream as a string.

95687 **dnl** The **dnl** macro shall cause *m4* to discard all input characters up to and including  
 95688 the next <newline>.

|       |                 |                                                                                                                                                                                                                                                                                                                                   |
|-------|-----------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 95689 | <b>dumpdef</b>  | The <b>dumpdef</b> macro shall write the defined text to standard error for each of the macros specified as arguments, or, if no arguments are specified, for all macros.                                                                                                                                                         |
| 95690 |                 |                                                                                                                                                                                                                                                                                                                                   |
| 95691 | <b>errprint</b> | The <b>errprint</b> macro shall write its arguments to standard error. The behavior is unspecified if <b>errprint</b> is not immediately followed by a <left-parenthesis>.                                                                                                                                                        |
| 95692 |                 |                                                                                                                                                                                                                                                                                                                                   |
| 95693 | <b>eval</b>     | The <b>eval</b> macro shall evaluate its first argument as an arithmetic expression, using signed integer arithmetic with at least 32-bit precision. At least the following C-language operators shall be supported, with precedence, associativity, and behavior as described in <a href="#">Section 1.1.2.1</a> (on page 2305): |
| 95694 |                 |                                                                                                                                                                                                                                                                                                                                   |
| 95695 |                 |                                                                                                                                                                                                                                                                                                                                   |
| 95696 |                 |                                                                                                                                                                                                                                                                                                                                   |
| 95697 |                 | ( )                                                                                                                                                                                                                                                                                                                               |
| 95698 |                 | unary +                                                                                                                                                                                                                                                                                                                           |
| 95699 |                 | unary -                                                                                                                                                                                                                                                                                                                           |
| 95700 |                 | ~                                                                                                                                                                                                                                                                                                                                 |
| 95701 |                 | !                                                                                                                                                                                                                                                                                                                                 |
| 95702 |                 | binary *                                                                                                                                                                                                                                                                                                                          |
| 95703 |                 | /                                                                                                                                                                                                                                                                                                                                 |
| 95704 |                 | %                                                                                                                                                                                                                                                                                                                                 |
| 95705 |                 | binary +                                                                                                                                                                                                                                                                                                                          |
| 95706 |                 | binary -                                                                                                                                                                                                                                                                                                                          |
| 95707 |                 | <<                                                                                                                                                                                                                                                                                                                                |
| 95708 |                 | >>                                                                                                                                                                                                                                                                                                                                |
| 95709 |                 | <                                                                                                                                                                                                                                                                                                                                 |
| 95710 |                 | <=                                                                                                                                                                                                                                                                                                                                |
| 95711 |                 | >                                                                                                                                                                                                                                                                                                                                 |
| 95712 |                 | >=                                                                                                                                                                                                                                                                                                                                |
| 95713 |                 | ==                                                                                                                                                                                                                                                                                                                                |
| 95714 |                 | !=                                                                                                                                                                                                                                                                                                                                |
| 95715 |                 | binary &                                                                                                                                                                                                                                                                                                                          |
| 95716 |                 | ^                                                                                                                                                                                                                                                                                                                                 |
| 95717 |                 |                                                                                                                                                                                                                                                                                                                                   |
| 95718 |                 | &&                                                                                                                                                                                                                                                                                                                                |
| 95719 |                 |                                                                                                                                                                                                                                                                                                                                   |
| 95720 |                 | Systems shall support octal and hexadecimal numbers as in the ISO C standard.                                                                                                                                                                                                                                                     |
| 95721 |                 | The second argument, if specified, shall set the radix for the result; if the argument                                                                                                                                                                                                                                            |
| 95722 |                 | is blank or unspecified, the default is 10. Behavior is unspecified if the radix falls                                                                                                                                                                                                                                            |
| 95723 |                 | outside the range 2 to 36, inclusive. The third argument, if specified, sets the                                                                                                                                                                                                                                                  |
| 95724 |                 | minimum number of digits in the result. Behavior is unspecified if the third                                                                                                                                                                                                                                                      |
| 95725 |                 | argument is less than zero. It shall be an error to specify the second or third                                                                                                                                                                                                                                                   |
| 95726 |                 | argument containing any non-numeric characters. The behavior is unspecified if                                                                                                                                                                                                                                                    |
| 95727 |                 | <b>eval</b> is not immediately followed by a <left-parenthesis>.                                                                                                                                                                                                                                                                  |
| 95728 | <b>ifdef</b>    | If the first argument to the <b>ifdef</b> macro is defined, the defining text shall be the                                                                                                                                                                                                                                        |
| 95729 |                 | second argument. Otherwise, the defining text shall be the third argument, if                                                                                                                                                                                                                                                     |
| 95730 |                 | specified, or the null string, if not. The behavior is unspecified if <b>ifdef</b> is not                                                                                                                                                                                                                                         |
| 95731 |                 | immediately followed by a <left-parenthesis>.                                                                                                                                                                                                                                                                                     |
| 95732 | <b>ifelse</b>   | The <b>ifelse</b> macro takes three or more arguments. If the first two arguments                                                                                                                                                                                                                                                 |
| 95733 |                 | compare as equal strings (after macro expansion of both arguments), the defining                                                                                                                                                                                                                                                  |
| 95734 |                 | text shall be the third argument. If the first two arguments do not compare as equal                                                                                                                                                                                                                                              |
| 95735 |                 | strings and there are three arguments, the defining text shall be null. If the first two                                                                                                                                                                                                                                          |
| 95736 |                 | arguments do not compare as equal strings and there are four or five arguments,                                                                                                                                                                                                                                                   |

|       |                    |                                                                                                 |
|-------|--------------------|-------------------------------------------------------------------------------------------------|
| 95737 |                    | the defining text shall be the fourth argument. If the first two arguments do not               |
| 95738 |                    | compare as equal strings and there are six or more arguments, the first three                   |
| 95739 |                    | arguments shall be discarded and processing shall restart with the remaining                    |
| 95740 |                    | arguments. The behavior is unspecified if <b>ifelse</b> is not immediately followed by a        |
| 95741 |                    | <left-parenthesis>.                                                                             |
| 95742 | <b>include</b>     | The defining text for the <b>include</b> macro shall be the contents of the file named by       |
| 95743 |                    | the first argument. It shall be an error if the file cannot be read. The behavior is            |
| 95744 |                    | unspecified if <b>include</b> is not immediately followed by a <left-parenthesis>.              |
| 95745 | <b>incr</b>        | The defining text of the <b>incr</b> macro shall be its first argument incremented by 1. It     |
| 95746 |                    | shall be an error to specify an argument containing any non-numeric characters.                 |
| 95747 |                    | The behavior is unspecified if <b>incr</b> is not immediately followed by a <left-              |
| 95748 |                    | parenthesis>.                                                                                   |
| 95749 | <b>index</b>       | The defining text of the <b>index</b> macro shall be the first character position (as a         |
| 95750 |                    | string) in the first argument where a string matching the second argument begins                |
| 95751 |                    | (zero origin), or -1 if the second argument does not occur. The behavior is                     |
| 95752 |                    | unspecified if <b>index</b> is not immediately followed by a <left-parenthesis>.                |
| 95753 | <b>len</b>         | The defining text of the <b>len</b> macro shall be the length (as a string) of the first        |
| 95754 |                    | argument. The behavior is unspecified if <b>len</b> is not immediately followed by a            |
| 95755 |                    | <left-parenthesis>.                                                                             |
| 95756 | <b>m4exit</b>      | Exit from the <i>m4</i> utility. If the first argument is specified, it is the exit code. The   |
| 95757 |                    | default is zero. It shall be an error to specify an argument containing any non-                |
| 95758 |                    | numeric characters.                                                                             |
| 95759 | <b>m4wrap</b>      | The first argument shall be processed when EOF is reached. If the <b>m4wrap</b> macro           |
| 95760 |                    | is used multiple times, the arguments specified shall be processed in the order in              |
| 95761 |                    | which the <b>m4wrap</b> macros were processed. The behavior is unspecified if <b>m4wrap</b>     |
| 95762 |                    | is not immediately followed by a <left-parenthesis>.                                            |
| 95763 | OB <b>maketemp</b> | The defining text shall be the first argument, with any trailing 'X' characters                 |
| 95764 |                    | replaced with the current process ID as a string. The behavior is unspecified if                |
| 95765 |                    | <b>maketemp</b> is not immediately followed by a <left-parenthesis>.                            |
| 95766 | <b>mkstemp</b>     | The first argument shall be taken as a template for creating an empty file, with                |
| 95767 |                    | trailing 'X' characters replaced with characters from the portable filename                     |
| 95768 |                    | character set. The behavior is unspecified if the first argument does not end in at             |
| 95769 |                    | least six 'X' characters. If a temporary file is successfully created, then the                 |
| 95770 |                    | defining text of the macro shall be the name of the new file. The user ID of the file           |
| 95771 |                    | shall be set to the effective user ID of the process. The group ID of the file shall be         |
| 95772 |                    | set to the group ID of the file's parent directory or to the effective group ID of the          |
| 95773 |                    | process. The file access permission bits are set such that only the owner can both              |
| 95774 |                    | read and write the file, regardless of the current <i>umask</i> of the process. If a file could |
| 95775 |                    | not be created, the defining text of the macro shall be the empty string. The                   |
| 95776 |                    | behavior is unspecified if <b>mkstemp</b> is not immediately followed by a <left-               |
| 95777 |                    | parenthesis>.                                                                                   |
| 95778 | <b>popdef</b>      | The <b>popdef</b> macro shall delete the current definition of its arguments, replacing         |
| 95779 |                    | that definition with the previous one. If there is no previous definition, the macro            |
| 95780 |                    | is undefined. The behavior is unspecified if <b>popdef</b> is not immediately followed by       |
| 95781 |                    | a <left-parenthesis>.                                                                           |

|       |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |  |
|-------|-----------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--|
| 95782 | <b>pushdef</b>  | The <b>pushdef</b> macro shall be equivalent to the <b>define</b> macro with the exception that it shall preserve any current definition for future retrieval using the <b>popdef</b> macro. The behavior is unspecified if <b>pushdef</b> is not immediately followed by a <left-parenthesis>.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |  |
| 95783 |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |  |
| 95784 |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |  |
| 95785 |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |  |
| 95786 | <b>shift</b>    | The defining text for the <b>shift</b> macro shall be a comma-separated list of its arguments except the first one. Each argument shall be quoted using the current quoting strings. The behavior is unspecified if <b>shift</b> is not immediately followed by a <left-parenthesis>.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |  |
| 95787 |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |  |
| 95788 |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |  |
| 95789 |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |  |
| 95790 | <b>sinclude</b> | The <b>sinclude</b> macro shall be equivalent to the <b>include</b> macro, except that it shall not be an error if the file is inaccessible. The behavior is unspecified if <b>sinclude</b> is not immediately followed by a <left-parenthesis>.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |  |
| 95791 |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |  |
| 95792 |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |  |
| 95793 | <b>substr</b>   | The defining text for the <b>substr</b> macro shall be the substring of the first argument beginning at the zero-offset character position specified by the second argument. The third argument, if specified, shall be the number of characters to select; if not specified, the characters from the starting point to the end of the first argument shall become the defining text. It shall not be an error to specify a starting point beyond the end of the first argument and the defining text shall be null. It shall be an error to specify an argument containing any non-numeric characters. The behavior is unspecified if <b>substr</b> is not immediately followed by a <left-parenthesis>.                                                                            |  |
| 95794 |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |  |
| 95795 |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |  |
| 95796 |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |  |
| 95797 |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |  |
| 95798 |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |  |
| 95799 |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |  |
| 95800 |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |  |
| 95801 |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |  |
| 95802 | <b>syscmd</b>   | The <b>syscmd</b> macro shall interpret its first argument as a shell command line. The defining text shall be the string result of that command. The string result shall not be rescanned for macros while setting the defining text. No output redirection shall be performed by the <i>m4</i> utility. The exit status value from the command can be retrieved using the <b>sysval</b> macro. The behavior is unspecified if <b>syscmd</b> is not immediately followed by a <left-parenthesis>.                                                                                                                                                                                                                                                                                   |  |
| 95803 |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |  |
| 95804 |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |  |
| 95805 |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |  |
| 95806 |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |  |
| 95807 |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |  |
| 95808 | <b>sysval</b>   | The defining text of the <b>sysval</b> macro shall be the exit value of the utility last invoked by the <b>syscmd</b> macro (as a string).                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |  |
| 95809 |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |  |
| 95810 | <b>traceon</b>  | The <b>traceon</b> macro shall enable tracing for the macros specified as arguments, or, if no arguments are specified, for all macros. The trace output shall be written to standard error in an unspecified format.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |  |
| 95811 |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |  |
| 95812 |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |  |
| 95813 | <b>traceoff</b> | The <b>traceoff</b> macro shall disable tracing for the macros specified as arguments, or, if no arguments are specified, for all macros.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |  |
| 95814 |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |  |
| 95815 | <b>translit</b> | The defining text of the <b>translit</b> macro shall be the first argument with every character that occurs in the second argument replaced with the corresponding character from the third argument. If no replacement character is specified for some source character because the second argument is longer than the third argument, that character shall be deleted from the first argument in <b>translit</b> 's defining text. The behavior is unspecified if the '-' character appears within the second or third argument anywhere besides the first or last character. The behavior is unspecified if the same character appears more than once in the second argument. The behavior is unspecified if <b>translit</b> is not immediately followed by a <left-parenthesis>. |  |
| 95816 |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |  |
| 95817 |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |  |
| 95818 |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |  |
| 95819 |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |  |
| 95820 |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |  |
| 95821 |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |  |
| 95822 |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |  |
| 95823 |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |  |
| 95824 |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |  |
| 95825 | <b>undefine</b> | The <b>undefine</b> macro shall delete all definitions (including those preserved using the <b>pushdef</b> macro) of the macros named by its arguments. The behavior is unspecified if <b>undefine</b> is not immediately followed by a <left-parenthesis>.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |  |
| 95826 |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |  |
| 95827 |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |  |

95828       **undivert**    The **undivert** macro shall cause immediate output of any text in temporary buffers  
 95829                    named as arguments, or all temporary buffers if no arguments are specified.  
 95830                    Buffers can be undiverted into other temporary buffers. Undiverting shall discard  
 95831                    the contents of the temporary buffer. The behavior is unspecified if an argument  
 95832                    contains any non-numeric characters.

#### 95833 **EXIT STATUS**

95834        The following exit values shall be returned:

95835            0    Successful completion.

95836            >0   An error occurred

95837        If the **m4exit** macro is used, the exit value can be specified by the input file.

#### 95838 **CONSEQUENCES OF ERRORS**

95839        Default.

#### 95840 **APPLICATION USAGE**

95841        The **defn** macro is useful for renaming macros, especially built-ins.

95842        Since **eval** defers to the ISO C standard, some operations have undefined behavior. In some  
 95843        implementations, division or remainder by zero cause a fatal signal, even if the division occurs  
 95844        on the short-circuited branch of "&&" or "||". Any operation that overflows in signed  
 95845        arithmetic produces undefined behavior. Likewise, using the **shift** operators with a shift amount  
 95846        that is not positive and smaller than the precision is undefined, as is shifting a negative number  
 95847        to the right. Historically, not all implementations obeyed C-language precedence rules: '~' and  
 95848        '!' were lower than '=='; '==' and '!=' were not lower than '<'; and '|' was not lower  
 95849        than '^'; the liberal use of "()" can force the desired precedence even with these non-  
 95850        compliant implementations. Furthermore, some traditional implementations treated '^' as an  
 95851        exponentiation operator, although most implementations now use "\*\*\*" as an extension for this  
 95852        purpose.

95853        When a macro has been multiply defined via the **pushdef** macro, it is unspecified whether the  
 95854        **define** macro will alter only the most recent definition (as though by **popdef** and **pushdef**), or  
 95855        replace the entire stack of definitions with a single definition (as though by **undefine** and  
 95856        **pushdef**). An application desiring particular behavior for the **define** macro in this case can  
 95857        redefine it accordingly.

95858        Applications should use the **mkstemp** macro instead of the obsolescent **maketemp** macro for  
 95859        creating temporary files.

#### 95860 **EXAMPLES**

95861        If the file **m4src** contains the lines:

```
95862 The value of 'VER' is "VER".
95863 ifdef('VER', ``VER'' is defined to be VER., VER is not defined.)
95864 ifelse(VER, 1, ``VER'' is 'VER'.)
95865 ifelse(VER, 2, ``VER'' is 'VER'., ``VER'' is not 2.)
95866 end
```

95867        then the command

95868        **m4 m4src**

95869        or the command:

95870        **m4 -U VER m4src**

95871        produces the output:

95872       The value of VER is "VER".  
95873       VER is not defined.  
  
95874       VER is not 2.  
95875       end

95876       **The command:**

95877       m4 -D VER m4src

95878       **produces the output:**

95879       The value of VER is "".  
95880       VER is defined to be .  
  
95881       VER is not 2.  
95882       end

95883       **The command:**

95884       m4 -D VER=1 m4src

95885       **produces the output:**

95886       The value of VER is "1".  
95887       VER is defined to be 1.  
95888       VER is 1.  
95889       VER is not 2.  
95890       end

95891       **The command:**

95892       m4 -D VER=2 m4src

95893       **produces the output:**

95894       The value of VER is "2".  
95895       VER is defined to be 2.  
  
95896       VER is 2.  
95897       end

#### 95898 **RATIONALE**

95899       Historic System V-based behavior treated "\${" in a macro definition as two literal characters.  
95900       However, this sequence is left unspecified so that implementations may offer extensions such as  
95901       "\${11}" meaning the eleventh positional parameter. Macros can still be defined with  
95902       appropriate uses of nested quoting to result in a literal "\${" in the output after rescanning  
95903       removes the nested quotes.

95904       In the **translit** built-in, historic System V-based behavior treated '–' as a literal; GNU behavior  
95905       treats it as a range. This version of the standard allows either behavior.

#### 95906 **FUTURE DIRECTIONS**

95907       None.

#### 95908 **SEE ALSO**

95909       c99

95910       XBD [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 215)

**CHANGE HISTORY**

First released in Issue 2.

**Issue 5**

The phrase “the defined text for macros written by the **dumpdef** macro” is added to the description of **STDERR**, and the description of **dumpdef** is updated to indicate that output is written to standard error. The description of **eval** is updated to indicate that the list of excluded C operators excludes unary **'&'** and **'.'**. In the description of **ifdef**, the phrase “and it is not defined to be zero” is deleted.

**Issue 6**

In the EXTENDED DESCRIPTION, the **eval** text is updated to include a **'&'** character in the excepted list.

The EXTENDED DESCRIPTION of **divert** is updated to clarify that there are only nine diversion buffers.

The normative text is reworded to avoid use of the term “must” for application requirements.

The Open Group Base Resolution bwg2000-006 is applied.

IEEE Std 1003.1-2001/Cor 1-2002, item XCU/TC1/D6/31 is applied, replacing the EXAMPLES section.

**Issue 7**

Austin Group Interpretation 1003.1-2001 #117 is applied, marking the **maketemp** macro obsolescent and adding a new **mkstemp** macro.

Austin Group Interpretation 1003.1-2001 #207 is applied, clarifying the handling of white-space characters that precede or trail any macro arguments.

SD5-XCU-ERN-6 is applied, clarifying that Guideline 9 of the Utility Syntax Guidelines does not apply (options can be interspersed with operands).

SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

SD5-XCU-ERN-99 is applied, clarifying the definition of the **divert** macro in the EXTENDED DESCRIPTION.

SD5-XCU-ERN-100 is applied, clarifying the definition of the **syscmd** macro in the EXTENDED DESCRIPTION.

SD5-XCU-ERN-101 is applied, clarifying the definition of the **undivert** macro in the EXTENDED DESCRIPTION.

SD5-XCU-ERN-111 is applied to the EXTENDED DESCRIPTION, clarifying that the string **"\${ "** produces unspecified behavior.

SD5-XCU-ERN-112 is applied, updating the **changequote** macro.

SD5-XCU-ERN-118 is applied, clarifying the definition of the **define** macro in the EXTENDED DESCRIPTION and APPLICATION USAGE sections.

SD5-XCU-ERN-119 is applied, clarifying the definition of the **translit** macro in the EXTENDED DESCRIPTION and RATIONALE sections.

SD5-XCU-ERN-130, SD5-XCU-ERN-131, and SD5-XCU-ERN-137 are applied.

The **m4** utility is moved from the XSI option to the Base.



95951  
95952

POSIX.1-2008, Technical Corrigendum 1, XCU/TC1-2008/0117 [241], XCU/TC1-2008/0118 +  
[242,431], XCU/TC1-2008/0119 [242,431], and XCU/TC1-2008/0120 [325,430] are applied.

95953 **NAME**

95954 mailx — process messages

95955 **SYNOPSIS**95956 **Send Mode**95957 mailx [-s *subject*] *address...*95958 **Receive Mode**

```
95959 UP mailx -e
95960 mailx [-HiNn] [-F] [-u user]
95961 mailx -f [-HiNn] [-F] [file]
```

95962 **DESCRIPTION**

95963 The *mailx* utility provides a message sending and receiving facility. It has two major modes,  
 95964 selected by the options used: Send Mode and Receive Mode.

95965 On systems that do not support the User Portability Utilities option, an application using *mailx*  
 95966 shall have the ability to send messages in an unspecified manner (Send Mode). Unless the first  
 95967 character of one or more lines is <tilde> (' ~ '), all characters in the input message shall appear in  
 95968 the delivered message, but additional characters may be inserted in the message before it is  
 95969 retrieved.

95970 UP On systems supporting the User Portability Utilities option, mail-receiving capabilities and other  
 95971 interactive features, Receive Mode, described below, also shall be enabled.

95972 **Send Mode**

95973 Send Mode can be used by applications or users to send messages from the text in standard  
 95974 input.

95975 UP **Receive Mode**

95976 Receive Mode is more oriented towards interactive users. Mail can be read and sent in this  
 95977 interactive mode.

95978 When reading mail, *mailx* provides commands to facilitate saving, deleting, and responding to  
 95979 messages. When sending mail, *mailx* allows editing, reviewing, and other modification of the  
 95980 message as it is entered.

95981 Incoming mail shall be stored in one or more unspecified locations for each user, collectively  
 95982 called the system *mailbox* for that user. When *mailx* is invoked in Receive Mode, the system  
 95983 mailbox shall be the default place to find new mail. As messages are read, they shall be marked  
 95984 to be moved to a secondary file for storage, unless specific action is taken. This secondary file is  
 95985 called the **mbox** and is normally located in the directory referred to by the *HOME* environment  
 95986 variable (see *MBOX* in the ENVIRONMENT VARIABLES section for a description of this file).  
 95987 Messages shall remain in this file until explicitly removed. When the **-f** option is used to read  
 95988 mail messages from secondary files, messages shall be retained in those files unless specifically  
 95989 removed. All three of these locations—system mailbox, **mbox**, and secondary file—are referred  
 95990 to in this section as simply “mailboxes”, unless more specific identification is required.

## OPTIONS

The *mailx* utility shall conform to XBD [Section 12.2](#) (on page 215).

The following options shall be supported. (Only the `-s subject` option shall be required on all systems. The other options are required only on systems supporting the User Portability Utilities option.)

**-e** Test for the presence of mail in the system mailbox. The *mailx* utility shall write nothing and exit with a successful return code if there is mail to read.

**-f** Read messages from the file named by the *file* operand instead of the system mailbox. (See also **folder**.) If no *file* operand is specified, read messages from **mbox** instead of the system mailbox.

**-F** Record the message in a file named after the first recipient. The name is the login-name portion of the address found first on the **To:** line in the mail header. Overrides the **record** variable, if set (see [Internal Variables in mailx](#), on page 2915).

**-H** Write a header summary only.

**-i** Ignore interrupts. (See also **ignore**.)

**-n** Do not initialize from the system default start-up file. See the EXTENDED DESCRIPTION section.

**-N** Do not write an initial header summary.

**-s subject** Set the **Subject** header field to *subject*. All characters in the *subject* string shall appear in the delivered message. The results are unspecified if *subject* is longer than {LINE\_MAX} – 10 bytes or contains a <newline>.

**-u user** Read the system mailbox of the login name *user*. This shall only be successful if the invoking user has appropriate privileges to read the system mailbox of that user.

## OPERANDS

The following operands shall be supported:

*address* Addressee of message. When **-n** is specified and no user start-up files are accessed (see the EXTENDED DESCRIPTION section), the user or application shall ensure this is an address to pass to the mail delivery system. Any system or user start-up files may enable aliases (see **alias** under [Commands in mailx](#), on page 2918) that may modify the form of *address* before it is passed to the mail delivery system.

*file* A pathname of a file to be read instead of the system mailbox when **-f** is specified. The meaning of the *file* option-argument shall be affected by the contents of the **folder** internal variable; see [Internal Variables in mailx](#) (on page 2915).

## STDIN

When *mailx* is invoked in Send Mode (the first synopsis line), standard input shall be the message to be delivered to the specified addresses. When in Receive Mode, user commands shall be accepted from *stdin*. If the User Portability Utilities option is not supported, standard input lines beginning with a <tilde> ('~') character produce unspecified results.

If the User Portability Utilities option is supported, then in both Send and Receive Modes, standard input lines beginning with the escape character (usually <tilde> ('~')) shall affect processing as described in [Command Escapes in mailx](#) (on page 2927).

## INPUT FILES

When *mailx* is used as described by this volume of POSIX.1-2008, the *file* option-argument (see the **-f** option) and the **mbox** shall be text files containing mail messages, formatted as described in the OUTPUT FILES section. The nature of the system mailbox is unspecified; it need not be a file.

## ENVIRONMENT VARIABLES

UP Some of the functionality described in this section shall be provided on implementations that support the User Portability Utilities option as described in the text, and is not further shaded for this option.

The following environment variables shall affect the execution of *mailx*:

**DEAD** Determine the pathname of the file in which to save partial messages in case of interrupts or delivery errors. The default shall be **dead.letter** in the directory named by the *HOME* variable. The behavior of *mailx* in saving partial messages is unspecified if the User Portability Utilities option is not supported and *DEAD* is not defined with the value **/dev/null**.

XSI **EDITOR** Determine the name of a utility to invoke when the **edit** (see [Commands in mailx](#), on page 2918) or **~e** (see [Command Escapes in mailx](#), on page 2927) command is used. The default editor is unspecified. On XSI-conformant systems it is *ed*. The effects of this variable are unspecified if the User Portability Utilities option is not supported.

**HOME** Determine the pathname of the user's home directory.

**LANG** Provide a default value for the internationalization variables that are unset or null. (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization variables used to determine the values of locale categories.)

**LC\_ALL** If set to a non-empty string value, override the values of all the other internationalization variables.

**LC\_CTYPE** Determine the locale for the interpretation of sequences of bytes of text data as characters (for example, single-byte as opposed to multi-byte characters in arguments and input files) and the handling of case-insensitive address and header-field comparisons.

**LC\_TIME** This variable may determine the format and contents of the date and time strings written by *mailx*. This volume of POSIX.1-2008 specifies the effects of this variable only for systems supporting the User Portability Utilities option.

**LC\_MESSAGES**

Determine the locale that should be used to affect the format and contents of diagnostic messages written to standard error and informative messages written to standard output.

**LISTER** Determine a string representing the command for writing the contents of the **folder** directory to standard output when the **folders** command is given (see **folders** in [Commands in mailx](#), on page 2918). Any string acceptable as a *command\_string* operand to the *sh -c* command shall be valid. If this variable is null or not set, the output command shall be *ls*. The effects of this variable are unspecified if the User Portability Utilities option is not supported.

**MAILRC** Determine the pathname of the start-up file. The default shall be **.mailrc** in the directory referred to by the *HOME* environment variable. The behavior of *mailx* is unspecified if the User Portability Utilities option is not supported and *MAILRC* is

|       |                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|-------|----------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 96079 |                            | not defined with the value <code>/dev/null</code> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 96080 | <i>MBOX</i>                | Determine a pathname of the file to save messages from the system mailbox that have been read. The <b>exit</b> command shall override this function, as shall saving the message explicitly in another file. The default shall be <b>mbox</b> in the directory named by the <i>HOME</i> variable. The effects of this variable are unspecified if the User Portability Utilities option is not supported.                                                                                                                                                                                                                                                                                                                                                                    |
| 96081 |                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 96082 |                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 96083 |                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 96084 |                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 96085 | XSI                        | <i>NLSPATH</i> Determine the location of message catalogs for the processing of <i>LC_MESSAGES</i> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 96086 | <i>PAGER</i>               | Determine a string representing an output filtering or pagination command for writing the output to the terminal. Any string acceptable as a <i>command_string</i> operand to the <i>sh -c</i> command shall be valid. When standard output is a terminal device, the message output shall be piped through the command if the <i>mailx</i> internal variable <b>cr</b> t is set to a value less the number of lines in the message; see <a href="#">Internal Variables in mailx</a> (on page 2915). If the <i>PAGER</i> variable is null or not set, the paginator shall be either <i>more</i> or another paginator utility documented in the system documentation. The effects of this variable are unspecified if the User Portability Utilities option is not supported. |
| 96087 |                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 96088 |                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 96089 |                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 96090 |                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 96091 |                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 96092 |                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 96093 |                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 96094 |                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 96095 | <i>SHELL</i>               | Determine the name of a preferred command interpreter. The default shall be <i>sh</i> . The effects of this variable are unspecified if the User Portability Utilities option is not supported.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 96096 |                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 96097 |                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 96098 | <i>TERM</i>                | If the internal variable <b>screen</b> is not specified, determine the name of the terminal type to indicate in an unspecified manner the number of lines in a screenful of headers. If <i>TERM</i> is not set or is set to null, an unspecified default terminal type shall be used and the value of a screenful is unspecified. The effects of this variable are unspecified if the User Portability Utilities option is not supported.                                                                                                                                                                                                                                                                                                                                    |
| 96099 |                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 96100 |                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 96101 |                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 96102 |                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 96103 | <i>TZ</i>                  | This variable may determine the timezone used to calculate date and time strings written by <i>mailx</i> . If <i>TZ</i> is unset or null, an unspecified default timezone shall be used.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 96104 |                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 96105 |                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 96106 | <i>VISUAL</i>              | Determine a pathname of a utility to invoke when the <b>visual</b> command (see <a href="#">Commands in mailx</a> , on page 2918) or <b>~v</b> command-escape (see <a href="#">Command Escapes in mailx</a> , on page 2927) is used. If this variable is null or not set, the full-screen editor shall be <i>vi</i> . The effects of this variable are unspecified if the User Portability Utilities option is not supported.                                                                                                                                                                                                                                                                                                                                                |
| 96107 |                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 96108 |                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 96109 |                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 96110 |                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 96111 | <b>ASYNCHRONOUS EVENTS</b> |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 96112 |                            | When <i>mailx</i> is in Send Mode and standard input is not a terminal, it shall take the standard action for all signals.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 96113 |                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 96114 | UP                         | In <b>Receive Mode</b> , or in Send Mode when standard input is a terminal, if a SIGINT signal is received:                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 96115 |                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 96116 | UP                         | 1. If in command mode, the current command, if there is one, shall be aborted, and a command-mode prompt shall be written.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 96117 |                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 96118 |                            | 2. If in input mode:                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 96119 | UP                         | a. If <b>ignore</b> is set, <i>mailx</i> shall write "@\n", discard the current input line, and continue processing, bypassing the message-abort mechanism described in item 2b.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| 96120 |                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 96121 |                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |

96122 UP           b. If the interrupt was received while sending mail, either when in **Receive Mode** or  
 96123           **in Send Mode**, a message shall be written, and another subsequent interrupt, with  
 96124           no other intervening characters typed, shall be required to abort the mail message.  
 96125 UP           **If in Receive Mode and another interrupt is received, a command-mode prompt**  
 96126           **shall be written.** If in Send Mode and another interrupt is received, *mailx* shall  
 96127           terminate with a non-zero status.

96128           In both cases listed in item b, if the message is not empty:

96129 UP           i. If **save** is enabled and the file named by *DEAD* can be created, the message  
 96130           shall be written to the file named by *DEAD*. If the file exists, the message  
 96131           shall be written to replace the contents of the file.  
 96132 UP           ii. If **save** is not enabled, or the file named by *DEAD* cannot be created, the  
 96133           message shall not be saved.

96134           The *mailx* utility shall take the standard action for all other signals.

## 96135 **STDOUT**

96136           In command and input modes, all output, including prompts and messages, shall be written to  
 96137           standard output.

## 96138 **STDERR**

96139           The standard error shall be used only for diagnostic messages.

## 96140 **OUTPUT FILES**

96141           Various *mailx* commands and command escapes can create or add to files, including the **mbox**,  
 96142           the dead-letter file, and secondary mailboxes. When *mailx* is used as described in this volume of  
 96143           POSIX.1-2008, these files shall be text files, formatted as follows:

96144           line beginning with **From<space>**  
 96145           [one or more *header-lines*; see [Commands in mailx](#) (on page 2918)]  
 96146           empty line  
 96147           [zero or more *body lines*  
 96148           empty line]  
 96149           [line beginning with **From<space>...**]

96150           where each message begins with the **From <space>** line shown, preceded by the beginning of  
 96151           the file or an empty line. (The **From <space>** line is considered to be part of the message header,  
 96152           but not one of the header-lines referred to in [Commands in mailx](#) (on page 2918); thus, it shall  
 96153           not be affected by the **discard**, **ignore**, or **retain** commands.) The formats of the remainder of the  
 96154           **From <space>** line and any additional header lines are unspecified, except that none shall be  
 96155           empty. The format of a message body line is also unspecified, except that no line following an  
 96156           empty line shall start with **From <space>**; *mailx* shall modify any such user-entered message  
 96157           body lines (following an empty line and beginning with **From <space>**) by adding one or more  
 96158           characters to precede the 'F'; it may add these characters to **From <space>** lines that are not  
 96159           preceded by an empty line.

96160           When a message from the system mailbox or entered by the user is not a text file, it is  
 96161           implementation-defined how such a message is stored in files written by *mailx*.

## 96162 **EXTENDED DESCRIPTION**

96163 UP           The functionality in the entire EXTENDED DESCRIPTION section shall be provided on  
 96164           implementations supporting the User Portability Utilities option. The functionality described in  
 96165           this section shall be provided on implementations that support the User Portability Utilities  
 96166           option (and the rest of this section is not further shaded for this option).

The *mailx* utility need not support for all character encodings in all circumstances. For example, inter-system mail may be restricted to 7-bit data by the underlying network, 8-bit data need not be portable to non-internationalized systems, and so on. Under these circumstances, it is recommended that only characters defined in the ISO/IEC 646:1991 standard International Reference Version (equivalent to ASCII) 7-bit range of characters be used.

When *mailx* is invoked using one of the Receive Mode synopsis forms, it shall write a page of header-summary lines (if `-N` was not specified and there are messages, see below), followed by a prompt indicating that *mailx* can accept regular commands (see [Commands in mailx](#), on page 2918); this is termed *command mode*. The page of header-summary lines shall contain the first new message if there are new messages, or the first unread message if there are unread messages, or the first message. When *mailx* is invoked using the Send Mode synopsis and standard input is a terminal, if no subject is specified on the command line and the **asksub** variable is set, a prompt for the subject shall be written. At this point, *mailx* shall be in input mode. This input mode shall also be entered when using one of the Receive Mode synopsis forms and a reply or new message is composed using the **reply**, **Reply**, **followup**, **Followup**, or **mail** commands and standard input is a terminal. When the message is typed and the end of the message is encountered, the message shall be passed to the mail delivery software. Commands can be entered by beginning a line with the escape character (by default, <tilde> (' ~ ')) followed by a single command letter and optional arguments. See [Commands in mailx](#) (on page 2918) for a summary of these commands. It is unspecified what effect these commands will have if standard input is not a terminal when a message is entered using either the Send Mode synopsis, or the Read Mode commands **reply**, **Reply**, **followup**, **Followup**, or **mail**.

**Note:** For notational convenience, this section uses the default escape character, <tilde>, in all references and examples.

At any time, the behavior of *mailx* shall be governed by a set of environmental and internal variables. These are flags and valued parameters that can be set and cleared via the *mailx* **set** and **unset** commands.

Regular commands are of the form:

```
[command] [msglist] [argument ...]
```

If no *command* is specified in command mode, **next** shall be assumed. In input mode, commands shall be recognized by the escape character, and lines not treated as commands shall be taken as input for the message.

In command mode, each message shall be assigned a sequential number, starting with 1.

All messages have a state that shall affect how they are displayed in the header summary and how they are retained or deleted upon termination of *mailx*. There is at any time the notion of a *current* message, which shall be marked by a '>' at the beginning of a line in the header summary. When *mailx* is invoked using one of the Receive Mode synopsis forms, the current message shall be the first new message, if there is a new message, or the first unread message if there is an unread message, or the first message if there are any messages, or unspecified if there are no messages in the mailbox. Each command that takes an optional list of messages (*msglist*) or an optional single message (*message*) on which to operate shall leave the current message set to the highest-numbered message of the messages specified, unless the command deletes messages, in which case the current message shall be set to the first undeleted message (that is, a message not in the deleted state) after the highest-numbered message deleted by the command, if one exists, or the first undeleted message before the highest-numbered message deleted by the command, if one exists, or to an unspecified value if there are no remaining undeleted messages. All messages shall be in one of the following states:

|       |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|-------|------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 96214 | <i>new</i>       | The message is present in the system mailbox and has not been viewed by the user or moved to any other state. Messages in state <i>new</i> when <i>mailx</i> quits shall be retained in the system mailbox.                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 96215 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 96216 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 96217 | <i>unread</i>    | The message has been present in the system mailbox for more than one invocation of <i>mailx</i> and has not been viewed by the user or moved to any other state. Messages in state <i>unread</i> when <i>mailx</i> quits shall be retained in the system mailbox.                                                                                                                                                                                                                                                                                                                                                                                                          |
| 96218 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 96219 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 96220 | <i>read</i>      | The message has been processed by one of the following commands: <b>~f, ~m, ~F, ~M, copy, mbox, next, pipe, print, Print, top, type, Type, undelete</b> . The <b>delete, dp,</b> and <b>dt</b> commands may also cause the next message to be marked as <i>read</i> , depending on the value of the <b>autoprint</b> variable. Messages that are in the system mailbox and in state <i>read</i> when <i>mailx</i> quits shall be saved in the <b>mbox</b> , unless the internal variable <b>hold</b> was set. Messages that are in the <b>mbox</b> or in a secondary mailbox and in state <i>read</i> when <i>mailx</i> quits shall be retained in their current location. |
| 96221 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 96222 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 96223 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 96224 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 96225 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 96226 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 96227 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 96228 | <i>deleted</i>   | The message has been processed by one of the following commands: <b>delete, dp, dt</b> . Messages in state <i>deleted</i> when <i>mailx</i> quits shall be deleted. Deleted messages shall be ignored until <i>mailx</i> quits or changes mailboxes or they are specified to the undelete command; for example, the message specification <i>/string</i> shall only search the subject lines of messages that have not yet been deleted, unless the command operating on the list of messages is <b>undelete</b> . No deleted message or deleted message header shall be displayed by any <i>mailx</i> command other than <b>undelete</b> .                                |
| 96229 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 96230 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 96231 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 96232 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 96233 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 96234 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 96235 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 96236 | <i>preserved</i> | The message has been processed by a <b>preserve</b> command. When <i>mailx</i> quits, the message shall be retained in its current location.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| 96237 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 96238 | <i>saved</i>     | The message has been processed by one of the following commands: <b>save</b> or <b>write</b> . If the current mailbox is the system mailbox, and the internal variable <b>keepsave</b> is set, messages in the state <i>saved</i> shall be saved to the file designated by the <i>MBOX</i> variable (see the ENVIRONMENT VARIABLES section). If the current mailbox is the system mailbox, messages in the state <i>saved</i> shall be deleted from the current mailbox, when the <b>quit</b> or <b>file</b> command is used to exit the current mailbox.                                                                                                                  |
| 96239 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 96240 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 96241 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 96242 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 96243 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 96244 |                  | The header-summary line for each message shall indicate the state of the message.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 96245 |                  | Many commands take an optional list of messages ( <i>msglist</i> ) on which to operate, which defaults to the current message. A <i>msglist</i> is a list of message specifications separated by <blank> characters, which can include:                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 96246 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 96247 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 96248 | <i>n</i>         | Message number <i>n</i> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 96249 | <i>+</i>         | The next undeleted message, or the next deleted message for the <b>undelete</b> command.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 96250 | <i>–</i>         | The next previous undeleted message, or the next previous deleted message for the <b>undelete</b> command.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| 96251 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 96252 | <i>.</i>         | The current message.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 96253 | <i>^</i>         | The first undeleted message, or the first deleted message for the <b>undelete</b> command.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| 96254 | <i>\$</i>        | The last message.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 96255 | <i>*</i>         | All messages.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |



- 96256        *n-m*        An inclusive range of message numbers.
- 96257        *address*    All messages from *address*; any address as shown in a header summary shall be  
96258                   matchable in this form.
- 96259        */string*    All messages with *string* in the subject line (case ignored).
- 96260        *: c*        All messages of type *c*, where *c* shall be one of:
- 96261                *d*       Deleted messages.
- 96262                *n*       New messages.
- 96263                *o*       Old messages (any not in state *read* or *new*).
- 96264                *r*       Read messages.
- 96265                *u*       Unread messages.

96266        Other commands take an optional message (*message*) on which to operate, which defaults to the  
96267                   current message. All of the forms allowed for *msglist* are also allowed for *message*, but if more  
96268                   than one message is specified, only the first shall be operated on.

96269        Other arguments are usually arbitrary strings whose usage depends on the command involved.

## 96270        **Start-Up in mailx**

96271        At start-up time, *mailx* shall take the following steps in sequence:

- 96272                1. Establish all variables at their stated default values.
- 96273                2. Process command line options, overriding corresponding default values.
- 96274                3. Import any of the *DEAD*, *EDITOR*, *MBOX*, *LISTER*, *PAGER*, *SHELL*, or *VISUAL* variables  
96275                   that are present in the environment, overriding the corresponding default values.
- 96276                4. Read *mailx* commands from an unspecified system start-up file, unless the *-n* option is  
96277                   given, to initialize any internal *mailx* variables and aliases.
- 96278                5. Process the start-up file of *mailx* commands named in the user *MAILRC* variable.

96279        Most regular *mailx* commands are valid inside start-up files, the most common use being to set  
96280                   up initial display options and alias lists. The following commands shall be invalid in the start-up  
96281                   file: **!**, **edit**, **hold**, **mail**, **preserve**, **reply**, **Reply**, **shell**, **visual**, **Copy**, **followup**, and **Followup**.  
96282                   Any errors in the start-up file shall either cause *mailx* to terminate with a diagnostic message and  
96283                   a non-zero status or to continue after writing a diagnostic message, ignoring the remainder of  
96284                   the lines in the start-up file.

96285        A blank line in a start-up file shall be ignored.

## 96286        **Internal Variables in mailx**

96287        The following variables are internal *mailx* variables. Each internal variable can be set via the  
96288                   *mailx set* command at any time. The **unset** and **set no name** commands can be used to erase  
96289                   variables.

96290        In the following list, variables shown as:

96291        *variable*

96292        represent Boolean values. Variables shown as:

96293        *variable=value*

|       |                         |                                                                                                               |
|-------|-------------------------|---------------------------------------------------------------------------------------------------------------|
| 96294 |                         | shall be assigned string or numeric values. For string values, the rules in <a href="#">Commands in mailx</a> |
| 96295 |                         | (on page 2918) concerning filenames and quoting shall also apply.                                             |
| 96296 |                         | The defaults specified here may be changed by the unspecified system start-up file unless the                 |
| 96297 |                         | user specifies the <b>-n</b> option.                                                                          |
| 96298 | <b>allnet</b>           | All network names whose login name components match shall be treated as                                       |
| 96299 |                         | identical. This shall cause the <i>msglist</i> message specifications to behave similarly.                    |
| 96300 |                         | The default shall be <b>noallnet</b> . See also the <b>alternates</b> command and the <b>metoo</b>            |
| 96301 |                         | variable.                                                                                                     |
| 96302 | <b>append</b>           | Append messages to the end of the <b>mbox</b> file upon termination instead of placing                        |
| 96303 |                         | them at the beginning. The default shall be <b>noappend</b> . This variable shall not                         |
| 96304 |                         | affect the <b>save</b> command when saving to <b>mbox</b> .                                                   |
| 96305 | <b>ask, asksub</b>      | Prompt for a subject line on outgoing mail if one is not specified on the command                             |
| 96306 |                         | line with the <b>-s</b> option. The <b>ask</b> and <b>asksub</b> forms are synonyms; the system shall         |
| 96307 |                         | refer to <b>asksub</b> and <b>noasksub</b> in its messages, but shall accept <b>ask</b> and <b>noask</b> as   |
| 96308 |                         | user input to mean <b>asksub</b> and <b>noasksub</b> . It shall not be possible to set both <b>ask</b>        |
| 96309 |                         | and <b>noasksub</b> , or <b>noask</b> and <b>asksub</b> . The default shall be <b>asksub</b> , but no         |
| 96310 |                         | prompting shall be done if standard input is not a terminal.                                                  |
| 96311 | <b>askbcc</b>           | Prompt for the blind copy list. The default shall be <b>noaskbcc</b> .                                        |
| 96312 | <b>askcc</b>            | Prompt for the copy list. The default shall be <b>noaskcc</b> .                                               |
| 96313 | <b>autoprint</b>        | Enable automatic writing of messages after <b>delete</b> and <b>undelete</b> commands. The                    |
| 96314 |                         | default shall be <b>noautoprint</b> .                                                                         |
| 96315 | <b>bang</b>             | Enable the special-case treatment of <exclamation-mark> characters ('!') in                                   |
| 96316 |                         | escape command lines; see the <b>escape</b> command and <a href="#">Command Escapes in mailx</a>              |
| 96317 |                         | (on page 2927). The default shall be <b>nobang</b> , disabling the expansion of '!' in the                    |
| 96318 |                         | <i>command</i> argument to the <b>~!</b> command and the <b>~!command</b> escape.                             |
| 96319 | <b>cmd=command</b>      |                                                                                                               |
| 96320 |                         | Set the default command to be invoked by the <b>pipe</b> command. The default shall be                        |
| 96321 |                         | <b>nocmd</b> .                                                                                                |
| 96322 | <b>crt=number</b>       | Pipe messages having more than <i>number</i> lines through the command specified by                           |
| 96323 |                         | the value of the <i>PAGER</i> variable. The default shall be <b>nocrt</b> . If it is set to null, the         |
| 96324 |                         | value used is implementation-defined.                                                                         |
| 96325 | XSI <b>debug</b>        | Enable verbose diagnostics for debugging. Messages are not delivered. The                                     |
| 96326 |                         | default shall be <b>nodebug</b> .                                                                             |
| 96327 | <b>dot</b>              | When <b>dot</b> is set, a <period> on a line by itself during message input from a                            |
| 96328 |                         | terminal shall also signify end-of-file (in addition to normal end-of-file). The                              |
| 96329 |                         | default shall be <b>nodot</b> . If <b>ignoreeof</b> is set (see below), a setting of <b>nodot</b> shall be    |
| 96330 |                         | ignored and the <period> is the only method to terminate input mode.                                          |
| 96331 | <b>escape=c</b>         | Set the command escape character to be the character 'c'. By default, the                                     |
| 96332 |                         | command escape character shall be <tilde>. If <b>escape</b> is unset, <tilde> shall be                        |
| 96333 |                         | used; if it is set to null, command escaping shall be disabled.                                               |
| 96334 | <b>flipr</b>            | Reverse the meanings of the <b>R</b> and <b>r</b> commands. The default shall be <b>noflipr</b> .             |
| 96335 | <b>folder=directory</b> |                                                                                                               |
| 96336 |                         | The default directory for saving mail files. User-specified filenames beginning with                          |
| 96337 |                         | a <plus-sign> ('+') shall be expanded by preceding the filename with this                                     |

|       |                            |                                                                                                          |
|-------|----------------------------|----------------------------------------------------------------------------------------------------------|
| 96338 |                            | directory name to obtain the real pathname. If <i>directory</i> does not start with a                    |
| 96339 |                            | <slash> (' / '), the contents of <i>HOME</i> shall be prefixed to it. The default shall be               |
| 96340 |                            | <b>nofolder</b> . If <b>folder</b> is unset or set to null, user-specified filenames beginning with      |
| 96341 |                            | '+' shall refer to files in the current directory that begin with the literal '+'                        |
| 96342 |                            | character. See also <b>outfolder</b> below. The <b>folder</b> value need not affect the processing       |
| 96343 |                            | of the files named in <i>MBOX</i> and <i>DEAD</i> .                                                      |
| 96344 | <b>header</b>              | Enable writing of the header summary when entering <i>mailx</i> in Receive Mode. The                     |
| 96345 |                            | default shall be <b>header</b> .                                                                         |
| 96346 | <b>hold</b>                | Preserve all messages that are read in the system mailbox instead of putting them                        |
| 96347 |                            | in the <b>mbox</b> save file. The default shall be <b>nohold</b> .                                       |
| 96348 | <b>ignore</b>              | Ignore interrupts while entering messages. The default shall be <b>noignore</b> .                        |
| 96349 | <b>ignoreeof</b>           | Ignore normal end-of-file during message input. Input can be terminated only by                          |
| 96350 |                            | entering a <period> (' . ') on a line by itself or by the ~. command escape. The                         |
| 96351 |                            | default shall be <b>noignoreeof</b> . See also <b>dot</b> above.                                         |
| 96352 | <b>indentprefix=string</b> |                                                                                                          |
| 96353 |                            | A string that shall be added as a prefix to each line that is inserted into the message                  |
| 96354 |                            | by the ~m command escape. This variable shall default to one <tab>.                                      |
| 96355 | <b>keep</b>                | When a system mailbox, secondary mailbox, or <b>mbox</b> is empty, truncate it to zero                   |
| 96356 |                            | length instead of removing it. The default shall be <b>nokeep</b> .                                      |
| 96357 | <b>keepsave</b>            | Keep the messages that have been saved from the system mailbox into other files                          |
| 96358 |                            | in the file designated by the variable <i>MBOX</i> , instead of deleting them. The default               |
| 96359 |                            | shall be <b>nokeepsave</b> .                                                                             |
| 96360 | <b>metoo</b>               | Suppress the deletion of the login name of the user from the recipient list when                         |
| 96361 |                            | replying to a message or sending to a group. The default shall be <b>nometoo</b> .                       |
| 96362 | XSI <b>onehop</b>          | When responding to a message that was originally sent to several recipients, the                         |
| 96363 |                            | other recipient addresses are normally forced to be relative to the originating                          |
| 96364 |                            | author's machine for the response. This flag disables alteration of the recipients'                      |
| 96365 |                            | addresses, improving efficiency in a network where all machines can send directly                        |
| 96366 |                            | to all other machines (that is, one hop away). The default shall be <b>noonehop</b> .                    |
| 96367 | <b>outfolder</b>           | Cause the files used to record outgoing messages to be located in the directory                          |
| 96368 |                            | specified by the <b>folder</b> variable unless the pathname is absolute. The default shall               |
| 96369 |                            | be <b>nooutfolder</b> . See the <b>record</b> variable.                                                  |
| 96370 | <b>page</b>                | Insert a <form-feed> after each message sent through the pipe created by the <b>pipe</b>                 |
| 96371 |                            | command. The default shall be <b>nopage</b> .                                                            |
| 96372 | <b>prompt=string</b>       |                                                                                                          |
| 96373 |                            | Set the command-mode prompt to <i>string</i> . If <i>string</i> is null or if <b>noprompt</b> is set, no |
| 96374 |                            | prompting shall occur. The default shall be to prompt with the string "? ".                              |
| 96375 | <b>quiet</b>               | Refrain from writing the opening message and version when entering <i>mailx</i> . The                    |
| 96376 |                            | default shall be <b>noquiet</b> .                                                                        |
| 96377 | <b>record=file</b>         | Record all outgoing mail in the file with the pathname <i>file</i> . The default shall be                |
| 96378 |                            | <b>norecord</b> . See also <b>outfolder</b> above.                                                       |
| 96379 | <b>save</b>                | Enable saving of messages in the dead-letter file on interrupt or delivery error. See                    |
| 96380 |                            | the variable <i>DEAD</i> for the location of the dead-letter file. The default shall be <b>save</b> .    |

96381 **screen=number**  
 96382 Set the number of lines in a screenful of headers for the **headers** and **z** commands.  
 96383 If **screen** is not specified, a value based on the terminal type identified by the  
 96384 *TERM* environment variable, the window size, the baud rate, or some combination  
 96385 of these shall be used.

96386 **sendwait** Wait for the background mailer to finish before returning. The default shall be  
 96387 **nosendwait**.

96388 **showto** When the sender of the message was the user who is invoking *mailx*, write the  
 96389 information from the **To:** line instead of the **From:** line in the header summary. The  
 96390 default shall be **noshowto**.

96391 **sign=string** Set the variable inserted into the text of a message when the **~a** command escape is  
 96392 given. The default shall be **nosign**. The character sequences **'\t'** and **'\n'** shall  
 96393 be recognized in the variable as <tab> and <newline> characters, respectively. (See  
 96394 also **~i** in [Command Escapes in mailx](#) (on page 2927).)

96395 **Sign=string** Set the variable inserted into the text of a message when the **~A** command escape is  
 96396 given. The default shall be **noSign**. The character sequences **'\t'** and **'\n'** shall  
 96397 be recognized in the variable as <tab> and <newline> characters, respectively.

96398 **toplines=number**  
 96399 Set the number of lines of the message to write with the **top** command. The default  
 96400 shall be 5.

## 96401 Commands in mailx

96402 The following *mailx* commands shall be provided. In the following list, header refers to lines  
 96403 from the message header, as shown in the OUTPUT FILES section. Header-line refers to lines  
 96404 within the header that begin with one or more non-white-space characters, immediately  
 96405 followed by a <colon> and white space and continuing until the next line beginning with a non-  
 96406 white-space character or an empty line. Header-field refers to the portion of a header line prior  
 96407 to the first <colon> in that line.

96408 For each of the commands listed below, the command can be entered as the abbreviation (those  
 96409 characters in the Synopsis command word preceding the **' ['**), the full command (all characters  
 96410 shown for the command word, omitting the **' ['** and **' ]'**), or any truncation of the full  
 96411 command down to the abbreviation. For example, the **exit** command (shown as **ex[it]** in the  
 96412 Synopsis) can be entered as **ex**, **exi**, or **exit**.

96413 The arguments to commands can be quoted, using the following methods:

- 96414 • An argument can be enclosed between paired double-quotes (**" "**) or single-quotes (**' '**);  
 96415 any white space, shell word expansion, or <backslash> characters within the quotes shall  
 96416 be treated literally as part of the argument. A double-quote shall be treated literally within  
 96417 single-quotes and *vice versa*. These special properties of the <quotation-mark> characters  
 96418 shall occur only when they are paired at the beginning and end of the argument.
- 96419 • A <backslash> outside of the enclosing quotes shall be discarded and the following  
 96420 character treated literally as part of the argument.
- 96421 • An unquoted <backslash> at the end of a command line shall be discarded and the next  
 96422 line shall continue the command.

96423 Filenames, where expected, shall be subjected to the following transformations, in sequence:

- 96424 • If the filename begins with an unquoted <plus-sign>, and the **folder** variable is defined  
96425 (see the **folder** variable), the <plus-sign> shall be replaced by the value of the **folder**  
96426 variable followed by a <slash>. If the **folder** variable is unset or is set to null, the filename  
96427 shall be unchanged.
- 96428 • Shell word expansions shall be applied to the filename (see [Section 2.6](#), on page 2327). If  
96429 more than a single pathname results from this expansion and the command is expecting  
96430 one file, the effects are unspecified.

## 96431 Declare Aliases

96432 *Synopsis:*     a[lias] [alias [address...]]  
96433               g[roup] [alias [address...]]

96434 Add the given addresses to the alias specified by *alias*. The names shall be substituted when  
96435 *alias* is used as a recipient address specified by the user in an outgoing message (that is, other  
96436 recipients addressed indirectly through the **reply** command shall not be substituted in this  
96437 manner). Mail address alias substitution shall apply only when the alias string is used as a full  
96438 address; for example, when **hlj** is an alias, *hlj@posix.com* does not trigger the alias substitution. If  
96439 no arguments are given, write a listing of the current aliases to standard output. If only an *alias*  
96440 argument is given, write a listing of the specified alias to standard output. These listings need  
96441 not reflect the same order of addresses that were entered.

## 96442 Declare Alternatives

96443 *Synopsis:*     alt[ernates] name...

96444 (See also the **metoo** variable.) Declare a list of alternative names for the user's login. When  
96445 responding to a message, these names shall be removed from the list of recipients for the  
96446 response. The comparison of names shall be in a case-insensitive manner. With no arguments,  
96447 **alternates** shall write the current list of alternative names.

## 96448 Change Current Directory

96449 *Synopsis:*     cd [directory]  
96450               ch[dir] [directory]

96451 Change directory. If *directory* is not specified, the contents of *HOME* shall be used.

## 96452 Copy Messages

96453 *Synopsis:*     c[opy] [file]  
96454               c[opy] [msglist] file  
96455               C[opy] [msglist]

96456 Copy messages to the file named by the pathname *file* without marking the messages as saved.  
96457 Otherwise, it shall be equivalent to the **save** command.

96458 In the capitalized form, save the specified messages in a file whose name is derived from the  
96459 author of the message to be saved, without marking the messages as saved. Otherwise, it shall  
96460 be equivalent to the **Save** command.

## Delete Messages

*Synopsis:*     d[delete] [msglist]

Mark messages for deletion from the mailbox. The deletions shall not occur until *mailx* quits (see the **quit** command) or changes mailboxes (see the **folder** command). If **autoprint** is set and there are messages remaining after the **delete** command, the current message shall be written as described for the **print** command (see the **print** command); otherwise, the *mailx* prompt shall be written.

## Discard Header Fields

*Synopsis:*     di[scard] [header-field...]  
              ig[nore] [header-field...]

Suppress the specified header fields when writing messages. Specified *header-fields* shall be added to the list of suppressed header fields. Examples of header fields to ignore are **status** and **cc**. The fields shall be included when the message is saved. The **Print** and **Type** commands shall override this command. The comparison of header fields shall be in a case-insensitive manner. If no arguments are specified, write a list of the currently suppressed header fields to standard output; the listing need not reflect the same order of header fields that were entered.

If both **retain** and **discard** commands are given, **discard** commands shall be ignored.

## Delete Messages and Display

*Synopsis:*     dp [msglist]  
              dt [msglist]

Delete the specified messages as described for the **delete** command, except that the **autoprint** variable shall have no effect, and the current message shall be written only if it was set to a message after the last message deleted by the command. Otherwise, an informational message to the effect that there are no further messages in the mailbox shall be written, followed by the *mailx* prompt.

## Echo a String

*Synopsis:*     ec[ho] string ...

Echo the given strings, equivalent to the shell *echo* utility.

## Edit Messages

*Synopsis:*     e[dit] [msglist]

Edit the given messages. The messages shall be placed in a temporary file and the utility named by the *EDITOR* variable is invoked to edit each file in sequence. The default *EDITOR* is unspecified.

The **edit** command does not modify the contents of those messages in the mailbox.

**Exit**

*Synopsis:*     ex[it]  
                  x[it]

Exit from *mailx* without changing the mailbox. No messages shall be saved in the **mbox** (see also **quit**).

**Change Folder**

*Synopsis:*     fi[le] [*file*]  
                 fold[er] [*file*]

Quit (see the **quit** command) from the current file of messages and read in the file named by the pathname *file*. If no argument is given, the name and status of the current mailbox shall be written.

Several unquoted special characters shall be recognized when used as *file* names, with the following substitutions:

- %           The system mailbox for the invoking user.
  - %*user*     The system mailbox for *user*.
  - #           The previous file.
  - &           The current **mbox**.
  - +*file*     The named file in the **folder** directory. (See the **folder** variable.)
- The default file shall be the current mailbox.

**Display List of Folders**

*Synopsis:*     folders

Write the names of the files in the directory set by the **folder** variable. The command specified by the *LISTER* environment variable shall be used (see the ENVIRONMENT VARIABLES section).

**Follow Up Specified Messages**

*Synopsis:*     fo[llowup] [*message*]  
                 F[ollowup] [*msglist*]

In the lowercase form, respond to a message, recording the response in a file whose name is derived from the author of the message. See also the **save** and **copy** commands and **outfolder**.

In the capitalized form, respond to the first message in the *msglist*, sending the message to the author of each message in the *msglist*. The subject line shall be taken from the first message and the response shall be recorded in a file whose name is derived from the author of the first message. See also the **Save** and **Copy** commands and **outfolder**.

Both forms shall override the **record** variable, if set.

**Display Header Summary for Specified Messages**

*Synopsis:*     f[rom] [msglist]

Write the header summary for the specified messages.

**Display Header Summary**

*Synopsis:*     h[eaders] [message]

Write the page of headers that includes the message specified. If the *message* argument is not specified, the current message shall not change. However, if the *message* argument is specified, the current message shall become the message that appears at the top of the page of headers that includes the message specified. The **screen** variable sets the number of headers per page. See also the **z** command.

**Help**

*Synopsis:*     hel[p]  
                 ?

Write a summary of commands.

**Hold Messages**

*Synopsis:*     ho[ld] [msglist]  
                 pre[serve] [msglist]

Mark the messages in *msglist* to be retained in the mailbox when *mailx* terminates. This shall override any commands that might previously have marked the messages to be deleted. During the current invocation of *mailx*, only the **delete**, **dp**, or **dt** commands shall remove the *preserve* marking of a message.

**Execute Commands Conditionally**

*Synopsis:*     i[f] s|r  
                 mail-commands  
                 el[se]  
                 mail-commands  
                 en[dif]

Execute commands conditionally, where **if s** executes the following *mail-commands*, up to an **else** or **endif**, if the program is in Send Mode, and **if r** shall cause the *mail-commands* to be executed only in Receive Mode.

**List Available Commands**

*Synopsis:*     l[ist]

Write a list of all commands available. No explanation shall be given.



**Mail a Message**

*Synopsis:*    m[ail] address...

Mail a message to the specified addresses or aliases.

**Direct Messages to mbox**

*Synopsis:*    mb[ox] [msglist]

Arrange for the given messages to end up in the **mbox** save file when *mailx* terminates normally. See **MBOX**. See also the **exit** and **quit** commands.

**Process Next Specified Message**

*Synopsis:*    n[ext] [message]

If the current message has not been written (for example, by the **print** command) since *mailx* started or since any other message was the current message, behave as if the **print** command was entered. Otherwise, if there is an undeleted message after the current message, make it the current message and behave as if the **print** command was entered. Otherwise, an informational message to the effect that there are no further messages in the mailbox shall be written, followed by the *mailx* prompt. Should the current message location be the result of an immediately preceding **hold**, **mbox**, **preserve**, or **touch** command, **next** will act as if the current message has already been written.

**Pipe Message**

*Synopsis:*    pi[pe] [[msglist] command]  
              | [[msglist] command]

Pipe the messages through the given *command* by invoking the command interpreter specified by *SHELL* with two arguments: **-c** and *command*. (See also *sh -c*.) The application shall ensure that the command is given as a single argument. Quoting, described previously, can be used to accomplish this. If no arguments are given, the current message shall be piped through the command specified by the value of the **cmd** variable. If the **page** variable is set, a <form-feed> shall be inserted after each message.

**Display Message with Headers**

*Synopsis:*    P[rint] [msglist]  
              T[ype] [msglist]

Write the specified messages, including all header lines, to standard output. Override suppression of lines by the **discard**, **ignore**, and **retain** commands. If **crt** is set, the messages longer than the number of lines specified by the **crt** variable shall be paged through the command specified by the *PAGER* environment variable.

**Display Message**

*Synopsis:*    p[rint] [msglist]  
              t[ype] [msglist]

Write the specified messages to standard output. If **crt** is set, the messages longer than the number of lines specified by the **crt** variable shall be paged through the command specified by the *PAGER* environment variable.

**Quit**

*Synopsis:*     q[uit]  
                  end-of-file

Terminate *mailx*, storing messages that were read in **mbox** (if the current mailbox is the system mailbox and unless **hold** is set), deleting messages that have been explicitly saved (unless **keepsave** is set), discarding messages that have been deleted, and saving all remaining messages in the mailbox.

**Reply to a Message List**

*Synopsis:*     R[eply] [msglist]  
                  R[espond] [msglist]

Mail a reply message to the sender of each message in the *msglist*. The subject line shall be formed by concatenating **Re:**<space> (unless it already begins with that string) and the subject from the first message. If **record** is set to a filename, the response shall be saved at the end of that file.

See also the **flipr** variable.

**Reply to a Message**

*Synopsis:*     r[eply] [message]  
                  r[espond] [message]

Mail a reply message to all recipients included in the header of the message. The subject line shall be formed by concatenating **Re:**<space> (unless it already begins with that string) and the subject from the message. If **record** is set to a filename, the response shall be saved at the end of that file.

See also the **flipr** variable.

**Retain Header Fields**

*Synopsis:*     ret[ain] [header-field...]

Retain the specified header fields when writing messages. This command shall override all **discard** and **ignore** commands. The comparison of header fields shall be in a case-insensitive manner. If no arguments are specified, write a list of the currently retained header fields to standard output; the listing need not reflect the same order of header fields that were entered.

**Save Messages**

*Synopsis:*     s[ave] [file]  
                  s[ave] [msglist] file  
                  S[ave] [msglist]

Save the specified messages in the file named by the pathname *file*, or the **mbox** if the *file* argument is omitted. The file shall be created if it does not exist; otherwise, the messages shall be appended to the file. The message shall be put in the state *saved*, and shall behave as specified in the description of the *saved* state when the current mailbox is exited by the **quit** or **file** command.

In the capitalized form, save the specified messages in a file whose name is derived from the author of the first message. The name of the file shall be taken to be the author's name with all network addressing stripped off. See also the **Copy**, **followup**, and **Followup** commands and

96642        **outfolder** variable.

### 96643        **Set Variables**

96644        *Synopsis:*     **se**[t] [*name*=[*string*]] ...] [*name=number* ...] [*noname* ...]

96645        Define one or more variables called *name*. The variable can be given a null, string, or numeric value. Quoting and <backslash>-escapes can occur anywhere in *string*, as described previously, as if the *string* portion of the argument were the entire argument. The forms *name* and *name=* shall be equivalent to *name=""* for variables that take string values. The **set** command without arguments shall write a list of all defined variables and their values. The **no name** form shall be equivalent to **unset name**.

### 96651        **Invoke a Shell**

96652        *Synopsis:*     **sh**[ell]

96653        Invoke an interactive command interpreter (see also *SHELL*).

### 96654        **Display Message Size**

96655        *Synopsis:*     **si**[ze] [*msglist*]

96656        Write the size in bytes of each of the specified messages.

### 96657        **Read mailx Commands From a File**

96658        *Synopsis:*     **so**[urce] *file*

96659        Read and execute commands from the file named by the pathname *file* and return to command mode.

### 96661        **Display Beginning of Messages**

96662        *Synopsis:*     **to**[p] [*msglist*]

96663        Write the top few lines of each of the specified messages. If the **toplines** variable is set, it is taken as the number of lines to write. The default shall be 5.

### 96665        **Touch Messages**

96666        *Synopsis:*     **tou**[ch] [*msglist*]

96667        Touch the specified messages. If any message in *msglist* is not specifically deleted nor saved in a file, it shall be placed in the **mbox** upon normal termination. See **exit** and **quit**.

### 96669        **Delete Aliases**

96670        *Synopsis:*     **una**[lias] [*alias*] ...

96671        Delete the specified alias names. If a specified alias does not exist, the results are unspecified.

**Undelete Messages**

*Synopsis:*     u[ndelete] [*msglist*]

Change the state of the specified messages from deleted to read. If **autoprint** is set, the last message of those restored shall be written. If *msglist* is not specified, the message shall be selected as follows:

- If there are any deleted messages that follow the current message, the first of these shall be chosen.
- Otherwise, the last deleted message that also precedes the current message shall be chosen.

**Unset Variables**

*Synopsis:*     uns[et] *name...*

Cause the specified variables to be erased.

**Edit Message with Full-Screen Editor**

*Synopsis:*     v[isual] [*msglist*]

Edit the given messages with a screen editor. Each message shall be placed in a temporary file, and the utility named by the *VISUAL* variable shall be invoked to edit each file in sequence. The default editor shall be *vi*.

The **visual** command does not modify the contents of those messages in the mailbox.

**Write Messages to a File**

*Synopsis:*     w[rite] [*msglist*] *file*

Write the given messages to the file specified by the pathname *file*, minus the message header. Otherwise, it shall be equivalent to the **save** command.

**Scroll Header Display**

*Synopsis:*     z[+|-]

Scroll the header display forward (if '+' is specified or if no option is specified) or backward (if '-' is specified) one screenful. The number of headers written shall be set by the **screen** variable.

**Invoke Shell Command**

*Synopsis:*     !*command*

Invoke the command interpreter specified by *SHELL* with two arguments: **-c** and *command*. (See also *sh -c*.) If the **bang** variable is set, each unescaped occurrence of '!' in *command* shall be replaced with the command executed by the previous ! command or ~! command escape.

**Null Command**

*Synopsis:*     # *comment*

This null command (comment) shall be ignored by *mailx*.

**Display Current Message Number**

*Synopsis:*     =

Write the current message number.

**Command Escapes in mailx**

The following commands can be entered only from input mode, by beginning a line with the escape character (by default, <tilde> (' ~')). See the **escape** variable description for changing this special character. The format for the commands shall be:

<escape-character><command-char><separator> [<arguments>]

where the <separator> can be zero or more <blank> characters.

In the following descriptions, the application shall ensure that the argument *command* (but not *mailx-command*) is a shell command string. Any string acceptable to the command interpreter specified by the *SHELL* variable when it is invoked as *SHELL -c command\_string* shall be valid. The command can be presented as multiple arguments (that is, quoting is not required).

Command escapes that are listed with *msglist* or *mailx-command* arguments are invalid in Send Mode and produce unspecified results.

**~! *command***   Invoke the command interpreter specified by *SHELL* with two arguments: **-c** and *command*; and then return to input mode. If the **bang** variable is set, each unescaped occurrence of '!' in *command* shall be replaced with the command executed by the previous ! command or ~! command escape.

**~.**             Simulate end-of-file (terminate message input).

**~: *mailx-command*, ~\_ *mailx-command***  
Perform the command-level request.

**~?**             Write a summary of command escapes.

**~A**             This shall be equivalent to **~i Sign**.

**~a**             This shall be equivalent to **~i sign**.

**~b *name...***   Add the *names* to the blind carbon copy (**Bcc**) list.

**~c *name...***   Add the *names* to the carbon copy (**Cc**) list.

**~d**             Read in the dead-letter file. See *DEAD* for a description of this file.

**~e**             Invoke the editor, as specified by the *EDITOR* environment variable, on the partial message.

**~f [*msglist*]**   Forward the specified messages. The specified messages shall be inserted into the current message without alteration. This command escape also shall insert message headers into the message with field selection affected by the **discard**, **ignore**, and **retain** commands.

|       |                                                                                                             |                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|-------|-------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 96740 | <b>~F</b> [ <i>msglist</i> ]                                                                                | This shall be the equivalent of the <b>~f</b> command escape, except that all headers shall be included in the message, regardless of previous <b>discard</b> , <b>ignore</b> , and <b>retain</b> commands.                                                                                                                                                                                                                                |
| 96741 |                                                                                                             |                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 96742 |                                                                                                             |                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 96743 | <b>~h</b>                                                                                                   | If standard input is a terminal, prompt for a <b>Subject</b> line and the <b>To</b> , <b>Cc</b> , and <b>Bcc</b> lists. Other implementation-defined headers may also be presented for editing. If the field is written with an initial value, it can be edited as if it had just been typed.                                                                                                                                              |
| 96744 |                                                                                                             |                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 96745 |                                                                                                             |                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 96746 | <b>~i</b> <i>string</i>                                                                                     | Insert the value of the named variable, followed by a <newline>, into the text of the message. If the string is unset or null, the message shall not be changed.                                                                                                                                                                                                                                                                           |
| 96747 |                                                                                                             |                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 96748 | <b>~m</b> [ <i>msglist</i> ]                                                                                | Insert the specified messages into the message, prefixing non-empty lines with the string in the <b>indentprefix</b> variable. This command escape also shall insert message headers into the message, with field selection affected by the <b>discard</b> , <b>ignore</b> , and <b>retain</b> commands.                                                                                                                                   |
| 96749 |                                                                                                             |                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 96750 |                                                                                                             |                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 96751 |                                                                                                             |                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 96752 | <b>~M</b> [ <i>msglist</i> ]                                                                                | This shall be the equivalent of the <b>~m</b> command escape, except that all headers shall be included in the message, regardless of previous <b>discard</b> , <b>ignore</b> , and <b>retain</b> commands.                                                                                                                                                                                                                                |
| 96753 |                                                                                                             |                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 96754 |                                                                                                             |                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 96755 | <b>~p</b>                                                                                                   | Write the message being entered. If the message is longer than <b>crt</b> lines (see <a href="#">Internal Variables in mailx</a> , on page 2915), the output shall be paginated as described for the <b>PAGER</b> variable.                                                                                                                                                                                                                |
| 96756 |                                                                                                             |                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 96757 |                                                                                                             |                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 96758 | <b>~q</b>                                                                                                   | Quit (see the <b>quit</b> command) from input mode by simulating an interrupt. If the body of the message is not empty, the partial message shall be saved in the dead-letter file. See <i>DEAD</i> for a description of this file.                                                                                                                                                                                                        |
| 96759 |                                                                                                             |                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 96760 |                                                                                                             |                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 96761 | <b>~r</b> <i>file</i> , <b>~&lt;</b> <i>file</i> , <b>~r</b> <i>!command</i> , <b>~&lt;</b> <i>!command</i> | Read in the file specified by the pathname <i>file</i> . If the argument begins with an <exclamation-mark> ('!'), the rest of the string shall be taken as an arbitrary system command; the command interpreter specified by <i>SHELL</i> shall be invoked with two arguments: <b>-c</b> and <i>command</i> . The standard output of <i>command</i> shall be inserted into the message.                                                    |
| 96762 |                                                                                                             |                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 96763 |                                                                                                             |                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 96764 |                                                                                                             |                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 96765 |                                                                                                             |                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 96766 |                                                                                                             |                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 96767 | <b>~s</b> <i>string</i>                                                                                     | Set the subject line to <i>string</i> .                                                                                                                                                                                                                                                                                                                                                                                                    |
| 96768 | <b>~t</b> <i>name...</i>                                                                                    | Add the given <i>names</i> to the <b>To</b> list.                                                                                                                                                                                                                                                                                                                                                                                          |
| 96769 | <b>~v</b>                                                                                                   | Invoke the full-screen editor, as specified by the <i>VISUAL</i> environment variable, on the partial message.                                                                                                                                                                                                                                                                                                                             |
| 96770 |                                                                                                             |                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 96771 | <b>~w</b> <i>file</i>                                                                                       | Write the partial message, without the header, onto the file named by the pathname <i>file</i> . The file shall be created or the message shall be appended to it if the file exists.                                                                                                                                                                                                                                                      |
| 96772 |                                                                                                             |                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 96773 |                                                                                                             |                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 96774 | <b>~x</b>                                                                                                   | Exit as with <b>~q</b> , except the message shall not be saved in the dead-letter file.                                                                                                                                                                                                                                                                                                                                                    |
| 96775 | <b>~ </b> <i>command</i>                                                                                    | Pipe the body of the message through the given <i>command</i> by invoking the command interpreter specified by <i>SHELL</i> with two arguments: <b>-c</b> and <i>command</i> . If the <i>command</i> returns a successful exit status, the standard output of the command shall replace the message. Otherwise, the message shall remain unchanged. If the <i>command</i> fails, an error message giving the exit status shall be written. |
| 96776 |                                                                                                             |                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 96777 |                                                                                                             |                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 96778 |                                                                                                             |                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 96779 |                                                                                                             |                                                                                                                                                                                                                                                                                                                                                                                                                                            |

96780 **EXIT STATUS**96781 UP When the **-e** option is specified, the following exit values are returned:

96782     0 Mail was found.

96783     &gt;0 Mail was not found or an error occurred.

96784 Otherwise, the following exit values are returned:

96785     0 Successful completion; note that this status implies that all messages were *sent*, but it gives  
96786 no assurances that any of them were actually *delivered*.

96787     &gt;0 An error occurred.

96788 **CONSEQUENCES OF ERRORS**96789 UP When in **input mode (Receive Mode)** or Send Mode:96790     • If an error is encountered processing an input line beginning with a <tilde> ('~')  
96791 UP character, (see [Command Escapes in mailx](#), on page 2927), a diagnostic message shall be  
96792 written to standard error, and the message being composed may be modified, but this  
96793 condition shall not prevent the message from being sent.

96794     • Other errors shall prevent the sending of the message.

96795 UP When in command mode:

96796     • Default.

96797 **APPLICATION USAGE**96798 Delivery of messages to remote systems requires the existence of communication paths to such  
96799 systems. These need not exist.96800 Input lines are limited to {LINE\_MAX} bytes, but mailers between systems may impose more  
96801 severe line-length restrictions. This volume of POSIX.1-2008 does not place any restrictions on  
96802 the length of messages handled by *mailx*, and for delivery of local messages the only limitations  
96803 should be the normal problems of available disk space for the target mail file. When sending  
96804 messages to external machines, applications are advised to limit messages to less than 100 000  
96805 bytes because some mail gateways impose message-length restrictions.96806 The format of the system mailbox is intentionally unspecified. Not all systems implement  
96807 system mailboxes as flat files, particularly with the advent of multimedia mail messages. Some  
96808 system mailboxes may be multiple files, others records in a database. The internal format of the  
96809 messages themselves is specified with the historical format from Version 7, but only after the  
96810 messages have been saved in some file other than the system mailbox. This was done so that  
96811 many historical applications expecting text-file mailboxes are not broken.96812 Some new formats for messages can be expected in the future, probably including binary data,  
96813 bit maps, and various multimedia objects. As described here, *mailx* is not prohibited from  
96814 handling such messages, but it must store them as text files in secondary mailboxes (unless some  
96815 extension, such as a variable or command line option, is used to change the stored format). Its  
96816 method of doing so is implementation-defined and might include translating the data into text  
96817 file-compatible or readable form or omitting certain portions of the message from the stored  
96818 output.96819 The **discard** and **ignore** commands are not inverses of the **retain** command. The **retain**  
96820 command discards all header-fields except those explicitly retained. The **discard** command  
96821 keeps all header-fields except those explicitly discarded. If headers exist on the retained header

list, **discard** and **ignore** commands are ignored.

## 96823 EXAMPLES

96824 None.

## 96825 RATIONALE

96826 The standard developers felt strongly that a method for applications to send messages to specific  
 96827 users was necessary. The obvious example is a batch utility, running non-interactively, that  
 96828 wishes to communicate errors or results to a user. However, the actual format, delivery  
 96829 mechanism, and method of reading the message are clearly beyond the scope of this volume of  
 96830 POSIX.1-2008.

96831 The intent of this command is to provide a simple, portable interface for sending messages non-  
 96832 interactively. It merely defines a “front-end” to the historical mail system. It is suggested that  
 96833 implementations explicitly denote the sender and recipient in the body of the delivered message.  
 96834 Further specification of formats for either the message envelope or the message itself were  
 96835 deliberately not made, as the industry is in the midst of changing from the current standards to a  
 96836 more internationalized standard and it is probably incorrect, at this time, to require either one.

96837 Implementations are encouraged to conform to the various delivery mechanisms described in  
 96838 the CCITT X.400 standards or to the equivalent Internet standards, described in Internet Request  
 96839 for Comment (RFC) documents RFC 819, RFC 822, RFC 920, RFC 921, and RFC 1123.

96840 Many historical systems modified each body line that started with **From** by prefixing the ‘F’  
 96841 with ‘>’. It is unnecessary, but allowed, to do that when the string does not follow a blank line  
 96842 because it cannot be confused with the next header.

96843 The **edit** and **visual** commands merely edit the specified messages in a temporary file. They do  
 96844 not modify the contents of those messages in the mailbox; such a capability could be added as an  
 96845 extension, such as by using different command names.

96846 The restriction on a subject line being {LINE\_MAX}-10 bytes is based on the historical format  
 96847 that consumes 10 bytes for **Subject:** and the trailing <newline>. Many historical mailers that a  
 96848 message may encounter on other systems are not able to handle lines that long, however.

96849 Like the utilities *logger* and *lp*, *mailx* admittedly is difficult to test. This was not deemed sufficient  
 96850 justification to exclude this utility from this volume of POSIX.1-2008. It is also arguable that it is,  
 96851 in fact, testable, but that the tests themselves are not portable.

96852 When *mailx* is being used by an application that wishes to receive the results as if none of the  
 96853 User Portability Utilities option features were supported, the *DEAD* environment variable must  
 96854 be set to **/dev/null**. Otherwise, it may be subject to the file creations described in *mailx*  
 96855 ASYNCHRONOUS EVENTS. Similarly, if the *MAILRC* environment variable is not set to  
 96856 **/dev/null**, historical versions of *mailx* and *Mail* read initialization commands from a file before  
 96857 processing begins. Since the initialization that a user specifies could alter the contents of  
 96858 messages an application is trying to send, such applications must set *MAILRC* to **/dev/null**.

96859 The description of *LC\_TIME* uses “may affect” because many historical implementations do not  
 96860 or cannot manipulate the date and time strings in the incoming mail headers. Some headers  
 96861 found in incoming mail do not have enough information to determine the timezone in which the  
 96862 mail originated, and, therefore, *mailx* cannot convert the date and time strings into the internal  
 96863 form that then is parsed by routines like *strftime()* that can take *LC\_TIME* settings into account.  
 96864 Changing all these times to a user-specified format is allowed, but not required.

96865 The paginator selected when *PAGER* is null or unset is partially unspecified to allow the System  
 96866 V historical practice of using *pg* as the default. Bypassing the pagination function, such as by  
 96867 declaring that *cat* is the paginator, would not meet with the intended meaning of this



description. However, any “portable user” would have to set *PAGER* explicitly to get his or her preferred paginator on all systems. The paginator choice was made partially unspecified, unlike the *VISUAL* editor choice (mandated to be *vi*) because most historical pagers follow a common theme of user input, whereas editors differ dramatically.

Options to specify addresses as **cc** (carbon copy) or **bcc** (blind carbon copy) were considered to be format details and were omitted.

A zero exit status implies that all messages were *sent*, but it gives no assurances that any of them were actually *delivered*. The reliability of the delivery mechanism is unspecified and is an appropriate marketing distinction between systems.

In order to conform to the Utility Syntax Guidelines, a solution was required to the optional *file* option-argument to **-f**. By making *file* an operand, the guidelines are satisfied and users remain portable. However, it does force implementations to support usage such as:

```
mailx -fin mymail.box
```

The **no name** method of unsetting variables is not present in all historical systems, but it is in System V and provides a logical set of commands corresponding to the format of the display of options from the *mailx set* command without arguments.

The **ask** and **asksub** variables are the names selected by BSD and System V, respectively, for the same feature. They are synonyms in this volume of POSIX.1-2008.

The *mailx echo* command was not documented in the BSD version and has been omitted here because it is not obviously useful for interactive users.

The default prompt on the System V *mailx* is a <question-mark>, on BSD *Mail* an <ampersand>. Since this volume of POSIX.1-2008 chose the *mailx* name, it kept the System V default, assuming that BSD users would not have difficulty with this minor incompatibility (that they can override).

The meanings of **r** and **R** are reversed between System V *mailx* and SunOS *Mail*. Once again, since this volume of POSIX.1-2008 chose the *mailx* name, it kept the System V default, but allows the SunOS user to achieve the desired results using **flpr**, an internal variable in System V *mailx*, although it has not been documented in the SVID.

The **indentprefix** variable, the **retain** and **unalias** commands, and the **~F** and **~M** command escapes were adopted from 4.3 BSD *Mail*.

The **version** command was not included because no sufficiently general specification of the version information could be devised that would still be useful to a portable user. This command name should be used by suppliers who wish to provide version information about the *mailx* command.

The “implementation-specific (unspecified) system start-up file” historically has been named **/etc/mailx.rc**, but this specific name and location are not required.

The intent of the wording for the **next** command is that if any command has already displayed the current message it should display a following message, but, otherwise, it should display the current message. Consider the command sequence:

```
next 3
delete 3
next
```

where the **autoprint** option was not set. The normative text specifies that the second **next** command should display a message following the third message, because even though the

96912 current message has not been displayed since it was set by the **delete** command, it has been  
96913 displayed since the current message was anything other than message number 3. This does not  
96914 always match historical practice in some implementations, where the command file address  
96915 followed by **next** (or the default command) would skip the message for which the user had  
96916 searched.

#### 96917 FUTURE DIRECTIONS

96918 None.

#### 96919 SEE ALSO

96920 [Chapter 2](#) (on page 2319), *ed*, *ls*, *more*, *vi*

96921 [XBD Chapter 8](#) (on page 173), [Section 12.2](#) (on page 215)

#### 96922 CHANGE HISTORY

96923 First released in Issue 2.

#### 96924 Issue 5

96925 The description of the *EDITOR* environment variable is changed to indicate that *ed* is the default  
96926 editor if this variable is not set. In previous issues, this default was not stated explicitly at this  
96927 point but was implied further down in the text.

96928 The FUTURE DIRECTIONS section is added.

#### 96929 Issue 6

96930 The following new requirements on POSIX implementations derive from alignment with the  
96931 Single UNIX Specification:

- 96932 • The **-F** option is added.
- 96933 • The **allnet**, **debug**, and **sendwait** internal variables are added.
- 96934 • The **C**, **ec**, **fo**, **F**, and **S** *mailx* commands are added.

96935 In the DESCRIPTION and ENVIRONMENT VARIABLES sections, text stating “*HOME*  
96936 directory” is replaced by “directory referred to by the *HOME* environment variable”.

96937 The *mailx* utility is aligned with the IEEE P1003.2b draft standard, which includes various  
96938 clarifications to resolve IEEE PASC Interpretations submitted for the ISO POSIX-2:1993  
96939 standard. In particular, the changes here address IEEE PASC Interpretations 1003.2 #10, #11,  
96940 #103, #106, #108, #114, #115, #122, and #129.

96941 The normative text is reworded to avoid use of the term “must” for application requirements.

96942 The *TZ* entry is added to the ENVIRONMENT VARIABLES section.

96943 IEEE Std 1003.1-2001/Cor 1-2002, item XCU/TC1/D6/32 is applied, applying a change to the  
96944 EXTENDED DESCRIPTION, raised by IEEE PASC Interpretation 1003.2 #122, which was  
96945 overlooked in the first version of this standard.

96946 IEEE Std 1003.1-2001/Cor 2-2004, item XCU/TC2/D6/17 is applied, updating the EXTENDED  
96947 DESCRIPTION (Internal Variables in *mailx*). The system start-up file is changed from  
96948 “implementation-defined” to “unspecified” for consistency with other text in the EXTENDED  
96949 DESCRIPTION.

#### 96950 Issue 7

96951 Austin Group Interpretation 1003.1-2001 #089 is applied, clarifying the effect of the *LC\_TIME*  
96952 environment variable.

96953 Austin Group Interpretation 1003.1-2001 #090 is applied, updating the description of the **next**  
96954 command.

- 96955 SD5-XCU-ERN-69 is applied.
- 96956 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.
- 96957 Shading to indicate support for the User Portability Utilities option is added.

**NAME**

make — maintain, update, and regenerate groups of programs (**DEVELOPMENT**)

**SYNOPSIS**

```
SD make [-einpqrst] [-f makefile]... [-k|-S] [macro=value...]
 [target_name...]
```

**DESCRIPTION**

The *make* utility shall update files that are derived from other files. A typical case is one where object files are derived from the corresponding source files. The *make* utility examines time relationships and shall update those derived files (called targets) that have modified times earlier than the modified times of the files (called prerequisites) from which they are derived. A description file (makefile) contains a description of the relationships between files, and the commands that need to be executed to update the targets to reflect changes in their prerequisites. Each specification, or rule, shall consist of a target, optional prerequisites, and optional commands to be executed when a prerequisite is newer than the target. There are two types of rule:

1. *Inference rules*, which have one target name with at least one <period> ('.') and no <slash> ('/')
2. *Target rules*, which can have more than one target name

In addition, *make* shall have a collection of built-in macros and inference rules that infer prerequisite relationships to simplify maintenance of programs.

To receive exactly the behavior described in this section, the user shall ensure that a portable makefile shall:

- Include the special target **.POSIX**
- Omit any special target reserved for implementations (a leading period followed by uppercase letters) that has not been specified by this section

The behavior of *make* is unspecified if either or both of these conditions are not met.

**OPTIONS**

The *make* utility shall conform to XBD [Section 12.2](#) (on page 215), except for Guideline 9.

The following options shall be supported:

- e** Cause environment variables, including those with null values, to override macro assignments within makefiles.
- f *makefile*** Specify a different makefile. The argument *makefile* is a pathname of a description file, which is also referred to as the *makefile*. A pathname of '-' shall denote the standard input. There can be multiple instances of this option, and they shall be processed in the order specified. The effect of specifying the same option-argument more than once is unspecified.
- i** Ignore error codes returned by invoked commands. This mode is the same as if the special target **.IGNORE** were specified without prerequisites.
- k** Continue to update other targets that do not depend on the current target if a non-ignored error occurs while executing the commands to bring a target up-to-date.
- n** Write commands that would be executed on standard output, but do not execute them. However, lines with a <plus-sign> ('+') prefix shall be executed. In this mode, lines with an at-sign ('@') character prefix shall be written to standard

|       |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|-------|-----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 97001 |           | output.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 97002 | <b>-p</b> | Write to standard output the complete set of macro definitions and target descriptions. The output format is unspecified.                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 97003 |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| 97004 | <b>-q</b> | Return a zero exit value if the target file is up-to-date; otherwise, return an exit value of 1. Targets shall not be updated if this option is specified. However, a makefile command line (associated with the targets) with a <plus-sign> ('+') prefix shall be executed.                                                                                                                                                                                                                                                                                |
| 97005 |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| 97006 |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| 97007 |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| 97008 | <b>-r</b> | Clear the suffix list and do not use the built-in rules.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 97009 | <b>-S</b> | Terminate <i>make</i> if an error occurs while executing the commands to bring a target up-to-date. This shall be the default and the opposite of <b>-k</b> .                                                                                                                                                                                                                                                                                                                                                                                               |
| 97010 |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| 97011 | <b>-s</b> | Do not write makefile command lines or touch messages (see <b>-t</b> ) to standard output before executing. This mode shall be the same as if the special target <b>.SILENT</b> were specified without prerequisites.                                                                                                                                                                                                                                                                                                                                       |
| 97012 |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| 97013 |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| 97014 | <b>-t</b> | Update the modification time of each target as though a <i>touch target</i> had been executed. Targets that have prerequisites but no commands (see <a href="#">Target Rules</a> , on page 2939), or that are already up-to-date, shall not be touched in this manner. Write messages to standard output for each target file indicating the name of the file and that it was touched. Normally, the <i>makefile</i> command lines associated with each target are not executed. However, a command line with a <plus-sign> ('+') prefix shall be executed. |
| 97015 |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| 97016 |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| 97017 |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| 97018 |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| 97019 |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| 97020 |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |

Any options specified in the *MAKEFLAGS* environment variable shall be evaluated before any options specified on the *make* utility command line. If the **-k** and **-S** options are both specified on the *make* utility command line or by the *MAKEFLAGS* environment variable, the last option specified shall take precedence. If the **-f** or **-p** options appear in the *MAKEFLAGS* environment variable, the result is undefined.

## 97026 OPERANDS

97027 The following operands shall be supported:

97028 *target\_name* Target names, as defined in the EXTENDED DESCRIPTION section. If no target is  
 97029 specified, while *make* is processing the makefiles, the first target that *make*  
 97030 encounters that is not a special target or an inference rule shall be used.

97031 *macro=value* Macro definitions, as defined in [Macros](#) (on page 2940).

97032 If the *target\_name* and *macro=value* operands are intermixed on the *make* utility command line,  
 97033 the results are unspecified.

## 97034 STDIN

97035 The standard input shall be used only if the *makefile* option-argument is '-'. See the INPUT  
 97036 FILES section.

## 97037 INPUT FILES

97038 The input file, otherwise known as the makefile, is a text file containing rules, macro definitions,  
 97039 and comments. See the EXTENDED DESCRIPTION section.

## 97040 ENVIRONMENT VARIABLES

97041 The following environment variables shall affect the execution of *make*:

97042 *LANG* Provide a default value for the internationalization variables that are unset or null.  
 97043 (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 97044 variables used to determine the values of locale categories.)

|       |     |                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|-------|-----|----------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 97045 |     | <b>LC_ALL</b>              | If set to a non-empty string value, override the values of all the other internationalization variables.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| 97046 |     |                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 97047 |     | <b>LC_CTYPE</b>            | Determine the locale for the interpretation of sequences of bytes of text data as characters (for example, single-byte as opposed to multi-byte characters in arguments and input files).                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 97048 |     |                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 97049 |     |                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 97050 |     | <b>LC_MESSAGES</b>         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 97051 |     |                            | Determine the locale that should be used to affect the format and contents of diagnostic messages written to standard error.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 97052 |     |                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 97053 |     | <b>MAKEFLAGS</b>           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 97054 |     |                            | This variable shall be interpreted as a character string representing a series of option characters to be used as the default options. The implementation shall accept both of the following formats (but need not accept them when intermixed):                                                                                                                                                                                                                                                                                                                                                                   |
| 97055 |     |                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 97056 |     |                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 97057 |     |                            | • The characters are option letters without the leading <hyphen> characters or <blank> separation used on a <i>make</i> utility command line.                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 97058 |     |                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 97059 |     |                            | • The characters are formatted in a manner similar to a portion of the <i>make</i> utility command line: options are preceded by <hyphen> characters and <blank>-separated as described in XBD <a href="#">Section 12.2</a> (on page 215). The <i>macro=value</i> macro definition operands can also be included. The difference between the contents of <b>MAKEFLAGS</b> and the <i>make</i> utility command line is that the contents of the variable shall not be subjected to the word expansions (see <a href="#">Section 2.6</a> , on page 2327) associated with parsing the command line values.            |
| 97060 |     |                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 97061 |     |                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 97062 |     |                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 97063 |     |                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 97064 |     |                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 97065 |     |                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 97066 |     |                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 97067 | XSI | <b>NLSPATH</b>             | Determine the location of message catalogs for the processing of <b>LC_MESSAGES</b> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 97068 | XSI | <b>PROJECTDIR</b>          |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 97069 |     |                            | Provide a directory to be used to search for SCCS files not found in the current directory. In all of the following cases, the search for SCCS files is made in the directory <b>SCCS</b> in the identified directory. If the value of <b>PROJECTDIR</b> begins with a <slash>, it shall be considered an absolute pathname; otherwise, the value of <b>PROJECTDIR</b> is treated as a user name and that user's initial working directory shall be examined for a subdirectory <b>src</b> or <b>source</b> . If such a directory is found, it shall be used. Otherwise, the value is used as a relative pathname. |
| 97070 |     |                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 97071 |     |                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 97072 |     |                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 97073 |     |                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 97074 |     |                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 97075 |     |                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 97076 |     |                            | If <b>PROJECTDIR</b> is not set or has a null value, the search for SCCS files shall be made in the directory <b>SCCS</b> in the current directory.                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 97077 |     |                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 97078 |     |                            | The setting of <b>PROJECTDIR</b> affects all files listed in the remainder of this utility description for files with a component named <b>SCCS</b> .                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 97079 |     |                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 97080 |     |                            | The value of the <b>SHELL</b> environment variable shall not be used as a macro and shall not be modified by defining the <b>SHELL</b> macro in a makefile or on the command line. All other environment variables, including those with null values, shall be used as macros, as defined in <a href="#">Macros</a> (on page 2940).                                                                                                                                                                                                                                                                                |
| 97081 |     |                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 97082 |     |                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 97083 |     |                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 97084 |     | <b>ASYNCHRONOUS EVENTS</b> |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 97085 |     |                            | If not already ignored, <i>make</i> shall trap <b>SIGHUP</b> , <b>SIGTERM</b> , <b>SIGINT</b> , and <b>SIGQUIT</b> and remove the current target unless the target is a directory or the target is a prerequisite of the special target <b>.PRECIOUS</b> or unless one of the <b>-n</b> , <b>-p</b> , or <b>-q</b> options was specified. Any targets removed in this manner shall be reported in diagnostic messages of unspecified format, written to standard error. After this cleanup process, if any, <i>make</i> shall take the standard action for all other signals.                                      |
| 97086 |     |                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 97087 |     |                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 97088 |     |                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 97089 |     |                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 97090 |     |                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |

**STDOUT**

The *make* utility shall write all commands to be executed to standard output unless the `-s` option was specified, the command is prefixed with an at-sign, or the special target **.SILENT** has either the current target as a prerequisite or has no prerequisites. If *make* is invoked without any work needing to be done, it shall write a message to standard output indicating that no action was taken. If the `-t` option is present and a file is touched, *make* shall write to standard output a message of unspecified format indicating that the file was touched, including the filename of the file.

**STDERR**

The standard error shall be used only for diagnostic messages.

**OUTPUT FILES**

Files can be created when the `-t` option is present. Additional files can also be created by the utilities invoked by *make*.

**EXTENDED DESCRIPTION**

The *make* utility attempts to perform the actions required to ensure that the specified targets are up-to-date. A target is considered out-of-date if it is older than any of its prerequisites or if it does not exist. The *make* utility shall treat all prerequisites as targets themselves and recursively ensure that they are up-to-date, processing them in the order in which they appear in the rule. The *make* utility shall use the modification times of files to determine whether the corresponding targets are out-of-date.

After *make* has ensured that all of the prerequisites of a target are up-to-date and if the target is out-of-date, the commands associated with the target entry shall be executed. If there are no commands listed for the target, the target shall be treated as up-to-date.

**Makefile Syntax**

A makefile can contain rules, macro definitions (see [Macros](#), on page 2940), include lines, and comments. There are two kinds of rules: *inference rules* and *target rules*. The *make* utility shall contain a set of built-in inference rules. If the `-r` option is present, the built-in rules shall not be used and the suffix list shall be cleared. Additional rules of both types can be specified in a makefile. If a rule is defined more than once, the value of the rule shall be that of the last one specified. Macros can also be defined more than once, and the value of the macro is specified in [Macros](#) (on page 2940). Comments start with a <number-sign> ('#') and continue until an unescaped <newline> is reached.

By default, the following files shall be tried in sequence: **./makefile** and **./Makefile**. If neither **./makefile** or **./Makefile** are found, other implementation-defined files may also be tried. On XSI-conformant systems, the additional files **./s.makefile**, **SCCS/s.makefile**, **./s.Makefile**, and **SCCS/s.Makefile** shall also be tried.

The `-f` option shall direct *make* to ignore any of these default files and use the specified argument as a makefile instead. If the `'-'` argument is specified, standard input shall be used.

The term *makefile* is used to refer to any rules provided by the user, whether in **./makefile** or its variants, or specified by the `-f` option.

The rules in makefiles shall consist of the following types of lines: target rules, including special targets (see [Target Rules](#), on page 2939), inference rules (see [Inference Rules](#), on page 2942), macro definitions (see [Macros](#), on page 2940), empty lines, and comments.

Target and Inference Rules may contain *command lines*. Command lines can have a prefix that shall be removed before execution (see [Makefile Execution](#), on page 2938).



When an escaped <newline> (one preceded by a <backslash>) is found anywhere in the makefile except in a command line, an include line, or a line immediately preceding an include line, it shall be replaced, along with any leading white space on the following line, with a single <space>. When an escaped <newline> is found in a command line in a makefile, the command line shall contain the <backslash>, the <newline>, and the next line, except that the first character of the next line shall not be included if it is a <tab>. When an escaped <newline> is found in an include line or in a line immediately preceding an include line, the behavior is unspecified.

## Include Lines

If the word **include** appears at the beginning of a line and is followed by one or more <blank> characters, the string formed by the remainder of the line shall be processed as follows to produce a pathname:

- The trailing <newline> and any comment shall be discarded. If the resulting string contains any double-quote characters ( ' " ' ) the behavior is unspecified.
- The resulting string shall be processed for macro expansion (see [Macros](#) (on page 2940)).
- Any <blank> characters that appear after the first non-<blank> shall be used as separators to divide the macro-expanded string into fields. It is unspecified whether any other white-space characters are also used as separators. It is unspecified whether pathname expansion (see [Section 2.13](#), on page 2354) is also performed.
- If the processing of separators and optional pathname expansion results in either zero or two or more non-empty fields, the behavior is unspecified. If it results in one non-empty field, that field is taken as the pathname.

If the pathname does not begin with a ' / ' it shall be treated as relative to the current working directory of the process, not relative to the directory containing the makefile. If the file does not exist in this location, it is unspecified whether additional directories are searched.

The contents of the file specified by the pathname shall be read and processed as if they appeared in the makefile in place of the include line. If the file ends with an escaped <newline> the behavior is unspecified.

The file may itself contain further include lines. Implementations shall support nesting of include files up to a depth of at least 16.

## Makefile Execution

Makefile command lines shall be processed one at a time.

Makefile command lines can have one or more of the following prefixes: a <hyphen> ( ' - ' ), an at-sign ( ' @ ' ), or a <plus-sign> ( ' + ' ). These shall modify the way in which *make* processes the command.

- If the command prefix contains a <hyphen>, or the **-i** option is present, or the special target **.IGNORE** has either the current target as a prerequisite or has no prerequisites, any error found while executing the command shall be ignored.
- @ If the command prefix contains an at-sign and the *make* utility command line **-n** option is not specified, or the **-s** option is present, or the special target **.SILENT** has either the current target as a prerequisite or has no prerequisites, the command shall not be written to standard output before it is executed.



- + If the command prefix contains a <plus-sign>, this indicates a makefile command line that shall be executed even if **-n**, **-q**, or **-t** is specified.

An *execution line* is built from the command line by removing any prefix characters. Except as described under the at-sign prefix, the execution line shall be written to the standard output, optionally preceded by a <tab>. The execution line shall then be executed by a shell as if it were passed as the argument to the *system()* interface, except that if errors are not being ignored then the shell **-e** option shall also be in effect. If errors are being ignored for the command (as a result of the **-i** option, a **'-'** command prefix, or a **.IGNORE** special target), the shell **-e** option shall not be in effect. The environment for the command being executed shall contain all of the variables in the environment of *make*.

By default, when *make* receives a non-zero status from the execution of a command, it shall terminate with an error message to standard error.

## Target Rules

Target rules are formatted as follows:

```
target [target...]: [prerequisite...][;command]
[<tab>command
<tab>command
...]
```

*line that does not begin with <tab>*

Target entries are specified by a <blank>-separated, non-null list of targets, then a <colon>, then a <blank>-separated, possibly empty list of prerequisites. Text following a <semicolon>, if any, and all following lines that begin with a <tab>, are makefile command lines to be executed to update the target. The first non-empty line that does not begin with a <tab> or **'#'** shall begin a new entry. An empty or blank line, or a line beginning with **'#'**, may begin a new entry.

Applications shall select target names from the set of characters consisting solely of periods, underscores, digits, and alphabets from the portable character set (see XBD [Section 6.1](#), on page 125). Implementations may allow other characters in target names as extensions. The interpretation of targets containing the characters **'%'** and **'\"'** is implementation-defined.

A target that has prerequisites, but does not have any commands, can be used to add to the prerequisite list for that target. Only one target rule for any given target can contain commands.

Lines that begin with one of the following are called *special targets* and control the operation of *make*:

**.DEFAULT** If the makefile uses this special target, the application shall ensure that it is specified with commands, but without prerequisites. The commands shall be used by *make* if there are no other rules available to build a target.

**.IGNORE** Prerequisites of this special target are targets themselves; this shall cause errors from commands associated with them to be ignored in the same manner as specified by the **-i** option. Subsequent occurrences of **.IGNORE** shall add to the list of targets ignoring command errors. If no prerequisites are specified, *make* shall behave as if the **-i** option had been specified and errors from all commands associated with all targets shall be ignored.

**.POSIX** The application shall ensure that this special target is specified without prerequisites or commands. If it appears as the first non-comment line in the makefile, *make* shall process the makefile as specified by this section; otherwise, the behavior of *make* is unspecified.

97223 **.PRECIOUS** Prerequisites of this special target shall not be removed if *make* receives one of the  
 97224 asynchronous events explicitly described in the ASYNCHRONOUS EVENTS  
 97225 section. Subsequent occurrences of **.PRECIOUS** shall add to the list of precious  
 97226 files. If no prerequisites are specified, all targets in the makefile shall be treated as  
 97227 if specified with **.PRECIOUS**.

97228 XSI **.SCCS\_GET** The application shall ensure that this special target is specified without  
 97229 prerequisites. If this special target is included in a makefile, the commands  
 97230 specified with this target shall replace the default commands associated with this  
 97231 special target (see [Default Rules](#), on page 2945). The commands specified with this  
 97232 target are used to get all SCCS files that are not found in the current directory.

97233 When source files are named in a dependency list, *make* shall treat them just like  
 97234 any other target. Because the source file is presumed to be present in the directory,  
 97235 there is no need to add an entry for it to the makefile. When a target has no  
 97236 dependencies, but is present in the directory, *make* shall assume that that file is up-  
 97237 to-date. If, however, an SCCS file named **SCCS/s.source\_file** is found for a target  
 97238 *source\_file*, *make* compares the timestamp of the target file with that of the  
 97239 **SCCS/s.source\_file** to ensure the target is up-to-date. If the target is missing, or if  
 97240 the SCCS file is newer, *make* shall automatically issue the commands specified for  
 97241 the **.SCCS\_GET** special target to retrieve the most recent version. However, if the  
 97242 target is writable by anyone, *make* shall not retrieve a new version.

97243 **.SILENT** Prerequisites of this special target are targets themselves; this shall cause  
 97244 commands associated with them not to be written to the standard output before  
 97245 they are executed. Subsequent occurrences of **.SILENT** shall add to the list of  
 97246 targets with silent commands. If no prerequisites are specified, *make* shall behave  
 97247 as if the **-s** option had been specified and no commands or touch messages  
 97248 associated with any target shall be written to standard output.

97249 **.SUFFIXES** Prerequisites of **.SUFFIXES** shall be appended to the list of known suffixes and are  
 97250 used in conjunction with the inference rules (see [Inference Rules](#), on page 2942). If  
 97251 **.SUFFIXES** does not have any prerequisites, the list of known suffixes shall be  
 97252 cleared.

97253 The special targets **.IGNORE**, **.POSIX**, **.PRECIOUS**, **.SILENT**, and **.SUFFIXES** shall be specified  
 97254 without commands.

97255 Targets with names consisting of a leading <period> followed by the uppercase letters "POSIX"  
 97256 and then any other characters are reserved for future standardization. Targets with names  
 97257 consisting of a leading <period> followed by one or more uppercase letters are reserved for  
 97258 implementation extensions.

## 97259 Macros

97260 Macro definitions are in the form:

97261 *string1* = [*string2*]

97262 The macro named *string1* is defined as having the value of *string2*, where *string2* is defined as all  
 97263 characters, if any, after the <equals-sign>, up to a comment character ('#') or an unescaped  
 97264 <newline>. Any <blank> characters immediately before or after the <equals-sign> shall be  
 97265 ignored.

97266 Applications shall select macro names from the set of characters consisting solely of periods,  
 97267 underscores, digits, and alphabets from the portable character set (see XBD [Section 6.1](#), on  
 97268 page 125). A macro name shall not contain an <equals-sign>. Implementations may allow other

characters in macro names as extensions.

Macros can appear anywhere in the makefile. Macro expansions using the forms  $\$(string1)$  or  $\${string1}$  shall be replaced by *string2*, as follows:

- Macros in target lines shall be evaluated when the target line is read.
- Macros in makefile command lines shall be evaluated when the command is executed.
- Macros in the string before the <equals-sign> in a macro definition shall be evaluated when the macro assignment is made.
- Macros after the <equals-sign> in a macro definition shall not be evaluated until the defined macro is used in a rule or command, or before the <equals-sign> in a macro definition.

The parentheses or braces are optional if *string1* is a single character. The macro  $\$\$$  shall be replaced by the single character '\$'. If *string1* in a macro expansion contains a macro expansion, the results are unspecified.

Macro expansions using the forms  $\$(string1[:subst1=[subst2]])$  or  $\${string1[:subst1=[subst2]]}$  can be used to replace all occurrences of *subst1* with *subst2* when the macro substitution is performed. The *subst1* to be replaced shall be recognized when it is a suffix at the end of a word in *string1* (where a *word*, in this context, is defined to be a string delimited by the beginning of the line, a <blank>, or a <newline>). If *string1* in a macro expansion contains a macro expansion, the results are unspecified.

Macro expansions in *string1* of macro definition lines shall be evaluated when read. Macro expansions in *string2* of macro definition lines shall be performed when the macro identified by *string1* is expanded in a rule or command.

Macro definitions shall be taken from the following sources, in the following logical order, before the makefile(s) are read.

1. Macros specified on the *make* utility command line, in the order specified on the command line. It is unspecified whether the internal macros defined in [Internal Macros](#) (on page 2943) are accepted from this source.
2. Macros defined by the *MAKEFLAGS* environment variable, in the order specified in the environment variable. It is unspecified whether the internal macros defined in [Internal Macros](#) (on page 2943) are accepted from this source.
3. The contents of the environment, excluding the *MAKEFLAGS* and *SHELL* variables and including the variables with null values.
4. Macros defined in the inference rules built into *make*.

Macro definitions from these sources shall not override macro definitions from a lower-numbered source. Macro definitions from a single source (for example, the *make* utility command line, the *MAKEFLAGS* environment variable, or the other environment variables) shall override previous macro definitions from the same source.

Macros defined in the makefile(s) shall override macro definitions that occur before them in the makefile(s) and macro definitions from source 4. If the *-e* option is not specified, macros defined in the makefile(s) shall override macro definitions from source 3. Macros defined in the makefile(s) shall not override macro definitions from source 1 or source 2.

Before the makefile(s) are read, all of the *make* utility command line options (except *-f* and *-p*) and *make* utility command line macro definitions (except any for the *MAKEFLAGS* macro), not already included in the *MAKEFLAGS* macro, shall be added to the *MAKEFLAGS* macro, quoted

in an implementation-defined manner such that when *MAKEFLAGS* is read by another instance of the *make* command, the original macro's value is recovered. Other implementation-defined options and macros may also be added to the *MAKEFLAGS* macro. If this modifies the value of the *MAKEFLAGS* macro, or, if the *MAKEFLAGS* macro is modified at any subsequent time, the *MAKEFLAGS* environment variable shall be modified to match the new value of the *MAKEFLAGS* macro. The result of setting *MAKEFLAGS* in the Makefile is unspecified.

Before the makefile(s) are read, all of the *make* utility command line macro definitions (except the *MAKEFLAGS* macro or the *SHELL* macro) shall be added to the environment of *make*. Other implementation-defined variables may also be added to the environment of *make*.

The **SHELL** macro shall be treated specially. It shall be provided by *make* and set to the pathname of the shell command language interpreter (see *sh*). The *SHELL* environment variable shall not affect the value of the **SHELL** macro. If **SHELL** is defined in the makefile or is specified on the command line, it shall replace the original value of the **SHELL** macro, but shall not affect the *SHELL* environment variable. Other effects of defining **SHELL** in the makefile or on the command line are implementation-defined.

## Inference Rules

Inference rules are formatted as follows:

```
target:
<tab>command
[<tab>command]
...
line that does not begin with <tab> or #
```

The application shall ensure that the *target* portion is a valid target name (see [Target Rules](#), on page 2939) of the form *.s2* or *.s1.s2* (where *.s1* and *.s2* are suffixes that have been given as prerequisites of the **.SUFFIXES** special target and *s1* and *s2* do not contain any *<slash>* or *<period>* characters.) If there is only one *<period>* in the target, it is a single-suffix inference rule. Targets with two periods are double-suffix inference rules. Inference rules can have only one target before the *<colon>*.

The application shall ensure that the makefile does not specify prerequisites for inference rules; no characters other than white space shall follow the *<colon>* in the first line, except when creating the *empty rule*, described below. Prerequisites are inferred, as described below.

Inference rules can be redefined. A target that matches an existing inference rule shall overwrite the old inference rule. An empty rule can be created with a command consisting of simply a *<semicolon>* (that is, the rule still exists and is found during inference rule search, but since it is empty, execution has no effect). The empty rule can also be formatted as follows:

```
rule: ;
```

where zero or more *<blank>* characters separate the *<colon>* and *<semicolon>*.

The *make* utility uses the suffixes of targets and their prerequisites to infer how a target can be made up-to-date. A list of inference rules defines the commands to be executed. By default, *make* contains a built-in set of inference rules. Additional rules can be specified in the makefile.

The special target **.SUFFIXES** contains as its prerequisites a list of suffixes that shall be used by the inference rules. The order in which the suffixes are specified defines the order in which the inference rules for the suffixes are used. New suffixes shall be appended to the current list by specifying a **.SUFFIXES** special target in the makefile. A **.SUFFIXES** target with no prerequisites shall clear the list of suffixes. An empty **.SUFFIXES** target followed by a new **.SUFFIXES** list is

required to change the order of the suffixes.

Normally, the user would provide an inference rule for each suffix. The inference rule to update a target with a suffix `.s1` from a prerequisite with a suffix `.s2` is specified as a target `.s2.s1`. The internal macros provide the means to specify general inference rules (see [Internal Macros](#)).

When no target rule is found to update a target, the inference rules shall be checked. The suffix of the target (`.s1`) to be built is compared to the list of suffixes specified by the `.SUFFIXES` special targets. If the `.s1` suffix is found in `.SUFFIXES`, the inference rules shall be searched in the order defined for the first `.s2.s1` rule whose prerequisite file (`$(s2)`) exists. If the target is out-of-date with respect to this prerequisite, the commands for that inference rule shall be executed.

If the target to be built does not contain a suffix and there is no rule for the target, the single suffix inference rules shall be checked. The single-suffix inference rules define how to build a target if a file is found with a name that matches the target name with one of the single suffixes appended. A rule with one suffix `.s2` is the definition of how to build *target* from `target.s2`. The other suffix (`.s1`) is treated as null.

A `<tilde> (' ~')` in the above rules refers to an SCCS file in the current directory. Thus, the rule `.c.o` would transform an SCCS C-language source file into an object file (`.o`). Because the `s.` of the SCCS files is a prefix, it is incompatible with *make's* suffix point of view. Hence, the `' ~'` is a way of changing any file reference into an SCCS file reference.

## Libraries

If a target or prerequisite contains parentheses, it shall be treated as a member of an archive library. For the `lib(member.o)` expression *lib* refers to the name of the archive library and *member.o* to the member name. The application shall ensure that the member is an object file with the `.o` suffix. The modification time of the expression is the modification time for the member as kept in the archive library; see [ar](#). The `.a` suffix shall refer to an archive library. The `.s2.a` rule shall be used to update a member in the library from a file with a suffix `.s2`.

## Internal Macros

The *make* utility shall maintain five internal macros that can be used in target and inference rules. In order to clearly define the meaning of these macros, some clarification of the terms *target rule*, *inference rule*, *target*, and *prerequisite* is necessary.

Target rules are specified by the user in a makefile for a particular target. Inference rules are user-specified or *make*-specified rules for a particular class of target name. Explicit prerequisites are those prerequisites specified in a makefile on target lines. Implicit prerequisites are those prerequisites that are generated when inference rules are used. Inference rules are applied to implicit prerequisites or to explicit prerequisites that do not have target rules defined for them in the makefile. Target rules are applied to targets specified in the makefile.

Before any target in the makefile is updated, each of its prerequisites (both explicit and implicit) shall be updated. This shall be accomplished by recursively processing each prerequisite. Upon recursion, each prerequisite shall become a target itself. Its prerequisites in turn shall be processed recursively until a target is found that has no prerequisites, at which point the recursion stops. The recursion shall then back up, updating each target as it goes.

In the definitions that follow, the word *target* refers to one of:

- A target specified in the makefile

- An explicit prerequisite specified in the makefile that becomes the target when *make* processes it during recursion
- An implicit prerequisite that becomes a target when *make* processes it during recursion

In the definitions that follow, the word *prerequisite* refers to one of the following:

- An explicit prerequisite specified in the makefile for a particular target
- An implicit prerequisite generated as a result of locating an appropriate inference rule and corresponding file that matches the suffix of the target

The five internal macros are:

**\$@** The **\$@** shall evaluate to the full target name of the current target, or the archive filename part of a library archive target. It shall be evaluated for both target and inference rules.

For example, in the **.c.a** inference rule, **\$@** represents the out-of-date **.a** file to be built. Similarly, in a makefile target rule to build **lib.a** from **file.c**, **\$@** represents the out-of-date **lib.a**.

**\$%** The **\$%** macro shall be evaluated only when the current target is an archive library member of the form *libname(member.o)*. In these cases, **\$@** shall evaluate to *libname* and **\$%** shall evaluate to *member.o*. The **\$%** macro shall be evaluated for both target and inference rules.

For example, in a makefile target rule to build **lib.a(file.o)**, **\$%** represents **file.o**, as opposed to **\$@**, which represents **lib.a**.

**\$?** The **\$?** macro shall evaluate to the list of prerequisites that are newer than the current target. It shall be evaluated for both target and inference rules.

For example, in a makefile target rule to build *prog* from **file1.o**, **file2.o**, and **file3.o**, and where *prog* is not out-of-date with respect to **file1.o**, but is out-of-date with respect to **file2.o** and **file3.o**, **\$?** represents **file2.o** and **file3.o**.

**\$<** In an inference rule, the **\$<** macro shall evaluate to the filename whose existence allowed the inference rule to be chosen for the target. In the **.DEFAULT** rule, the **\$<** macro shall evaluate to the current target name. The meaning of the **\$<** macro shall be otherwise unspecified.

For example, in the **.c.a** inference rule, **\$<** represents the prerequisite **.c** file.

**\$\*** The **\$\*** macro shall evaluate to the current target name with its suffix deleted. It shall be evaluated at least for inference rules.

For example, in the **.c.a** inference rule, **\$\*.o** represents the out-of-date **.o** file that corresponds to the prerequisite **.c** file.

Each of the internal macros has an alternative form. When an uppercase **'D'** or **'F'** is appended to any of the macros, the meaning shall be changed to the *directory part* for **'D'** and *filename part* for **'F'**. The directory part is the path prefix of the file without a trailing **<slash>**; for the current directory, the directory part is **'.'**. When the **\$?** macro contains more than one prerequisite filename, the **\$(?D)** and **\$(?F)** (or **\${?D}** and **\${?F}**) macros expand to a list of directory name parts and filename parts respectively.

For the target **lib(member.o)** and the **s2.a** rule, the internal macros shall be defined as:

```

97441 $< member.s2
97442 $* member
97443 @$ lib
97444 $? member.s2
97445 $% member.o

```

## 97446 Default Rules

97447 The default rules for *make* shall achieve results that are the same as if the following were used.  
 97448 Implementations that do not support the C-Language Development Utilities option may omit  
 97449 **CC**, **CFLAGS**, **YACC**, **YFLAGS**, **LEX**, **LFLAGS**, **LDFLAGS**, and the **.c**, **.y**, and **.l** inference rules.  
 97450 Implementations that do not support FORTRAN may omit **FC**, **FFLAGS**, and the **.f** inference  
 97451 rules. Implementations may provide additional macros and rules.

## 97452 SPECIAL TARGETS

```

97453 XSI .SCCS_GET: sccs $(SCCSFLAGS) get $(SCCSGETFLAGS) $@

```

```

97454 XSI .SUFFIXES: .o .c .y .l .a .sh .f .c~ .y~ .l~ .sh~ .f~

```

## 97455 MACROS

```

97456 MAKE=make
97457 AR=ar
97458 ARFLAGS=-rv
97459 YACC=yacc
97460 YFLAGS=
97461 LEX=lex
97462 LFLAGS=
97463 LDFLAGS=
97464 CC=c99
97465 CFLAGS=-O
97466 FC=fort77
97467 FFLAGS=-O 1
97468 XSI GET=get
97469 GFLAGS=
97470 SCCSFLAGS=
97471 SCCSGETFLAGS=-s

```

## 97472 SINGLE SUFFIX RULES

```

97473 .c:
97474 $(CC) $(CFLAGS) $(LDFLAGS) -o $@ $<
97475 .f:
97476 $(FC) $(FFLAGS) $(LDFLAGS) -o $@ $<
97477 .sh:
97478 cp $< $@
97479 chmod a+x $@
97480 XSI .c~:
97481 $(GET) $(GFLAGS) -p $< > $*.c

```

```

97482 $(CC) $(CFLAGS) $(LDFLAGS) -o $@ $*.c
97483 .f~:
97484 $(GET) $(GFLAGS) -p $< > $*.f
97485 $(FC) $(FFLAGS) $(LDFLAGS) -o $@ $*.f
97486 .sh~:
97487 $(GET) $(GFLAGS) -p $< > $*.sh
97488 cp $*.sh $@
97489 chmod a+x $@

```

97490 *DOUBLE SUFFIX RULES*

```

97491 .c.o:
97492 $(CC) $(CFLAGS) -c $<
97493 .f.o:
97494 $(FC) $(FFLAGS) -c $<
97495 .y.o:
97496 $(YACC) $(YFLAGS) $<
97497 $(CC) $(CFLAGS) -c y.tab.c
97498 rm -f y.tab.c
97499 mv y.tab.o $@
97500 .l.o:
97501 $(LEX) $(LFLAGS) $<
97502 $(CC) $(CFLAGS) -c lex.yy.c
97503 rm -f lex.yy.c
97504 mv lex.yy.o $@
97505 .y.c:
97506 $(YACC) $(YFLAGS) $<
97507 mv y.tab.c $@
97508 .l.c:
97509 $(LEX) $(LFLAGS) $<
97510 mv lex.yy.c $@

```

```

97511 XSI .c~.o:
97512 $(GET) $(GFLAGS) -p $< > $*.c
97513 $(CC) $(CFLAGS) -c $*.c
97514 .f~.o:
97515 $(GET) $(GFLAGS) -p $< > $*.f
97516 $(FC) $(FFLAGS) -c $*.f
97517 .y~.o:
97518 $(GET) $(GFLAGS) -p $< > $*.y
97519 $(YACC) $(YFLAGS) $*.y
97520 $(CC) $(CFLAGS) -c y.tab.c
97521 rm -f y.tab.c
97522 mv y.tab.o $@
97523 .l~.o:
97524 $(GET) $(GFLAGS) -p $< > $*.l
97525 $(LEX) $(LFLAGS) $*.l

```



```

97526 $(CC) $(CFLAGS) -c lex.yy.c
97527 rm -f lex.yy.c
97528 mv lex.yy.o $@

97529 .y~.c:
97530 $(GET) $(GFLAGS) -p $< > $*.y
97531 $(YACC) $(YFLAGS) $*.y
97532 mv y.tab.c $@

97533 .l~.c:
97534 $(GET) $(GFLAGS) -p $< > $*.l
97535 $(LEX) $(LFLAGS) $*.l
97536 mv lex.yy.c $@

```

```

97537 .c.a:
97538 $(CC) -c $(CFLAGS) $<
97539 $(AR) $(ARFLAGS) $@ $*.o
97540 rm -f $*.o

97541 .f.a:
97542 $(FC) -c $(FFLAGS) $<
97543 $(AR) $(ARFLAGS) $@ $*.o
97544 rm -f $*.o

```

## EXIT STATUS

When the **-q** option is specified, the *make* utility shall exit with one of the following values:

- 0 Successful completion.
- 1 The target was not up-to-date.
- >1 An error occurred.

When the **-q** option is not specified, the *make* utility shall exit with one of the following values:

- 0 Successful completion.
- >0 An error occurred.

## CONSEQUENCES OF ERRORS

Default.

## APPLICATION USAGE

If there is a source file (such as *./source.c*) and there are two SCCS files corresponding to it (*./s.source.c* and *./SCCS/s.source.c*), on XSI-conformant systems *make* uses the SCCS file in the current directory. However, users are advised to use the underlying SCCS utilities (*admin*, *delta*, *get*, and so on) or the *sccs* utility for all source files in a given directory. If both forms are used for a given source file, future developers are very likely to be confused.

It is incumbent upon portable makefiles to specify the **.POSIX** special target in order to guarantee that they are not affected by local extensions.

The **-k** and **-S** options are both present so that the relationship between the command line, the *MAKEFLAGS* variable, and the makefile can be controlled precisely. If the **k** flag is passed in *MAKEFLAGS* and a command is of the form:

```
$(MAKE) -S foo
```

then the default behavior is restored for the child *make*.

When the `-n` option is specified, it is always added to `MAKEFLAGS`. This allows a recursive `make -n target` to be used to see all of the action that would be taken to update *target*.

Because of widespread historical practice, interpreting a `<number-sign>` (`'#'`) inside a variable as the start of a comment has the unfortunate side-effect of making it impossible to place a `<number-sign>` in a variable, thus forbidding something like:

```
CFLAGS = "-D COMMENT_CHAR='#' "
```

Many historical *make* utilities stop chaining together inference rules when an intermediate target is nonexistent. For example, it might be possible for a *make* to determine that both `.y.c` and `.c.o` could be used to convert a `.y` to a `.o`. Instead, in this case, *make* requires the use of a `.y.o` rule.

The best way to provide portable makefiles is to include all of the rules needed in the makefile itself. The rules provided use only features provided by other parts of this volume of POSIX.1-2008. The default rules include rules for optional commands in this volume of POSIX.1-2008. Only rules pertaining to commands that are provided are needed in an implementation's default set.

Macros used within other macros are evaluated when the new macro is used rather than when the new macro is defined. Therefore:

```
MACRO = value1
NEW = $(MACRO)
MACRO = value2
```

```
target:
 echo $(NEW)
```

would produce *value2* and not *value1* since `NEW` was not expanded until it was needed in the *echo* command line.

Some historical applications have been known to intermix *target\_name* and *macro=name* operands on the command line, expecting that all of the macros are processed before any of the targets are dealt with. Conforming applications do not do this, although some backwards-compatibility support may be included in some implementations.

The following characters in filenames may give trouble: `'=`, `':`, `'`, `'`, `'`, `'`, single-quote, and `'@`. In include filenames, pattern matching characters and `'` `"` `'` should also be avoided, as they may be treated as special by some implementations.

For inference rules, the description of `$<` and `$?` seem similar. However, an example shows the minor difference. In a makefile containing:

```
foo.o: foo.h
```

if `foo.h` is newer than `foo.o`, yet `foo.c` is older than `foo.o`, the built-in rule to make `foo.o` from `foo.c` is used, with `$<` equal to `foo.c` and `$?` equal to `foo.h`. If `foo.c` is also newer than `foo.o`, `$<` is equal to `foo.c` and `$?` is equal to `foo.h foo.c`.

## EXAMPLES

1. The following command:

```
make
```

makes the first target found in the makefile.

2. The following command:

```
make junk
```

97610 makes the target **junk**.

97611 3. The following makefile says that **pgm** depends on two files, **a.o** and **b.o**, and that they in

97612 turn depend on their corresponding source files (**a.c** and **b.c**), and a common file **incl.h**:

97613 `pgm: a.o b.o`

97614 `c99 a.o b.o -o pgm`

97615 `a.o: incl.h a.c`

97616 `c99 -c a.c`

97617 `b.o: incl.h b.c`

97618 `c99 -c b.c`

97619 4. An example for making optimized **.o** files from **.c** files is:

97620 `.c.o:`

97621 `c99 -c -O $*.c`

97622 or:

97623 `.c.o:`

97624 `c99 -c -O $<`

97625 5. The most common use of the archive interface follows. Here, it is assumed that the source

97626 files are all C-language source:

97627 `lib: lib(file1.o) lib(file2.o) lib(file3.o)`

97628 `@echo lib is now up-to-date`

97629 The **.c.a** rule is used to make **file1.o**, **file2.o**, and **file3.o** and insert them into **lib**.

97630 The treatment of escaped <newline> characters throughout the makefile is historical

97631 practice. For example, the inference rule:

97632 `.c.o\`

97633 `:`

97634 works, and the macro:

97635 `f= bar baz\`

97636 `biz`

97637 `a:`

97638 `echo ==$f==`

97639 echoes `"==bar baz biz=="`.

97640 If  `$?`  were:

97641 `/usr/include/stdio.h /usr/include/unistd.h foo.h`

97642 then `$(?D)` would be:

97643 `/usr/include /usr/include .`

97644 and `$(?F)` would be:

97645 `stdio.h unistd.h foo.h`

97646 6. The contents of the built-in rules can be viewed by running:

97647 `make -p -f /dev/null 2>/dev/null`

**RATIONALE**

The *make* utility described in this volume of POSIX.1-2008 is intended to provide the means for changing portable source code into executables that can be run on an POSIX.1-2008-conforming system. It reflects the most common features present in System V and BSD *makes*.

Historically, the *make* utility has been an especially fertile ground for vendor and research organization-specific syntax modifications and extensions. Examples include:

- Syntax supporting parallel execution (such as from various multi-processor vendors, GNU, and others)
- Additional “operators” separating targets and their prerequisites (System V, BSD, and others)
- Specifying that command lines containing the strings “`${MAKE}`” and “`$(MAKE)`” are executed when the `-n` option is specified (GNU and System V)
- Modifications of the meaning of internal macros when referencing libraries (BSD and others)
- Using a single instance of the shell for all of the command lines of the target (BSD and others)
- Allowing `<space>` characters as well as `<tab>` characters to delimit command lines (BSD)
- Adding C preprocessor-style “include” and “ifdef” constructs (System V, GNU, BSD, and others)
- Remote execution of command lines (Sprite and others)
- Specifying additional special targets (BSD, System V, and most others)

Additionally, many vendors and research organizations have rethought the basic concepts of *make*, creating vastly extended, as well as completely new, syntaxes. Each of these versions of *make* fulfills the needs of a different community of users; it is unreasonable for this volume of POSIX.1-2008 to require behavior that would be incompatible (and probably inferior) to historical practice for such a community.

In similar circumstances, when the industry has enough sufficiently incompatible formats as to make them irreconcilable, this volume of POSIX.1-2008 has followed one or both of two courses of action. Commands have been renamed (*cksum*, *echo*, and *pax*) and/or command line options have been provided to select the desired behavior (*grep*, *od*, and *pax*).

Because the syntax specified for the *make* utility is, by and large, a subset of the syntaxes accepted by almost all versions of *make*, it was decided that it would be counter-productive to change the name. And since the makefile itself is a basic unit of portability, it would not be completely effective to reserve a new option letter, such as *make -P*, to achieve the portable behavior. Therefore, the special target **.POSIX** was added to the makefile, allowing users to specify “standard” behavior. This special target does not preclude extensions in the *make* utility, nor does it preclude such extensions being used by the makefile specifying the target; it does, however, preclude any extensions from being applied that could alter the behavior of previously valid syntax; such extensions must be controlled via command line options or new special targets. It is incumbent upon portable makefiles to specify the **.POSIX** special target in order to guarantee that they are not affected by local extensions.

The portable version of *make* described in this reference page is not intended to be the state-of-the-art software generation tool and, as such, some newer and more leading-edge features have not been included. An attempt has been made to describe the portable makefile in a manner that does not preclude such extensions as long as they do not disturb the portable behavior described

here.

When the `-n` option is specified, it is always added to `MAKEFLAGS`. This allows a recursive `make -n target` to be used to see all of the action that would be taken to update *target*.

The definition of `MAKEFLAGS` allows both the System V letter string and the BSD command line formats. The two formats are sufficiently different to allow implementations to support both without ambiguity.

Early proposals stated that an “unquoted” `<number-sign>` was treated as the start of a comment. The *make* utility does not pay any attention to quotes. A `<number-sign>` starts a comment regardless of its surroundings.

The text about “other implementation-defined pathnames may also be tried” in addition to `./makefile` and `./Makefile` is to allow such extensions as `SCCS/s.Makefile` and other variations. It was made an implementation-defined requirement (as opposed to unspecified behavior) to highlight surprising implementations that might select something unexpected like `/etc/Makefile`. XSI-conformant systems also try `./s.makefile`, `SCCS/s.makefile`, `./s.Makefile`, and `SCCS/s.Makefile`.

Early proposals contained the macro `NPROC` as a means of specifying that *make* should use *n* processes to do the work required. While this feature is a valuable extension for many systems, it is not common usage and could require other non-trivial extensions to makefile syntax. This extension is not required by this volume of POSIX.1-2008, but could be provided as a compatible extension. The macro `PARALLEL` is used by some historical systems with essentially the same meaning (but without using a name that is a common system limit value). It is suggested that implementors recognize the existing use of `NPROC` and/or `PARALLEL` as extensions to *make*.

The default rules are based on System V. The default `CC=` value is `c99` instead of `cc` because this volume of POSIX.1-2008 does not standardize the utility named `cc`. Thus, every conforming application would be required to define `CC=c99` to expect to run. There is no advantage conferred by the hope that the makefile might hit the “preferred” compiler because this cannot be guaranteed to work. Also, since the portable makescript can only use the `c99` options, no advantage is conferred in terms of what the script can do. It is a quality-of-implementation issue as to whether `c99` is as valuable as `cc`.

The `-d` option to *make* is frequently used to produce debugging information, but is too implementation-defined to add to this volume of POSIX.1-2008.

The `-p` option is not passed in `MAKEFLAGS` on most historical implementations and to change this would cause many implementations to break without sufficiently increased portability.

Commands that begin with a `<plus-sign>` (`'+'`) are executed even if the `-n` option is present. Based on the GNU version of *make*, the behavior of `-n` when the `<plus-sign>` prefix is encountered has been extended to apply to `-q` and `-t` as well. However, the System V convention of forcing command execution with `-n` when the command line of a target contains either of the strings `"$(MAKE)"` or `"${MAKE}"` has not been adopted. This functionality appeared in early proposals, but the danger of this approach was pointed out with the following example of a portion of a makefile:

```
subdir:
 cd subdir; rm all_the_files; $(MAKE)
```

The loss of the System V behavior in this case is well-balanced by the safety afforded to other makefiles that were not aware of this situation. In any event, the command line `<plus-sign>` prefix can provide the desired functionality.

The double `<colon>` in the target rule format is supported in BSD systems to allow more than

one target line containing the same target name to have commands associated with it. Since this is not functionality described in the SVID or XPG3 it has been allowed as an extension, but not mandated.

The default rules are provided with text specifying that the built-in rules shall be the same as if the listed set were used. The intent is that implementations should be able to use the rules without change, but will be allowed to alter them in ways that do not affect the primary behavior.

The best way to provide portable makefiles is to include all of the rules needed in the makefile itself. The rules provided use only features provided by other portions of this volume of POSIX.1-2008. The default rules include rules for optional commands in this volume of POSIX.1-2008. Only rules pertaining to commands that are provided are needed in the default set of an implementation.

One point of discussion was whether to drop the default rules list from this volume of POSIX.1-2008. They provide convenience, but do not enhance portability of applications. The prime benefit is in portability of users who wish to type *make command* and have the command build from a **command.c** file.

The historical **MAKESHELL** feature was omitted. In some implementations it is used to let a user override the shell to be used to run *make* commands. This was confusing; for a portable *make*, the shell should be chosen by the makefile writer or specified on the *make* command line and not by a user running *make*.

The *make* utilities in most historical implementations process the prerequisites of a target in left-to-right order, and the makefile format requires this. It supports the standard idiom used in many makefiles that produce *yacc* programs; for example:

```
foo: y.tab.o lex.o main.o
 $(CC) $(CFLAGS) -o $@ t.tab.o lex.o main.o
```

In this example, if *make* chose any arbitrary order, the **lex.o** might not be made with the correct **y.tab.h**. Although there may be better ways to express this relationship, it is widely used historically. Implementations that desire to update prerequisites in parallel should require an explicit extension to *make* or the makefile format to accomplish it, as described previously.

The algorithm for determining a new entry for target rules is partially unspecified. Some historical *makes* allow blank, empty, or comment lines within the collection of commands marked by leading <tab> characters. A conforming makefile must ensure that each command starts with a <tab>, but implementations are free to ignore blank, empty, and comment lines without triggering the start of a new entry.

The ASYNCHRONOUS EVENTS section includes having SIGTERM and SIGHUP, along with the more traditional SIGINT and SIGQUIT, remove the current target unless directed not to do so. SIGTERM and SIGHUP were added to parallel other utilities that have historically cleaned up their work as a result of these signals. When *make* receives any signal other than SIGQUIT, it is required to resend itself the signal it received so that it exits with a status that reflects the signal. The results from SIGQUIT are partially unspecified because, on systems that create **core** files upon receipt of SIGQUIT, the **core** from *make* would conflict with a **core** file from the command that was running when the SIGQUIT arrived. The main concern was to prevent damaged files from appearing up-to-date when *make* is rerun.

The **.PRECIOUS** special target was extended to affect all targets globally (by specifying no prerequisites). The **.IGNORE** and **.SILENT** special targets were extended to allow prerequisites; it was judged to be more useful in some cases to be able to turn off errors or echoing for a list of targets than for the entire makefile. These extensions to *make* in System V were made to match

historical practice from the BSD *make*.

Macros are not exported to the environment of commands to be run. This was never the case in any historical *make* and would have serious consequences. The environment is the same as the environment to *make* except that *MAKEFLAGS* and macros defined on the *make* command line are added.

Some implementations do not use *system()* for all command lines, as required by the portable makefile format; as a performance enhancement, they select lines without shell metacharacters for direct execution by *execve()*. There is no requirement that *system()* be used specifically, but merely that the same results be achieved. The metacharacters typically used to bypass the direct *execve()* execution have been any of:

```
= | ^ () ; & < > * ? [] : $ ` ' " \ \n
```

The default in some advanced versions of *make* is to group all the command lines for a target and execute them using a single shell invocation; the System V method is to pass each line individually to a separate shell. The single-shell method has the advantages in performance and the lack of a requirement for many continued lines. However, converting to this newer method has caused portability problems with many historical makefiles, so the behavior with the POSIX makefile is specified to be the same as that of System V. It is suggested that the special target **.ONESHELL** be used as an implementation extension to achieve the single-shell grouping for a target or group of targets.

Novice users of *make* have had difficulty with the historical need to start commands with a <tab>. Since it is often difficult to discern differences between <tab> and <space> characters on terminals or printed listings, confusing bugs can arise. In early proposals, an attempt was made to correct this problem by allowing leading <blank> characters instead of <tab> characters. However, implementors reported many makefiles that failed in subtle ways following this change, and it is difficult to implement a *make* that unambiguously can differentiate between macro and command lines. There is extensive historical practice of allowing leading <space> characters before macro definitions. Forcing macro lines into column 1 would be a significant backwards-compatibility problem for some makefiles. Therefore, historical practice was restored.

There is substantial variation in the handling of include lines by different implementations. However, there is enough commonality for the standard to be able to specify a minimum set of requirements that allow the feature to be used portably. Known variations have been explicitly called out as unspecified behavior in the description.

The System V dynamic dependency feature was not included. It would support:

```
cat: $$@.c
```

that would expand to;

```
cat: cat.c
```

This feature exists only in the new version of System V *make* and, while useful, is not in wide usage. This means that macros are expanded twice for prerequisites: once at makefile parse time and once at target update time.

Consideration was given to adding metarules to the POSIX *make*. This would make **%.o: %.c** the same as **.c.o:**. This is quite useful and available from some vendors, but it would cause too many changes to this *make* to support. It would have introduced rule chaining and new substitution rules. However, the rules for target names have been set to reserve the **'%'** and **'\"'** characters. These are traditionally used to implement metarules and quoting of target names, respectively. Implementors are strongly encouraged to use these characters only for these purposes.

A request was made to extend the suffix delimiter character from a <period> to any character. The metarules feature in newer *makes* solves this problem in a more general way. This volume of POSIX.1-2008 is staying with the more conservative historical definition.

The standard output format for the **-p** option is not described because it is primarily a debugging option and because the format is not generally useful to programs. In historical implementations the output is not suitable for use in generating makefiles. The **-p** format has been variable across historical implementations. Therefore, the definition of **-p** was only to provide a consistently named option for obtaining *make* script debugging information.

Some historical implementations have not cleared the suffix list with **-r**.

Implementations should be aware that some historical applications have intermixed *target\_name* and *macro=value* operands on the command line, expecting that all of the macros are processed before any of the targets are dealt with. Conforming applications do not do this, but some backwards-compatibility support may be warranted.

Empty inference rules are specified with a <semicolon> command rather than omitting all commands, as described in an early proposal. The latter case has no traditional meaning and is reserved for implementation extensions, such as in GNU *make*.

## FUTURE DIRECTIONS

None.

## SEE ALSO

[Chapter 2](#) (on page 2319), *ar*, *c99*, *get*, *lex*, *sccs*, *sh*, *yacc*

[XBD Section 6.1](#) (on page 125), [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 215)

XSH *exec*, *system()*

## CHANGE HISTORY

First released in Issue 2.

### Issue 5

The FUTURE DIRECTIONS section is added.

### Issue 6

This utility is marked as part of the Software Development Utilities option.

The Open Group Corrigendum U029/1 is applied, correcting a typographical error in the SPECIAL TARGETS section.

In the ENVIRONMENT VARIABLES section, the *PROJECTDIR* description is updated from “otherwise, the home directory of a user of that name is examined” to “otherwise, the value of *PROJECTDIR* is treated as a user name and that user’s initial working directory is examined”.

It is specified whether the command line is related to the makefile or to the *make* command, and the macro processing rules are updated to align with the IEEE P1003.2b draft standard.

The normative text is reworded to avoid use of the term “must” for application requirements.

PASC Interpretation 1003.2 #193 is applied.

### Issue 7

SD5-XCU-ERN-6 is applied, clarifying that Guideline 9 of the Utility Syntax Guidelines does not apply.

SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

Include lines in makefiles are introduced.



97874  
97875  
97876

Austin Group Interpretation 1003.1-2001 #131 is applied, changing the **Makefile Execution** section.

POSIX.1-2008, Technical Corrigendum 1, XCU/TC1-2008/0121 [257] is applied.

+

97877 **NAME**

97878       man — display system documentation

97879 **SYNOPSIS**97880       man [-k] *name*...97881 **DESCRIPTION**

97882       The *man* utility shall write information about each of the *name* operands. If *name* is the name of a  
 97883       standard utility, *man* at a minimum shall write a message describing the syntax used by the  
 97884       standard utility, its options, and operands. If more information is available, the *man* utility shall  
 97885       provide it in an implementation-defined manner.

97886       An implementation may provide information for values of *name* other than the standard utilities.  
 97887       Standard utilities that are listed as optional and that are not supported by the implementation  
 97888       either shall cause a brief message indicating that fact to be displayed or shall cause a full display  
 97889       of information as described previously.

97890 **OPTIONS**97891       The *man* utility shall conform to XBD [Section 12.2](#) (on page 215).

97892       The following option shall be supported:

97893       **-k**       Interpret *name* operands as keywords to be used in searching a utilities summary  
 97894       database that contains a brief purpose entry for each standard utility and write lines  
 97895       from the summary database that match any of the keywords. The keyword search shall  
 97896       produce results that are the equivalent of the output of the following command:

```
97897 grep -Ei '
97898 name
97899 name
97900 ...
97901 ' summary-database
```

97902       This assumes that the *summary-database* is a text file with a single entry per line; this  
 97903       organization is not required and the example using *grep -Ei* is merely illustrative of the  
 97904       type of search intended. The purpose entry to be included in the database shall consist  
 97905       of a terse description of the purpose of the utility.

97906 **OPERANDS**

97907       The following operand shall be supported:

97908       *name*       A keyword or the name of a standard utility. When **-k** is not specified and *name*  
 97909       does not represent one of the standard utilities, the results are unspecified.

97910 **STDIN**

97911       Not used.

97912 **INPUT FILES**

97913       None.

97914 **ENVIRONMENT VARIABLES**97915       The following environment variables shall affect the execution of *man*:

97916       **LANG**       Provide a default value for the internationalization variables that are unset or null.  
 97917       (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 97918       variables used to determine the values of locale categories.)

97919       **LC\_ALL**      If set to a non-empty string value, override the values of all the other  
 97920       internationalization variables.

97921 **LC\_CTYPE** Determine the locale for the interpretation of sequences of bytes of text data as  
 97922 characters (for example, single-byte as opposed to multi-byte characters in  
 97923 arguments and in the summary database). The value of *LC\_CTYPE* need not affect  
 97924 the format of the information written about the *name* operands.

97925 **LC\_MESSAGES**  
 97926 Determine the locale that should be used to affect the format and contents of  
 97927 diagnostic messages written to standard error and informative messages written to  
 97928 standard output.

97929 XSI **NLSPATH** Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

97930 **PAGER** Determine an output filtering command for writing the output to a terminal. Any  
 97931 string acceptable as a *command\_string* operand to the *sh -c* command shall be valid.  
 97932 When standard output is a terminal device, the reference page output shall be  
 97933 piped through the command. If the *PAGER* variable is null or not set, the command  
 97934 shall be either *more* or another paginator utility documented in the system  
 97935 documentation.

## 97936 ASYNCHRONOUS EVENTS

97937 Default.

## 97938 STDOUT

97939 The *man* utility shall write text describing the syntax of the utility *name*, its options and its  
 97940 operands, or, when *-k* is specified, lines from the summary database. The format of this text is  
 97941 implementation-defined.

## 97942 STDERR

97943 The standard error shall be used for diagnostic messages, and may also be used for  
 97944 informational messages of unspecified format.

## 97945 OUTPUT FILES

97946 None.

## 97947 EXTENDED DESCRIPTION

97948 None.

## 97949 EXIT STATUS

97950 The following exit values shall be returned:

97951 0 Successful completion.

97952 >0 An error occurred.

## 97953 CONSEQUENCES OF ERRORS

97954 Default.

## 97955 APPLICATION USAGE

97956 None.

## 97957 EXAMPLES

97958 None.

## 97959 RATIONALE

97960 It is recognized that the *man* utility is only of minimal usefulness as specified. The opinion of the  
 97961 standard developers was strongly divided as to how much or how little information *man* should  
 97962 be required to provide. They considered, however, that the provision of some portable way of  
 97963 accessing documentation would aid user portability. The arguments against a fuller specification  
 97964 were:

- Large quantities of documentation should not be required on a system that does not have excess disk space.
- The current manual system does not present information in a manner that greatly aids user portability.
- A “better help system” is currently an area in which vendors feel that they can add value to their POSIX implementations.

The `-f` option was considered, but due to implementation differences, it was not included in this volume of POSIX.1-2008.

The description was changed to be more specific about what has to be displayed for a utility. The standard developers considered it insufficient to allow a display of only the synopsis without giving a short description of what each option and operand does.

The “purpose” entry to be included in the database can be similar to the section title (less the numeric prefix) from this volume of POSIX.1-2008 for each utility. These titles are similar to those used in historical systems for this purpose.

See *mailx* for rationale concerning the default paginator.

The caveat in the *LC\_CTYPE* description was added because it is not a requirement that an implementation provide reference pages for all of its supported locales on each system; changing *LC\_CTYPE* does not necessarily translate the reference page into another language. This is equivalent to the current state of *LC\_MESSAGES* in POSIX.1-2008—locale-specific messages are not yet a requirement.

The historical *MANPATH* variable is not included in POSIX because no attempt is made to specify naming conventions for reference page files, nor even to mandate that they are files at all. On some implementations they could be a true database, a hypertext file, or even fixed strings within the *man* executable. The standard developers considered the portability of reference pages to be outside their scope of work. However, users should be aware that *MANPATH* is implemented on a number of historical systems and that it can be used to tailor the search pattern for reference pages from the various categories (utilities, functions, file formats, and so on) when the system administrator reveals the location and conventions for reference pages on the system.

The keyword search can rely on at least the text of the section titles from these utility descriptions, and the implementation may add more keywords. The term “section titles” refers to the strings such as:

```
man - Display system documentation
ps - Report process status
```

## FUTURE DIRECTIONS

None.

## SEE ALSO

[more](#)

XBD [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 215)

## CHANGE HISTORY

First released in Issue 4.

98006 **Issue 5**

98007 The FUTURE DIRECTIONS section is added.

98008 **Issue 7**

98009 Austin Group Interpretation 1003.1-2001 #108 is applied, clarifying that informational messages  
98010 may appear on standard error.

98011 **NAME**

98012       mesg — permit or deny messages

98013 **SYNOPSIS**98014       mesg [*y*|*n*]98015 **DESCRIPTION**

98016       The *mesg* utility shall control whether other users are allowed to send messages via *write*, *talk*, or  
 98017       other utilities to a terminal device. The terminal device affected shall be determined by searching  
 98018       for the first terminal in the sequence of devices associated with standard input, standard output,  
 98019       and standard error, respectively. With no arguments, *mesg* shall report the current state without  
 98020       changing it. Processes with appropriate privileges may be able to send messages to the terminal  
 98021       independent of the current state.

98022 **OPTIONS**

98023       None.

98024 **OPERANDS**

98025       The following operands shall be supported in the POSIX locale:

98026       *y*               Grant permission to other users to send messages to the terminal device.98027       *n*               Deny permission to other users to send messages to the terminal device.98028 **STDIN**

98029       Not used.

98030 **INPUT FILES**

98031       None.

98032 **ENVIRONMENT VARIABLES**98033       The following environment variables shall affect the execution of *mesg*:

98034       *LANG*           Provide a default value for the internationalization variables that are unset or null.  
 98035                       (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 98036                       variables used to determine the values of locale categories.)

98037       *LC\_ALL*       If set to a non-empty string value, override the values of all the other  
 98038                       internationalization variables.

98039       *LC\_CTYPE*   Determine the locale for the interpretation of sequences of bytes of text data as  
 98040                       characters (for example, single-byte as opposed to multi-byte characters in  
 98041                       arguments).

98042       *LC\_MESSAGES*

98043                       Determine the locale that should be used to affect the format and contents of  
 98044                       diagnostic messages written (by *mesg*) to standard error.

98045 XSI       *NLSPATH*   Determine the location of message catalogs for the processing of *LC\_MESSAGES*.98046 **ASYNCHRONOUS EVENTS**

98047       Default.

98048 **STDOUT**98049       If no operand is specified, *mesg* shall display the current terminal state in an unspecified format.98050 **STDERR**

98051       The standard error shall be used only for diagnostic messages.

**98052 OUTPUT FILES**

98053 None.

**98054 EXTENDED DESCRIPTION**

98055 None.

**98056 EXIT STATUS**

98057 The following exit values shall be returned:

98058 0 Receiving messages is allowed.

98059 1 Receiving messages is not allowed.

98060 >1 An error occurred.

**98061 CONSEQUENCES OF ERRORS**

98062 Default.

**98063 APPLICATION USAGE**

98064 The mechanism by which the message status of the terminal is changed is unspecified.  
98065 Therefore, unspecified actions may cause the status of the terminal to change after *mesg* has  
98066 successfully completed. These actions may include, but are not limited to: another invocation of  
98067 the *mesg* utility, login procedures; invocation of the *stty* utility, invocation of the *chmod* utility or  
98068 *chmod()* function, and so on.

**98069 EXAMPLES**

98070 None.

**98071 RATIONALE**

98072 The terminal changed by *mesg* is that associated with the standard input, output, or error, rather  
98073 than the controlling terminal for the session. This is because users logged in more than once  
98074 should be able to change any of their login terminals without having to stop the job running in  
98075 those sessions. This is not a security problem involving the terminals of other users because  
98076 appropriate privileges would be required to affect the terminal of another user.

98077 The method of checking each of the first three file descriptors in sequence until a terminal is  
98078 found was adopted from System V.

98079 The file */dev/tty* is not specified for the terminal device because it was thought to be too  
98080 restrictive. Typical environment changes for the *n* operand are that write permissions are  
98081 removed for *others* and *group* from the appropriate device. It was decided to leave the actual  
98082 description of what is done as unspecified because of potential differences between  
98083 implementations.

98084 The format for standard output is unspecified because of differences between historical  
98085 implementations. This output is generally not useful to shell scripts (they can use the exit status),  
98086 so exact parsing of the output is unnecessary.

**98087 FUTURE DIRECTIONS**

98088 None.

**98089 SEE ALSO**

98090 *talk*, *write*

98091 XBD Chapter 8 (on page 173)

98092 **CHANGE HISTORY**

98093 First released in Issue 2.

98094 **Issue 6**

98095 This utility is marked as part of the User Portability Utilities option.

98096 **Issue 7**98097 The *mesg* utility is moved from the User Portability Utilities option to the Base. User Portability  
98098 Utilities is now an option for interactive utilities.



98099 **NAME**98100 `mkdir` — make directories98101 **SYNOPSIS**98102 `mkdir [-p] [-m mode] dir...`98103 **DESCRIPTION**98104 The *mkdir* utility shall create the directories specified by the operands, in the order specified.98105 For each *dir* operand, the *mkdir* utility shall perform actions equivalent to the *mkdir()* function  
98106 defined in the System Interfaces volume of POSIX.1-2008, called with the following arguments:

- 98107 1. The *dir* operand is used as the *path* argument.
- 98108 2. The value of the bitwise-inclusive OR of `S_IRWXU`, `S_IRWXG`, and `S_IRWXO` is used as  
98109 the *mode* argument. (If the `-m` option is specified, the value of the *mkdir()* *mode* argument  
98110 is unspecified, but the directory shall at no time have permissions less restrictive than the  
98111 `-m mode` option-argument.)

98112 **OPTIONS**98113 The *mkdir* utility shall conform to XBD [Section 12.2](#) (on page 215).

98114 The following options shall be supported:

98115 `-m mode` Set the file permission bits of the newly-created directory to the specified *mode*  
98116 value. The *mode* option-argument shall be the same as the *mode* operand defined  
98117 for the *chmod* utility. In the *symbolic\_mode* strings, the *op* characters '+' and '-'  
98118 shall be interpreted relative to an assumed initial mode of *a=rwx*; '+' shall add  
98119 permissions to the default mode, '-' shall delete permissions from the default  
98120 mode.

98121 `-p` Create any missing intermediate pathname components.

98122 For each *dir* operand that does not name an existing directory, before performing  
98123 the actions described in the DESCRIPTION above, the *mkdir* utility shall create any  
98124 pathname components of the path prefix of *dir* that do not name an existing  
98125 directory by performing actions equivalent to first calling the *mkdir()* function with  
98126 the following arguments:

- 98127 1. A pathname naming the missing pathname component, ending with a  
98128 trailing <slash> character, as the *path* argument
- 98129 2. The value zero as the *mode* argument

98130 and then calling the *chmod()* function with the following arguments:

- 98131 1. The same *path* argument as in the *mkdir()* call
- 98132 2. The value `(S_IWUSR|S_IXUSR|~filemask)&0777` as the *mode*  
98133 argument, where *filemask* is the file mode creation mask of the process (see  
98134 XSH [umask\(\)](#))

98135 Each *dir* operand that names an existing directory shall be ignored without error.98136 **OPERANDS**

98137 The following operand shall be supported:

98138 *dir* A pathname of a directory to be created.

98139 **STDIN**

98140 Not used.

98141 **INPUT FILES**

98142 None.

98143 **ENVIRONMENT VARIABLES**98144 The following environment variables shall affect the execution of *mkdir*:

98145 *LANG* Provide a default value for the internationalization variables that are unset or null.  
 98146 (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 98147 variables used to determine the values of locale categories.)

98148 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
 98149 internationalization variables.

98150 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
 98151 characters (for example, single-byte as opposed to multi-byte characters in  
 98152 arguments).

98153 *LC\_MESSAGES*

98154 Determine the locale that should be used to affect the format and contents of  
 98155 diagnostic messages written to standard error.

98156 *XSI* *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

98157 **ASYNCHRONOUS EVENTS**

98158 Default.

98159 **STDOUT**

98160 Not used.

98161 **STDERR**

98162 The standard error shall be used only for diagnostic messages.

98163 **OUTPUT FILES**

98164 None.

98165 **EXTENDED DESCRIPTION**

98166 None.

98167 **EXIT STATUS**

98168 The following exit values shall be returned:

98169 0 All the specified directories were created successfully or the **-p** option was specified and all  
 98170 the specified directories now exist.

98171 &gt;0 An error occurred.

98172 **CONSEQUENCES OF ERRORS**

98173 Default.

**APPLICATION USAGE**

The default file mode for directories is *a=rwx* (777 on most systems) with selected permissions removed in accordance with the file mode creation mask. For intermediate pathname components created by *mkdir*, the mode is the default modified by *u+rwx* so that the subdirectories can always be created regardless of the file mode creation mask; if different ultimate permissions are desired for the intermediate directories, they can be changed afterwards with *chmod*.

Note that some of the requested directories may have been created even if an error occurs.

**EXAMPLES**

None.

**RATIONALE**

The System V *-m* option was included to control the file mode.

The System V *-p* option was included to create any needed intermediate directories and to complement the functionality provided by *rmdir* for removing directories in the path prefix as they become empty. Because no error is produced if any path component already exists, the *-p* option is also useful to ensure that a particular directory exists.

The functionality of *mkdir* is described substantially through a reference to the *mkdir()* function in the System Interfaces volume of POSIX.1-2008. For example, by default, the mode of the directory is affected by the file mode creation mask in accordance with the specified behavior of the *mkdir()* function. In this way, there is less duplication of effort required for describing details of the directory creation.

**FUTURE DIRECTIONS**

None.

**SEE ALSO**

*chmod*, *rm*, *rmdir*, *umask*

XBD [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 215)

XSH *mkdir()*, *umask()*

**CHANGE HISTORY**

First released in Issue 2.

**Issue 5**

The FUTURE DIRECTIONS section is added.

**Issue 7**

SD5-XCU-ERN-56 is applied, aligning the *-m* option with the IEEE P1003.2b draft standard to clarify an ambiguity.

SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

POSIX.1-2008, Technical Corrigendum 1, XCU/TC1-2008/0122 [161] is applied.

+

**NAME**

mkfifo — make FIFO special files

**SYNOPSIS**

mkfifo [-m *mode*] *file*...

**DESCRIPTION**

The *mkfifo* utility shall create the FIFO special files specified by the operands, in the order specified.

For each *file* operand, the *mkfifo* utility shall perform actions equivalent to the *mkfifo()* function defined in the System Interfaces volume of POSIX.1-2008, called with the following arguments:

1. The *file* operand is used as the *path* argument.
2. The value of the bitwise-inclusive OR of S\_IRUSR, S\_IWUSR, S\_IRGRP, S\_IWGRP, S\_IROTH, and S\_IWOTH is used as the *mode* argument. (If the -m option is specified, the value of the *mkfifo()* *mode* argument is unspecified, but the FIFO shall at no time have permissions less restrictive than the -m *mode* option-argument.)

**OPTIONS**

The *mkfifo* utility shall conform to XBD [Section 12.2](#) (on page 215).

The following option shall be supported:

-m *mode* Set the file permission bits of the newly-created FIFO to the specified *mode* value. The *mode* option-argument shall be the same as the *mode* operand defined for the *chmod* utility. In the *symbolic\_mode* strings, the *op* characters '+' and '-' shall be interpreted relative to an assumed initial mode of *a=rw*.

**OPERANDS**

The following operand shall be supported:

*file* A pathname of the FIFO special file to be created.

**STDIN**

Not used.

**INPUT FILES**

None.

**ENVIRONMENT VARIABLES**

The following environment variables shall affect the execution of *mkfifo*:

*LANG* Provide a default value for the internationalization variables that are unset or null. (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization variables used to determine the values of locale categories.)

*LC\_ALL* If set to a non-empty string value, override the values of all the other internationalization variables.

*LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as characters (for example, single-byte as opposed to multi-byte characters in arguments).

*LC\_MESSAGES*

Determine the locale that should be used to affect the format and contents of diagnostic messages written to standard error.

|       |     |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|-------|-----|-------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 98251 | XSI | <b>NLSPATH</b>                | Determine the location of message catalogs for the processing of <i>LC_MESSAGES</i> .                                                                                                                                                                                                                                                                                                                                                              |
| 98252 |     | <b>ASYNCHRONOUS EVENTS</b>    |                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 98253 |     |                               | Default.                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| 98254 |     | <b>STDOUT</b>                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 98255 |     |                               | Not used.                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 98256 |     | <b>STDERR</b>                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 98257 |     |                               | The standard error shall be used only for diagnostic messages.                                                                                                                                                                                                                                                                                                                                                                                     |
| 98258 |     | <b>OUTPUT FILES</b>           |                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 98259 |     |                               | None.                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 98260 |     | <b>EXTENDED DESCRIPTION</b>   |                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 98261 |     |                               | None.                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 98262 |     | <b>EXIT STATUS</b>            |                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 98263 |     |                               | The following exit values shall be returned:                                                                                                                                                                                                                                                                                                                                                                                                       |
| 98264 |     | 0                             | All the specified FIFO special files were created successfully.                                                                                                                                                                                                                                                                                                                                                                                    |
| 98265 |     | >0                            | An error occurred.                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| 98266 |     | <b>CONSEQUENCES OF ERRORS</b> |                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 98267 |     |                               | Default.                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| 98268 |     | <b>APPLICATION USAGE</b>      |                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 98269 |     |                               | None.                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 98270 |     | <b>EXAMPLES</b>               |                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 98271 |     |                               | None.                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 98272 |     | <b>RATIONALE</b>              |                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 98273 |     |                               | This utility was added to permit shell applications to create FIFO special files.                                                                                                                                                                                                                                                                                                                                                                  |
| 98274 |     |                               | The <b>-m</b> option was added to control the file mode, for consistency with the similar functionality provided by the <i>mkdir</i> utility.                                                                                                                                                                                                                                                                                                      |
| 98276 |     |                               | Early proposals included a <b>-p</b> option similar to the <i>mkdir -p</i> option that created intermediate directories leading up to the FIFO specified by the final component. This was removed because it is not commonly needed and is not common practice with similar utilities.                                                                                                                                                             |
| 98277 |     |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 98278 |     |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 98279 |     |                               | The functionality of <i>mkfifo</i> is described substantially through a reference to the <i>mkfifo()</i> function in the System Interfaces volume of POSIX.1-2008. For example, by default, the mode of the FIFO file is affected by the file mode creation mask in accordance with the specified behavior of the <i>mkfifo()</i> function. In this way, there is less duplication of effort required for describing details of the file creation. |
| 98280 |     |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 98281 |     |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 98282 |     |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 98283 |     |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 98284 |     | <b>FUTURE DIRECTIONS</b>      |                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 98285 |     |                               | None.                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 98286 |     | <b>SEE ALSO</b>               |                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 98287 |     |                               | <i>chmod</i> , <i>umask</i>                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 98288 |     |                               | XBD <a href="#">Chapter 8</a> (on page 173), <a href="#">Section 12.2</a> (on page 215)                                                                                                                                                                                                                                                                                                                                                            |
| 98289 |     |                               | XSH <a href="#">mkfifo()</a>                                                                                                                                                                                                                                                                                                                                                                                                                       |

|       |                                                                                                |
|-------|------------------------------------------------------------------------------------------------|
| 98290 | <b>CHANGE HISTORY</b>                                                                          |
| 98291 | First released in Issue 3.                                                                     |
| 98292 | <b>Issue 6</b>                                                                                 |
| 98293 | The <b>-m</b> option is aligned with the IEEE P1003.2b draft standard to clarify an ambiguity. |

98294 **NAME**98295 **more** — display files on a page-by-page basis98296 **SYNOPSIS**98297 UP **more** [**-ceisu**] [**-n** *number*] [**-p** *command*] [**-t** *tagstring*] [*file...*]98298 **DESCRIPTION**

98299 The *more* utility shall read files and either write them to the terminal on a page-by-page basis or  
 98300 filter them to standard output. If standard output is not a terminal device, all input files shall be  
 98301 copied to standard output in their entirety, without modification, except as specified for the **-s**  
 98302 option. If standard output is a terminal device, the files shall be written a number of lines (one  
 98303 screenful) at a time under the control of user commands. See the EXTENDED DESCRIPTION  
 98304 section.

98305 Certain block-mode terminals do not have all the capabilities necessary to support the complete  
 98306 *more* definition; they are incapable of accepting commands that are not terminated with a  
 98307 <newline>. Implementations that support such terminals shall provide an operating mode to  
 98308 *more* in which all commands can be terminated with a <newline> on those terminals. This mode:

- 98309 • Shall be documented in the system documentation
- 98310 • Shall, at invocation, inform the user of the terminal deficiency that requires the <newline>  
 98311 usage and provide instructions on how this warning can be suppressed in future  
 98312 invocations
- 98313 • Shall not be required for implementations supporting only fully capable terminals
- 98314 • Shall not affect commands already requiring <newline> characters
- 98315 • Shall not affect users on the capable terminals from using *more* as described in this volume  
 98316 of POSIX.1-2008

98317 **OPTIONS**

98318 The *more* utility shall conform to XBD [Section 12.2](#) (on page 215), except that '+' may be  
 98319 recognized as an option delimiter as well as '-'.

98320 The following options shall be supported:

- 98321 **-c** If a screen is to be written that has no lines in common with the current screen, or  
 98322 *more* is writing its first screen, *more* shall not scroll the screen, but instead shall  
 98323 redraw each line of the screen in turn, from the top of the screen to the bottom. In  
 98324 addition, if *more* is writing its first screen, the screen shall be cleared. This option  
 98325 may be silently ignored on devices with insufficient terminal capabilities.
- 98326 **-e** Exit immediately after writing the last line of the last file in the argument list; see  
 98327 the EXTENDED DESCRIPTION section.
- 98328 **-i** Perform pattern matching in searches without regard to case; see XBD [Section 9.2](#)  
 98329 (on page 182).
- 98330 **-n** *number* Specify the number of lines per screenful. The *number* argument is a positive  
 98331 decimal integer. The **-n** option shall override any values obtained from any other  
 98332 source.
- 98333 **-p** *command* Each time a screen from a new file is displayed or redisplayed (including as a  
 98334 result of *more* commands; for example, **:p**), execute the *more* command(s) in the  
 98335 command arguments in the order specified, as if entered by the user after the first  
 98336 screen has been displayed. No intermediate results shall be displayed (that is, if the  
 98337 command is a movement to a screen different from the normal first screen, only the

98338 screen resulting from the command shall be displayed.) If any of the commands  
 98339 fail for any reason, an informational message to this effect shall be written, and no  
 98340 further commands specified using the **-p** option shall be executed for this file.

98341 **-s** Behave as if consecutive empty lines were a single empty line.

98342 **-t tagstring** Write the screenful of the file containing the tag named by the *tagstring* argument.  
 98343 See the *ctags* utility. The tags feature represented by **-t tagstring** and the **:t**  
 98344 command is optional. It shall be provided on any system that also provides a  
 98345 conforming implementation of *ctags*; otherwise, the use of **-t** produces undefined  
 98346 results.

98347 The filename resulting from the **-t** option shall be logically added as a prefix to the  
 98348 list of command line files, as if specified by the user. If the tag named by the  
 98349 *tagstring* argument is not found, it shall be an error, and *more* shall take no further  
 98350 action.

98351 If the tag specifies a line number, the first line of the display shall contain the  
 98352 beginning of that line. If the tag specifies a pattern, the first line of the display shall  
 98353 contain the beginning of the matching text from the first line of the file that  
 98354 contains that pattern. If the line does not exist in the file or matching text is not  
 98355 found, an informational message to this effect shall be displayed, and *more* shall  
 98356 display the default screen as if **-t** had not been specified.

98357 If both the **-t tagstring** and **-p command** options are given, the **-t tagstring** shall be  
 98358 processed first; that is, the file and starting line for the display shall be as specified  
 98359 by **-t**, and then the **-p more** command shall be executed. If the line (matching text)  
 98360 specified by the **-t** command does not exist (is not found), no **-p more** command  
 98361 shall be executed for this file at any time.

98362 **-u** Treat a <backspace> as a printable control character, displayed as an  
 98363 implementation-defined character sequence (see the EXTENDED DESCRIPTION  
 98364 section), suppressing backspacing and the special handling that produces  
 98365 underlined or standout mode text on some terminal types. Also, do not ignore a  
 98366 <carriage-return> at the end of a line.

## 98367 OPERANDS

98368 The following operand shall be supported:

98369 *file* A pathname of an input file. If no *file* operands are specified, the standard input  
 98370 shall be used. If a *file* is *'-'*, the standard input shall be read at that point in the  
 98371 sequence.

## 98372 STDIN

98373 The standard input shall be used only if no *file* operands are specified, or if a *file* operand is *'-'*.

## 98374 INPUT FILES

98375 The input files being examined shall be text files. If standard output is a terminal, standard error  
 98376 shall be used to read commands from the user. If standard output is a terminal, standard error is  
 98377 not readable, and command input is needed, *more* may attempt to obtain user commands from  
 98378 the controlling terminal (for example, */dev/tty*); otherwise, *more* shall terminate with an error  
 98379 indicating that it was unable to read user commands. If standard output is not a terminal, no  
 98380 error shall result if standard error cannot be opened for reading.



98381 **ENVIRONMENT VARIABLES**98382 The following environment variables shall affect the execution of *more*:

98383 *COLUMNS* Override the system-selected horizontal display line size. See XBD [Chapter 8](#) (on  
98384 page 173) for valid values and results when it is unset or null.

98385 *EDITOR* Used by the *v* command to select an editor. See the EXTENDED DESCRIPTION  
98386 section.

98387 *LANG* Provide a default value for the internationalization variables that are unset or null.  
98388 (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
98389 variables used to determine the values of locale categories.)

98390 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
98391 internationalization variables.

98392 *LC\_COLLATE*  
98393 Determine the locale for the behavior of ranges, equivalence classes, and multi-  
98394 character collating elements within regular expressions.

98395 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
98396 characters (for example, single-byte as opposed to multi-byte characters in  
98397 arguments and input files) and the behavior of character classes within regular  
98398 expressions.

98399 *LC\_MESSAGES*  
98400 Determine the locale that should be used to affect the format and contents of  
98401 diagnostic messages written to standard error and informative messages written to  
98402 standard output.

98403 XSI *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

98404 *LINES* Override the system-selected vertical screen size, used as the number of lines in a  
98405 screenful. See XBD [Chapter 8](#) (on page 173) for valid values and results when it is  
98406 unset or null. The *-n* option shall take precedence over the *LINES* variable for  
98407 determining the number of lines in a screenful.

98408 *MORE* Determine a string containing options described in the OPTIONS section preceded  
98409 with <hyphen> characters and <blank>-separated as on the command line. Any  
98410 command line options shall be processed after those in the *MORE* variable, as if  
98411 the command line were:

98412 *more \$MORE options operands*

98413 The *MORE* variable shall take precedence over the *TERM* and *LINES* variables for  
98414 determining the number of lines in a screenful.

98415 *TERM* Determine the name of the terminal type. If this variable is unset or null, an  
98416 unspecified default terminal type is used.

98417 **ASYNCHRONOUS EVENTS**

98418 Default.

98419 **STDOUT**

98420 The standard output shall be used to write the contents of the input files.

**STDERR**

The standard error shall be used for diagnostic messages and user commands (see the INPUT FILES section), and, if standard output is a terminal device, to write a prompting string. The prompting string shall appear on the screen line below the last line of the file displayed in the current screenful. The prompt shall contain the name of the file currently being examined and shall contain an end-of-file indication and the name of the next file, if any, when prompting at the end-of-file. If an error or informational message is displayed, it is unspecified whether it is contained in the prompt. If it is not contained in the prompt, it shall be displayed and then the user shall be prompted for a continuation character, at which point another message or the user prompt may be displayed. The prompt is otherwise unspecified. It is unspecified whether informational messages are written for other user commands.

**OUTPUT FILES**

None.

**EXTENDED DESCRIPTION**

The following section describes the behavior of *more* when the standard output is a terminal device. If the standard output is not a terminal device, no options other than *-s* shall have any effect, and all input files shall be copied to standard output otherwise unmodified, at which time *more* shall exit without further action.

The number of lines available per screen shall be determined by the *-n* option, if present, or by examining values in the environment (see the ENVIRONMENT VARIABLES section). If neither method yields a number, an unspecified number of lines shall be used.

The maximum number of lines written shall be one less than this number, because the screen line after the last line written shall be used to write a user prompt and user input. If the number of lines in the screen is less than two, the results are undefined. It is unspecified whether user input is permitted to be longer than the remainder of the single line where the prompt has been written.

The number of columns available per line shall be determined by examining values in the environment (see the ENVIRONMENT VARIABLES section), with a default value as described in XBD Chapter 8 (on page 173).

Lines that are longer than the display shall be folded; the length at which folding occurs is unspecified, but should be appropriate for the output device. Folding may occur between glyphs of single characters that take up multiple display columns.

When standard output is a terminal and *-u* is not specified, *more* shall treat <backspace> and <carriage-return> characters specially:

- A character, followed first by a sequence of *n* <backspace> characters (where *n* is the same as the number of column positions that the character occupies), then by *n* <underscore> characters ('\_'), shall cause that character to be written as underlined text, if the terminal type supports that. The *n* <underscore> characters, followed first by *n* <backspace> characters, then any character with *n* column positions, shall also cause that character to be written as underlined text, if the terminal type supports that.
- A sequence of *n* <backspace> characters (where *n* is the same as the number of column positions that the previous character occupies) that appears between two identical printable characters shall cause the first of those two characters to be written as emboldened text (that is, visually brighter, standout mode, or inverse-video mode), if the terminal type supports that, and the second to be discarded. Immediately subsequent occurrences of <backspace>/character pairs for that same character shall also be discarded. (For example, the sequence "a\ba\ba\ba" is interpreted as a single

emboldened 'a'.)

- The *more* utility shall logically discard all other <backspace> characters from the line as well as the character which precedes them, if any.
- A <carriage-return> at the end of a line shall be ignored, rather than being written as a non-printable character, as described in the next paragraph.

It is implementation-defined how other non-printable characters are written. Implementations should use the same format that they use for the *ex* **print** command; see the OPTIONS section within the *ed* utility. It is unspecified whether a multi-column character shall be separated if it crosses a display line boundary; it shall not be discarded. The behavior is unspecified if the number of columns on the display is less than the number of columns any single character in the line being displayed would occupy.

When each new file is displayed (or redisplayed), *more* shall write the first screen of the file. Once the initial screen has been written, *more* shall prompt for a user command. If the execution of the user command results in a screen that has lines in common with the current screen, and the device has sufficient terminal capabilities, *more* shall scroll the screen; otherwise, it is unspecified whether the screen is scrolled or redrawn.

For all files but the last (including standard input if no file was specified, and for the last file as well, if the **-e** option was not specified), when *more* has written the last line in the file, *more* shall prompt for a user command. This prompt shall contain the name of the next file as well as an indication that *more* has reached end-of-file. If the user command is **f**, <control>-F, <space>, **j**, <newline>, **d**, <control>-D, or **s**, *more* shall display the next file. Otherwise, if displaying the last file, *more* shall exit. Otherwise, *more* shall execute the user command specified.

Several of the commands described in this section display a previous screen from the input stream. In the case that text is being taken from a non-rewindable stream, such as a pipe, it is implementation-defined how much backwards motion is supported. If a command cannot be executed because of a limitation on backwards motion, an error message to this effect shall be displayed, the current screen shall not change, and the user shall be prompted for another command.

If a command cannot be performed because there are insufficient lines to display, *more* shall alert the terminal. If a command cannot be performed because there are insufficient lines to display or a **/** command fails: if the input is the standard input, the last screen in the file may be displayed; otherwise, the current file and screen shall not change, and the user shall be prompted for another command.

The interactive commands in the following sections shall be supported. Some commands can be preceded by a decimal integer, called *count* in the following descriptions. If not specified with the command, *count* shall default to 1. In the following descriptions, *pattern* is a basic regular expression, as described in XBD [Section 9.3](#) (on page 183). The term “examine” is historical usage meaning “open the file for viewing”; for example, *more* **foo** would be expressed as examining file **foo**.

In the following descriptions, unless otherwise specified, *line* is a line in the *more* display, not a line from the file being examined.

In the following descriptions, the *current position* refers to two things:

1. The position of the current line on the screen
2. The line number (in the file) of the current line on the screen

Usually, the line on the screen corresponding to the current position is the third line on the

screen. If this is not possible (there are fewer than three lines to display or this is the first page of the file, or it is the last page of the file), then the current position is either the first or last line on the screen as described later.

## Help

*Synopsis:*     **h**

Write a summary of these commands and other implementation-defined commands. The behavior shall be as if the *more* utility were executed with the **-e** option on a file that contained the summary information. The user shall be prompted as described earlier in this section when end-of-file is reached. If the user command is one of those specified to continue to the next file, *more* shall return to the file and screen state from which the **h** command was executed.

## Scroll Forward One Screenful

*Synopsis:*     **[count]f**  
                 **[count]<control>-F**

Scroll forward *count* lines, with a default of one screenful. If *count* is more than the screen size, only the final screenful shall be written.

## Scroll Backward One Screenful

*Synopsis:*     **[count]b**  
                 **[count]<control>-B**

Scroll backward *count* lines, with a default of one screenful (see the **-n** option). If *count* is more than the screen size, only the final screenful shall be written.

## Scroll Forward One Line

*Synopsis:*     **[count]<space>**  
                 **[count]j**  
                 **[count]<newline>**

Scroll forward *count* lines. The default *count* for the **<space>** shall be one screenful; for **j** and **<newline>**, one line. The entire *count* lines shall be written, even if *count* is more than the screen size.

## Scroll Backward One Line

*Synopsis:*     **[count]k**

Scroll backward *count* lines. The entire *count* lines shall be written, even if *count* is more than the screen size.

## Scroll Forward One Half Screenful

*Synopsis:*     **[count]d**  
                 **[count]<control>-D**

Scroll forward *count* lines, with a default of one half of the screen size. If *count* is specified, it shall become the new default for subsequent **d**, **<control>-D**, and **u** commands.

**Skip Forward One Line**

*Synopsis:*     [*count*]s

Display the screenful beginning with the line *count* lines after the last line on the current screen. If *count* would cause the current position to be such that less than one screenful would be written, the last screenful in the file shall be written.

**Scroll Backward One Half Screenful**

*Synopsis:*     [*count*]u  
              [*count*]<control>-U

Scroll backward *count* lines, with a default of one half of the screen size. If *count* is specified, it shall become the new default for subsequent **d**, <control>-D, **u**, and <control>-U commands. The entire *count* lines shall be written, even if *count* is more than the screen size.

**Go to Beginning of File**

*Synopsis:*     [*count*]g

Display the screenful beginning with line *count*.

**Go to End-of-File**

*Synopsis:*     [*count*]G

If *count* is specified, display the screenful beginning with the line *count*. Otherwise, display the last screenful of the file.

**Refresh the Screen**

*Synopsis:*     r  
              <control>-L

Refresh the screen.

**Discard and Refresh**

*Synopsis:*     R

Refresh the screen, discarding any buffered input. If the current file is non-seekable, buffered input shall not be discarded and the **R** command shall be equivalent to the **r** command.

**Mark Position**

*Synopsis:*     m*letter*

Mark the current position with the letter named by *letter*, where *letter* represents the name of one of the lowercase letters of the portable character set. When a new file is examined, all marks may be lost.

**Return to Mark**

*Synopsis:*     ' *letter*

Return to the position that was previously marked with the letter named by *letter*, making that line the current position.

**Return to Previous Position**

*Synopsis:*     ''

Return to the position from which the last large movement command was executed (where a "large movement" is defined as any movement of more than a screenful of lines). If no such movements have been made, return to the beginning of the file.

**Search Forward for Pattern**

*Synopsis:*     [*count*]/[!]*pattern*<newline>

Display the screenful beginning with the *count*th line containing the pattern. The search shall start after the first line currently displayed. The null regular expression ('/' followed by a <newline>) shall repeat the search using the previous regular expression, with a default *count*. If the character '!' is included, the matching lines shall be those that do not contain the *pattern*. If no match is found for the *pattern*, a message to that effect shall be displayed.

**Search Backward for Pattern**

*Synopsis:*     [*count*?][!]*pattern*<newline>

Display the screenful beginning with the *count*th previous line containing the pattern. The search shall start on the last line before the first line currently displayed. The null regular expression ('?' followed by a <newline>) shall repeat the search using the previous regular expression, with a default *count*. If the character '!' is included, matching lines shall be those that do not contain the *pattern*. If no match is found for the *pattern*, a message to that effect shall be displayed.

**Repeat Search**

*Synopsis:*     [*count*]n

Repeat the previous search for *count*th line containing the last *pattern* (or not containing the last *pattern*, if the previous search was "/" or "?").

**Repeat Search in Reverse**

*Synopsis:*     [*count*]N

Repeat the search in the opposite direction of the previous search for the *count*th line containing the last *pattern* (or not containing the last *pattern*, if the previous search was "/" or "?").

**Examine New File**

*Synopsis:*        :e [*filename*]<newline>

Examine a new file. If the *filename* argument is not specified, the current file (see the :n and :p commands below) shall be re-examined. The *filename* shall be subjected to the process of shell word expansions (see [Section 2.6](#), on page 2327); if more than a single pathname results, the effects are unspecified. If *filename* is a <number-sign> ('#'), the previously examined file shall be re-examined. If *filename* is not accessible for any reason (including that it is a non-seekable file), an error message to this effect shall be displayed and the current file and screen shall not change.

**Examine Next File**

*Synopsis:*        [*count*]:n

Examine the next file. If a number *count* is specified, the *count*th next file shall be examined. If *filename* refers to a non-seekable file, the results are unspecified.

**Examine Previous File**

*Synopsis:*        [*count*]:p

Examine the previous file. If a number *count* is specified, the *count*th previous file shall be examined. If *filename* refers to a non-seekable file, the results are unspecified.

**Go to Tag**

*Synopsis:*        :t *tagstring*<newline>

If the file containing the tag named by the *tagstring* argument is not the current file, examine the file, as if the :e command was executed with that file as the argument. Otherwise, or in addition, display the screenful beginning with the tag, as described for the -t option (see the OPTIONS section). If the *ctags* utility is not supported by the system, the use of :t produces undefined results.

**Invoke Editor**

*Synopsis:*        v

Invoke an editor to edit the current file being examined. If standard input is being examined, the results are unspecified. The name of the editor shall be taken from the environment variable *EDITOR*, or shall default to *vi*. If the last pathname component in *EDITOR* is either *vi* or *ex*, the editor shall be invoked with a -c *linenumber* command line argument, where *linenumber* is the line number of the file line containing the display line currently displayed as the first line of the screen. It is implementation-defined whether line-setting options are passed to editors other than *vi* and *ex*.

When the editor exits, *more* shall resume with the same file and screen as when the editor was invoked.

**Display Position**

*Synopsis:*       =

                  <control>-G

Write a message for which the information references the first byte of the line after the last line of the file on the screen. This message shall include the name of the file currently being examined, its number relative to the total number of files there are to examine, the line number in the file, the byte number and the total bytes in the file, and what percentage of the file precedes the current position. If *more* is reading from standard input, or the file is shorter than a single screen, the line number, the byte number, the total bytes, and the percentage need not be written.

**Quit**

*Synopsis:*       q

                  :q

                  ZZ

Exit *more*.

**EXIT STATUS**

The following exit values shall be returned:

- 0   Successful completion.
- >0   An error occurred.

**CONSEQUENCES OF ERRORS**

If an error is encountered accessing a file when using the **:n** command, *more* shall attempt to examine the next file in the argument list, but the final exit status shall be affected. If an error is encountered accessing a file via the **:p** command, *more* shall attempt to examine the previous file in the argument list, but the final exit status shall be affected. If an error is encountered accessing a file via the **:e** command, *more* shall remain in the current file and the final exit status shall not be affected.

**APPLICATION USAGE**

When the standard output is not a terminal, only the **-s** filter-modification option is effective. This is based on historical practice. For example, a typical implementation of *man* pipes its output through *more -s* to squeeze excess white space for terminal users. When *man* is piped to *lp*, however, it is undesirable for this squeezing to happen.

**EXAMPLES**

The **-p** allows arbitrary commands to be executed at the start of each file. Examples are:

*more -p G file1 file2*  
Examine each file starting with its last screenful.

*more -p 100 file1 file2*  
Examine each file starting with line 100 in the current position (usually the third line, so line 98 would be the first line written).

*more -p /100 file1 file2*  
Examine each file starting with the first line containing the string "100" in the current position



## RATIONALE

The *more* utility, available in BSD and BSD-derived systems, was chosen as the prototype for the POSIX file display program since it is more widely available than either the public-domain program *less* or than *pg*, a pager provided in System V. The 4.4 BSD *more* is the model for the features selected; it is almost fully upwards-compatible from the 4.3 BSD version in wide use and has become more amenable for *vi* users. Several features originally derived from various file editors, found in both *less* and *pg*, have been added to this volume of POSIX.1-2008 as they have proved extremely popular with users.

There are inconsistencies between *more* and *vi* that result from historical practice. For example, the single-character commands **h**, **f**, **b**, and **<space>** are screen movers in *more*, but cursor movers in *vi*. These inconsistencies were maintained because the cursor movements are not applicable to *more* and the powerful functionality achieved without the use of the control key justifies the differences.

The tags interface has been included in a program that is not a text editor because it promotes another degree of consistent operation with *vi*. It is conceivable that the paging environment of *more* would be superior for browsing source code files in some circumstances.

The operating mode referred to for block-mode terminals effectively adds a **<newline>** to each Synopsis line that currently has none. So, for example, **d<newline>** would page one screenful. The mode could be triggered by a command line option, environment variable, or some other method. The details are not imposed by this volume of POSIX.1-2008 because there are so few systems known to support such terminals. Nevertheless, it was considered that all systems should be able to support *more* given the exception cited for this small community of terminals because, in comparison to *vi*, the cursor movements are few and the command set relatively amenable to the optional **<newline>** characters.

Some versions of *more* provide a shell escaping mechanism similar to the *ex* **!** command. The standard developers did not consider that this was necessary in a paginator, particularly given the wide acceptance of multiple window terminals and job control features. (They chose to retain such features in the editors and *mailx* because the shell interaction also gives an opportunity to modify the editing buffer, which is not applicable to *more*.)

The **-p** (position) option replaces the **+** command because of the Utility Syntax Guidelines. The **+command** option is no longer specified by POSIX.1-2008 but may be present in some implementations. In early proposals, it took a *pattern* argument, but historical *less* provided the *more* general facility of a command. It would have been desirable to use the same **-c** as *ex* and *vi*, but the letter was already in use.

The text stating “from a non-rewindable stream ... implementations may limit the amount of backwards motion supported” would allow an implementation that permitted no backwards motion beyond text already on the screen. It was not possible to require a minimum amount of backwards motion that would be effective for all conceivable device types. The implementation should allow the user to back up as far as possible, within device and reasonable memory allocation constraints.

Historically, non-printable characters were displayed using the ARPA standard mappings, which are as follows:

1. Printable characters are left alone.
2. Control characters less than **\177** are represented as followed by the character offset from the **'@'** character in the ASCII map; for example, **\007** is represented as **'G'**.

98732 3. \177 is represented as followed by ' ? '.

98733 The display of characters having their eighth bit set was less standard. Existing implementations  
 98734 use hex (0x00), octal (\000), and a meta-bit display. (The latter displayed characters with their  
 98735 eighth bit set as the two characters "M-", followed by the seven-bit display as described  
 98736 previously.) The latter probably has the best claim to historical practice because it was used with  
 98737 the -v option of 4 BSD and 4 BSD-derived versions of the *cat* utility since 1980.

98738 No specific display format is required by POSIX.1-2008. Implementations are encouraged to  
 98739 conform to historic practice in the absence of any strong reason to diverge.

#### 98740 FUTURE DIRECTIONS

98741 None.

#### 98742 SEE ALSO

98743 [Chapter 2](#) (on page 2319), *ctags*, *ed*, *ex*, *vi*

98744 XBD [Chapter 8](#) (on page 173), [Section 9.2](#) (on page 182), [Section 9.3](#) (on page 183), [Section 12.2](#)  
 98745 (on page 215)

#### 98746 CHANGE HISTORY

98747 First released in Issue 4.

#### 98748 Issue 5

98749 The FUTURE DIRECTIONS section is added.

#### 98750 Issue 6

98751 This utility is marked as part of the User Portability Utilities option.

98752 The obsolescent SYNOPSIS is removed.

98753 The utility has been extensively reworked for alignment with the IEEE P1003.2b draft standard:

- 98754 • Changes have been made as a result of IEEE PASC Interpretations 1003.2 #37 and #109.
- 98755 • The *more* utility should be able to handle underlined and emboldened displays of
- 98756 characters that are wider than a single column position.

#### 98757 Issue 7

98758 Austin Group Interpretation 1003.1-2001 #027 is applied, clarifying that ' +' may be recognized  
 98759 as an option delimiter in the OPTIONS section.

98760 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

98761 POSIX.1-2008, Technical Corrigendum 1, XCU/TC1-2008/0123 [265] is applied.

+

**NAME**

mv — move files

**SYNOPSIS**

mv [-if] *source\_file target\_file*

mv [-if] *source\_file... target\_dir*

**DESCRIPTION**

In the first synopsis form, the *mv* utility shall move the file named by the *source\_file* operand to the destination specified by the *target\_file*. This first synopsis form is assumed when the final operand does not name an existing directory and is not a symbolic link referring to an existing directory. In this case, if *source\_file* names a non-directory file and *target\_file* ends with a trailing <slash> character, *mv* shall treat this as an error and no *source\_file* operands will be processed. +

In the second synopsis form, *mv* shall move each file named by a *source\_file* operand to a destination file in the existing directory named by the *target\_dir* operand, or referenced if *target\_dir* is a symbolic link referring to an existing directory. The destination path for each *source\_file* shall be the concatenation of the target directory, a single <slash> character if the target did not end in a <slash>, and the last pathname component of the *source\_file*. This second form is assumed when the final operand names an existing directory.

If any operand specifies an existing file of a type not specified by the System Interfaces volume of POSIX.1-2008, the behavior is implementation-defined.

For each *source\_file* the following steps shall be taken:

1. If the destination path exists, the *-f* option is not specified, and either of the following conditions is true:

- a. The permissions of the destination path do not permit writing and the standard input is a terminal.
- b. The *-i* option is specified.

the *mv* utility shall write a prompt to standard error and read a line from standard input. If the response is not affirmative, *mv* shall do nothing more with the current *source\_file* and go on to any remaining *source\_files*.

2. If the *source\_file* operand and destination path name the same existing file, then the destination path shall not be removed, and one of the following shall occur:

- a. No change is made to *source\_file*, no error occurs, and no diagnostic is issued.
- b. No change is made to *source\_file*, a diagnostic is issued to standard error identifying the two names, and the exit status is affected.
- c. If the *source\_file* operand and destination path name distinct directory entries, then the *source\_file* operand is removed, no error occurs, and no diagnostic is issued.

The *mv* utility shall do nothing more with the current *source\_file*, and go on to any remaining *source\_files*.

3. The *mv* utility shall perform actions equivalent to the *rename()* function defined in the System Interfaces volume of POSIX.1-2008, called with the following arguments:

- a. The *source\_file* operand is used as the *old* argument.
- b. The destination path is used as the *new* argument.

If this succeeds, *mv* shall do nothing more with the current *source\_file* and go on to any remaining *source\_files*. If this fails for any reasons other than those described for the *errno*

[EXDEV] in the System Interfaces volume of POSIX.1-2008, *mv* shall write a diagnostic message to standard error, do nothing more with the current *source\_file*, and go on to any remaining *source\_files*.

4. If the destination path exists, and it is a file of type directory and *source\_file* is not a file of type directory, or it is a file not of type directory and *source\_file* is a file of type directory, *mv* shall write a diagnostic message to standard error, do nothing more with the current *source\_file*, and go on to any remaining *source\_files*. If the destination path exists and was created by a previous step, it is unspecified whether this will be treated as an error or the destination path will be overwritten.
5. If the destination path exists, *mv* shall attempt to remove it. If this fails for any reason, *mv* shall write a diagnostic message to standard error, do nothing more with the current *source\_file*, and go on to any remaining *source\_files*.
6. The file hierarchy rooted in *source\_file* shall be duplicated as a file hierarchy rooted in the destination path. If *source\_file* or any of the files below it in the hierarchy are symbolic links, the links themselves shall be duplicated, including their contents, rather than any files to which they refer. The following characteristics of each file in the file hierarchy shall be duplicated:
  - The time of last data modification and time of last access
  - The user ID and group ID
  - The file mode

If the user ID, group ID, or file mode of a regular file cannot be duplicated, the file mode bits `S_ISUID` and `S_ISGID` shall not be duplicated.

When files are duplicated to another file system, the implementation may require that the process invoking *mv* has read access to each file being duplicated.

If files being duplicated to another file system have hard links to other files, it is unspecified whether the files copied to the new file system have the hard links preserved or separate copies are created for the linked files.

If the duplication of the file hierarchy fails for any reason, *mv* shall write a diagnostic message to standard error, do nothing more with the current *source\_file*, and go on to any remaining *source\_files*.

If the duplication of the file characteristics fails for any reason, *mv* shall write a diagnostic message to standard error, but this failure shall not cause *mv* to modify its exit status.
7. The file hierarchy rooted in *source\_file* shall be removed. If this fails for any reason, *mv* shall write a diagnostic message to the standard error, do nothing more with the current *source\_file*, and go on to any remaining *source\_files*.

## OPTIONS

The *mv* utility shall conform to XBD [Section 12.2](#) (on page 215).

The following options shall be supported:

- f** Do not prompt for confirmation if the destination path exists. Any previous occurrence of the **-i** option is ignored.
- i** Prompt for confirmation if the destination path exists. Any previous occurrence of the **-f** option is ignored.

Specifying more than one of the **-f** or **-i** options shall not be considered an error. The last option

98848 specified shall determine the behavior of *mv*.

#### 98849 OPERANDS

98850 The following operands shall be supported:

98851 *source\_file* A pathname of a file or directory to be moved.

98852 *target\_file* A new pathname for the file or directory being moved.

98853 *target\_dir* A pathname of an existing directory into which to move the input files.

#### 98854 STDIN

98855 The standard input shall be used to read an input line in response to each prompt specified in  
98856 the STDERR section. Otherwise, the standard input shall not be used.

#### 98857 INPUT FILES

98858 The input files specified by each *source\_file* operand can be of any file type.

#### 98859 ENVIRONMENT VARIABLES

98860 The following environment variables shall affect the execution of *mv*:

98861 *LANG* Provide a default value for the internationalization variables that are unset or null.  
98862 (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
98863 variables used to determine the values of locale categories.)

98864 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
98865 internationalization variables.

98866 *LC\_COLLATE*

98867 Determine the locale for the behavior of ranges, equivalence classes, and multi-  
98868 character collating elements used in the extended regular expression defined for  
98869 the **yesexpr** locale keyword in the *LC\_MESSAGES* category.

98870 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
98871 characters (for example, single-byte as opposed to multi-byte characters in  
98872 arguments and input files), the behavior of character classes used in the extended  
98873 regular expression defined for the **yesexpr** locale keyword in the *LC\_MESSAGES*  
98874 category.

98875 *LC\_MESSAGES*

98876 Determine the locale used to process affirmative responses, and the locale used to  
98877 affect the format and contents of diagnostic messages and prompts written to  
98878 standard error.

98879 XSI *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

#### 98880 ASYNCHRONOUS EVENTS

98881 Default.

#### 98882 STDOUT

98883 Not used.

#### 98884 STDERR

98885 Prompts shall be written to the standard error under the conditions specified in the  
98886 DESCRIPTION section. The prompts shall contain the destination pathname, but their format is  
98887 otherwise unspecified. Otherwise, the standard error shall be used only for diagnostic  
98888 messages.

**OUTPUT FILES**

The output files may be of any file type.

**EXTENDED DESCRIPTION**

None.

**EXIT STATUS**

The following exit values shall be returned:

0 All input files were moved successfully.

>0 An error occurred.

**CONSEQUENCES OF ERRORS**

If the copying or removal of *source\_file* is prematurely terminated by a signal or error, *mv* may leave a partial copy of *source\_file* at the source or destination. The *mv* utility shall not modify both *source\_file* and the destination path simultaneously; termination at any point shall leave either *source\_file* or the destination path complete.

**APPLICATION USAGE**

Some implementations mark for update the last file status change timestamp of renamed files and some do not. Applications which make use of the last file status change timestamp may behave differently with respect to renamed files unless they are designed to allow for either behavior.

The specification ensures that *mv a a* will not alter the contents of file **a**, and allows the implementation to issue an error that a file cannot be moved onto itself. Likewise, when **a** and **b** are hard links to the same file, *mv a b* will not alter **b**, but if a diagnostic is not issued, then it is unspecified whether **a** is left untouched (as it would be by the *rename()* function) or unlinked (reducing the link count of **b**).

**EXAMPLES**

If the current directory contains only files **a** (of any type defined by the System Interfaces volume of POSIX.1-2008), **b** (also of any type), and a directory **c**:

```
mv a b c
```

```
mv c d
```

results with the original files **a** and **b** residing in the directory **d** in the current directory.

**RATIONALE**

Early proposals diverged from the SVID and BSD historical practice in that they required that when the destination path exists, the *-f* option is not specified, and input is not a terminal, *mv* fails. This was done for compatibility with *cp*. The current text returns to historical practice. It should be noted that this is consistent with the *rename()* function defined in the System Interfaces volume of POSIX.1-2008, which does not require write permission on the target.

For absolute clarity, paragraph (1), describing the behavior of *mv* when prompting for confirmation, should be interpreted in the following manner:

```
if (exists AND (NOT f_option) AND
 ((not_writable AND input_is_terminal) OR i_option))
```

The *-i* option exists on BSD systems, giving applications and users a way to avoid accidentally unlinking files when moving others. When the standard input is not a terminal, the 4.3 BSD *mv* deletes all existing destination paths without prompting, even when *-i* is specified; this is inconsistent with the behavior of the 4.3 BSD *cp* utility, which always generates an error when the file is unwritable and the standard input is not a terminal. The standard developers decided that use of *-i* is a request for interaction, so when the destination path exists, the utility takes

instructions from whatever responds to standard input.

The `rename()` function is able to move directories within the same file system. Some historical versions of `mv` have been able to move directories, but not to a different file system. The standard developers considered that this was an annoying inconsistency, so this volume of POSIX.1-2008 requires directories to be able to be moved even across file systems. There is no `-R` option to confirm that moving a directory is actually intended, since such an option was not required for moving directories in historical practice. Requiring the application to specify it sometimes, depending on the destination, seemed just as inconsistent. The semantics of the `rename()` function were preserved as much as possible. For example, `mv` is not permitted to “rename” files to or from directories, even though they might be empty and removable.

Historic implementations of `mv` did not exit with a non-zero exit status if they were unable to duplicate any file characteristics when moving a file across file systems, nor did they write a diagnostic message for the user. The former behavior has been preserved to prevent scripts from breaking; a diagnostic message is now required, however, so that users are alerted that the file characteristics have changed.

The exact format of the interactive prompts is unspecified. Only the general nature of the contents of prompts are specified because implementations may desire more descriptive prompts than those used on historical implementations. Therefore, an application not using the `-f` option or using the `-i` option relies on the system to provide the most suitable dialog directly with the user, based on the behavior specified.

When `mv` is dealing with a single file system and `source_file` is a symbolic link, the link itself is moved as a consequence of the dependence on the `rename()` functionality, per the DESCRIPTION. Across file systems, this has to be made explicit.

## FUTURE DIRECTIONS

None.

## SEE ALSO

[\*cp\*](#), [\*ln\*](#)

XBD [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 215)

XSH [`rename\(\)`](#)

## CHANGE HISTORY

First released in Issue 2.

### Issue 6

The `mv` utility is changed to describe processing of symbolic links as specified in the IEEE P1003.2b draft standard.

The APPLICATION USAGE section is added.

### Issue 7

Austin Group Interpretation 1003.1-2001 #016 is applied.

Austin Group Interpretation 1003.1-2001 #126 is applied, changing the description of the `LC_MESSAGES` environment variable.

Austin Group Interpretations 1003.1-2001 #164, #168, and #169 are applied.

SD5-XCU-ERN-13 is applied, making an editorial correction to the SYNOPSIS.

SD5-XCU-ERN-51 is applied to the DESCRIPTION, defining the behavior for when files are being duplicated to another file system while having hard links.

|       |                                                                           |                                                                                       |
|-------|---------------------------------------------------------------------------|---------------------------------------------------------------------------------------|
| 98977 | Changes are made related to support for finegrained timestamps.           |                                                                                       |
| 98978 | POSIX.1-2008, Technical Corrigendum 1, XCU/TC1-2008/0124 [48] is applied. | +  |



98979 **NAME**98980 `newgrp` — change to a new group98981 **SYNOPSIS**98982 `newgrp [-l] [group]`98983 **DESCRIPTION**

98984 The *newgrp* utility shall create a new shell execution environment with a new real and effective  
 98985 group identification. Of the attributes listed in [Section 2.12](#) (on page 2353), the new shell  
 98986 execution environment shall retain the working directory, file creation mask, and exported  
 98987 variables from the previous environment (that is, open files, traps, unexported variables, alias  
 98988 definitions, shell functions, and *set* options may be lost). All other aspects of the process  
 98989 environment that are preserved by the *exec* family of functions defined in the System Interfaces  
 98990 volume of POSIX.1-2008 shall also be preserved by *newgrp*; whether other aspects are preserved  
 98991 is unspecified.

98992 A failure to assign the new group identifications (for example, for security or password-related  
 98993 reasons) shall not prevent the new shell execution environment from being created.

98994 The *newgrp* utility shall affect the supplemental groups for the process as follows:

- 98995 • On systems where the effective group ID is normally in the supplementary group list (or  
 98996 whenever the old effective group ID actually is in the supplementary group list):
  - 98997 — If the new effective group ID is also in the supplementary group list, *newgrp* shall  
 98998 change the effective group ID.
  - 98999 — If the new effective group ID is not in the supplementary group list, *newgrp* shall add  
 99000 the new effective group ID to the list, if there is room to add it.
- 99001 • On systems where the effective group ID is not normally in the supplementary group list  
 99002 (or whenever the old effective group ID is not in the supplementary group list):
  - 99003 — If the new effective group ID is in the supplementary group list, *newgrp* shall delete  
 99004 it.
  - 99005 — If the old effective group ID is not in the supplementary list, *newgrp* shall add it if  
 99006 there is room.

99007 **Note:** The System Interfaces volume of POSIX.1-2008 does not specify whether the effective group ID  
 99008 of a process is included in its supplementary group list.

99009 With no operands, *newgrp* shall change the effective group back to the groups identified in the  
 99010 user's user entry, and shall set the list of supplementary groups to that set in the user's group  
 99011 database entries.

99012 If the first argument is `'-'`, the results are unspecified.

99013 If a password is required for the specified group, and the user is not listed as a member of that  
 99014 group in the group database, the user shall be prompted to enter the correct password for that  
 99015 group. If the user is listed as a member of that group, no password shall be requested. If no  
 99016 password is required for the specified group, it is implementation-defined whether users not  
 99017 listed as members of that group can change to that group. Whether or not a password is  
 99018 required, implementation-defined system accounting or security mechanisms may impose  
 99019 additional authorization restrictions that may cause *newgrp* to write a diagnostic message and  
 99020 suppress the changing of the group identification.

**OPTIONS**

The *newgrp* utility shall conform to XBD [Section 12.2](#) (on page 215), except for the unspecified usage of *'-'*.

The following option shall be supported:

**-l** (The letter ell.) Change the environment to what would be expected if the user actually logged in again.

**OPERANDS**

The following operand shall be supported:

*group* A group name from the group database or a non-negative numeric group ID. Specifies the group ID to which the real and effective group IDs shall be set. If *group* is a non-negative numeric string and exists in the group database as a group name (see *getgrnam()*), the numeric group ID associated with that group name shall be used as the group ID.

**STDIN**

Not used.

**INPUT FILES**

The file */dev/tty* shall be used to read a single line of text for password checking, when one is required.

**ENVIRONMENT VARIABLES**

The following environment variables shall affect the execution of *newgrp*:

**LANG** Provide a default value for the internationalization variables that are unset or null. (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization variables used to determine the values of locale categories.)

**LC\_ALL** If set to a non-empty string value, override the values of all the other internationalization variables.

**LC\_CTYPE** Determine the locale for the interpretation of sequences of bytes of text data as characters (for example, single-byte as opposed to multi-byte characters in arguments).

**LC\_MESSAGES**

Determine the locale that should be used to affect the format and contents of diagnostic messages written to standard error.

**NSLSPATH** Determine the location of message catalogs for the processing of **LC\_MESSAGES**.

**ASYNCHRONOUS EVENTS**

Default.

**STDOUT**

Not used.

**STDERR**

The standard error shall be used for diagnostic messages and a prompt string for a password, if one is required. Diagnostic messages may be written in cases where the exit status is not available. See the EXIT STATUS section.

**99061 OUTPUT FILES**

99062 None.

**99063 EXTENDED DESCRIPTION**

99064 None.

**99065 EXIT STATUS**

99066 If *newgrp* succeeds in creating a new shell execution environment, whether or not the group  
99067 identification was changed successfully, the exit status shall be the exit status of the shell.  
99068 Otherwise, the following exit value shall be returned:

99069 >0 An error occurred.

**99070 CONSEQUENCES OF ERRORS**

99071 The invoking shell may terminate.

**99072 APPLICATION USAGE**

99073 There is no convenient way to enter a password into the group database. Use of group  
99074 passwords is not encouraged, because by their very nature they encourage poor security  
99075 practices. Group passwords may disappear in the future.

99076 A common implementation of *newgrp* is that the current shell uses *exec* to overlay itself with  
99077 *newgrp*, which in turn overlays itself with a new shell after changing group. On some  
99078 implementations, however, this may not occur and *newgrp* may be invoked as a subprocess.

99079 The *newgrp* command is intended only for use from an interactive terminal. It does not offer a  
99080 useful interface for the support of applications.

99081 The exit status of *newgrp* is generally inapplicable. If *newgrp* is used in a script, in most cases it  
99082 successfully invokes a new shell and the rest of the original shell script is bypassed when the  
99083 new shell exits. Used interactively, *newgrp* displays diagnostic messages to indicate problems.  
99084 But usage such as:

99085 `newgrp foo`  
99086 `echo $?`

99087 is not useful because the new shell might not have access to any status *newgrp* may have  
99088 generated (and most historical systems do not provide this status). A zero status echoed here  
99089 does not necessarily indicate that the user has changed to the new group successfully. Following  
99090 *newgrp* with the *id* command provides a portable means of determining whether the group  
99091 change was successful or not.

**99092 EXAMPLES**

99093 None.

**99094 RATIONALE**

99095 Most historical implementations use one of the *exec* functions to implement the behavior of  
99096 *newgrp*. Errors detected before the *exec* leave the environment unchanged, while errors detected  
99097 after the *exec* leave the user in a changed environment. While it would be useful to have *newgrp*  
99098 issue a diagnostic message to tell the user that the environment changed, it would be  
99099 inappropriate to require this change to some historical implementations.

99100 The password mechanism is allowed in the group database, but how this would be  
99101 implemented is not specified.

99102 The *newgrp* utility was retained in this volume of POSIX.1-2008, even given the existence of the  
99103 multiple group permissions feature in the System Interfaces volume of POSIX.1-2008, for several  
99104 reasons. First, in some implementations, the group ownership of a newly created file is  
99105 determined by the group of the directory in which the file is created, as allowed by the System

99106 Interfaces volume of POSIX.1-2008; on other implementations, the group ownership of a newly  
99107 created file is determined by the effective group ID. On implementations of the latter type,  
99108 *newgrp* allows files to be created with a specific group ownership. Finally, many  
99109 implementations use the real group ID in accounting, and on such systems, *newgrp* allows the  
99110 accounting identity of the user to be changed.

#### 99111 FUTURE DIRECTIONS

99112 None.

#### 99113 SEE ALSO

99114 [Chapter 2](#) (on page 2319), *sh*

99115 XBD [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 215)

99116 XSH *exec*, *getgrnam()*

#### 99117 CHANGE HISTORY

99118 First released in Issue 2.

#### 99119 Issue 6

99120 This utility is marked as part of the User Portability Utilities option.

99121 The obsolescent SYNOPSIS is removed.

99122 The text describing supplemental groups is no longer conditional on {NGROUPS\_MAX} being  
99123 greater than 1. This is because {NGROUPS\_MAX} now has a minimum value of 8. This is a FIPS  
99124 requirement.

#### 99125 Issue 7

99126 Austin Group Interpretation 1003.1-2001 #027 is applied, clarifying the behavior if the first  
99127 argument is *'-'*.

99128 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

99129 The *newgrp* utility is moved from the User Portability Utilities option to the Base. User  
99130 Portability Utilities is now an option for interactive utilities.

99131 **NAME**99132        *nice* — invoke a utility with an altered nice value99133 **SYNOPSIS**99134        *nice* [*-n increment*] *utility* [*argument...*]99135 **DESCRIPTION**

99136        The *nice* utility shall invoke a utility, requesting that it be run with a different nice value (see  
 99137        XBD [Section 3.240](#), on page 71). With no options, the executed utility shall be run with a nice  
 99138        value that is some implementation-defined quantity greater than or equal to the nice value of the  
 99139        current process. If the user lacks appropriate privileges to affect the nice value in the requested  
 99140        manner, the *nice* utility shall not affect the nice value; in this case, a warning message may be  
 99141        written to standard error, but this shall not prevent the invocation of *utility* or affect the exit  
 99142        status.

99143 **OPTIONS**99144        The *nice* utility shall conform to XBD [Section 12.2](#) (on page 215).

99145        The following option is supported:

99146        *-n increment* A positive or negative decimal integer which shall have the same effect on the  
 99147        execution of the utility as if the utility had called the *nice()* function with the  
 99148        numeric value of the *increment* option-argument.

99149 **OPERANDS**

99150        The following operands shall be supported:

99151        *utility*        The name of a utility that is to be invoked. If the *utility* operand names any of the  
 99152        special built-in utilities in [Section 2.14](#) (on page 2356), the results are undefined.

99153        *argument*      Any string to be supplied as an argument when invoking the utility named by the  
 99154        *utility* operand.

99155 **STDIN**

99156        Not used.

99157 **INPUT FILES**

99158        None.

99159 **ENVIRONMENT VARIABLES**99160        The following environment variables shall affect the execution of *nice*:

99161        *LANG*        Provide a default value for the internationalization variables that are unset or null.  
 99162        (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 99163        variables used to determine the values of locale categories.)

99164        *LC\_ALL*        If set to a non-empty string value, override the values of all the other  
 99165        internationalization variables.

99166        *LC\_CTYPE*      Determine the locale for the interpretation of sequences of bytes of text data as  
 99167        characters (for example, single-byte as opposed to multi-byte characters in  
 99168        arguments).

99169        *LC\_MESSAGES*

99170        Determine the locale that should be used to affect the format and contents of  
 99171        diagnostic messages written to standard error.

99172        *XSIX*        *NLSPATH*      Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

99173 *PATH* Determine the search path used to locate the utility to be invoked. See XBD  
 99174 Chapter 8 (on page 173).

99175 **ASYNCHRONOUS EVENTS**

99176 Default.

99177 **STDOUT**

99178 Not used.

99179 **STDERR**

99180 The standard error shall be used only for diagnostic messages.

99181 **OUTPUT FILES**

99182 None.

99183 **EXTENDED DESCRIPTION**

99184 None.

99185 **EXIT STATUS**

99186 If *utility* is invoked, the exit status of *nice* shall be the exit status of *utility*; otherwise, the *nice*  
 99187 utility shall exit with one of the following values:

99188 1-125 An error occurred in the *nice* utility.

99189 126 The utility specified by *utility* was found but could not be invoked.

99190 127 The utility specified by *utility* could not be found.

99191 **CONSEQUENCES OF ERRORS**

99192 Default.

99193 **APPLICATION USAGE**

99194 The only guaranteed portable uses of this utility are:

99195 *nice utility*

99196 Run *utility* with the default higher or equal nice value.

99197 *nice -n <positive integer> utility*

99198 Run *utility* with a higher nice value.

99199 On some implementations they have no discernible effect on the invoked utility and on some  
 99200 others they are exactly equivalent.

99201 Historical systems have frequently supported the *<positive integer>* up to 20. Since there is no  
 99202 error penalty associated with guessing a number that is too high, users without access to the  
 99203 system conformance document (to see what limits are actually in place) could use the historical 1  
 99204 to 20 range or attempt to use very large numbers if the job should be truly low priority.

99205 The nice value of a process can be displayed using the command:

99206 `ps -o nice`

99207 The *command*, *env*, *nice*, *nohup*, *time*, and *xargs* utilities have been specified to use exit code 127 if  
 99208 an error occurs so that applications can distinguish “failure to find a utility” from “invoked  
 99209 utility exited with an error indication”. The value 127 was chosen because it is not commonly  
 99210 used for other meanings; most utilities use small values for “normal error conditions” and the  
 99211 values above 128 can be confused with termination due to receipt of a signal. The value 126 was  
 99212 chosen in a similar manner to indicate that the utility could be found, but not invoked. Some  
 99213 scripts produce meaningful error messages differentiating the 126 and 127 cases. The distinction  
 99214 between exit codes 126 and 127 is based on KornShell practice that uses 127 when all attempts to  
 99215 *exec* the utility fail with [ENOENT], and uses 126 when any attempt to *exec* the utility fails for

99216 any other reason.

#### 99217 EXAMPLES

99218 None.

#### 99219 RATIONALE

99220 The 4.3 BSD version of *nice* does not check whether *increment* is a valid decimal integer. The  
99221 command *nice -x utility*, for example, would be treated the same as the command *nice --1*  
99222 *utility*. If the user does not have appropriate privileges, this results in a “permission denied”  
99223 error. This is considered a bug.

99224 When a user without appropriate privileges gives a negative *increment*, System V treats it like  
99225 the command *nice -0 utility*, while 4.3 BSD writes a “permission denied” message and does not  
99226 run the utility. The standard specifies the System V behavior together with an optional BSD-style  
99227 “permission denied” message.

99228 The C shell has a built-in version of *nice* that has a different interface from the one described in  
99229 this volume of POSIX.1-2008.

99230 The term “utility” is used, rather than “command”, to highlight the fact that shell compound  
99231 commands, pipelines, and so on, cannot be used. Special built-ins also cannot be used.  
99232 However, “utility” includes user application programs and shell scripts, not just utilities defined  
99233 in this volume of POSIX.1-2008.

99234 Historical implementations of *nice* provide a nice value range of 40 or 41 discrete steps, with the  
99235 default nice value being the midpoint of that range. By default, they raise the nice value of the  
99236 executed utility by 10.

99237 Some historical documentation states that the *increment* value must be within a fixed range. This  
99238 is misleading; the valid *increment* values on any invocation are determined by the current  
99239 process nice value, which is not always the default.

99240 The definition of nice value is not intended to suggest that all processes in a system have  
99241 priorities that are comparable. Scheduling policy extensions such as the realtime priorities in the  
99242 System Interfaces volume of POSIX.1-2008 make the notion of a single underlying priority for all  
99243 scheduling policies problematic. Some implementations may implement the *nice*-related features  
99244 to affect all processes on the system, others to affect just the general time-sharing activities  
99245 implied by this volume of POSIX.1-2008, and others may have no effect at all. Because of the use  
99246 of “implementation-defined” in *nice* and *renice*, a wide range of implementation strategies are  
99247 possible.

99248 Earlier versions of this standard allowed a *-increment* option. This form is no longer specified by  
99249 POSIX.1-2008 but may be present in some implementations.

#### 99250 FUTURE DIRECTIONS

99251 None.

#### 99252 SEE ALSO

99253 [Chapter 2](#) (on page 2319), *renice*

99254 XBD [Section 3.240](#) (on page 71), [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 215)

99255 XSH *nice()*

#### 99256 CHANGE HISTORY

99257 First released in Issue 4.

99258 **Issue 6**

99259 This utility is marked as part of the User Portability Utilities option.

99260 The obsolescent SYNOPSIS is removed.

99261 IEEE Std 1003.1-2001/Cor 2-2004, item XCU/TC2/D6/18 is applied, deleting a paragraph of  
99262 RATIONALE that referred to text no longer in the standard.

99263 **Issue 7**

99264 Austin Group Interpretation 1003.1-2001 #027 is applied.

99265 SD5-XCU-ERN-32 and SD5-XCU-ERN-33 are applied, updating the DESCRIPTION,  
99266 APPLICATION USAGE, and RATIONALE sections.

99267 The *nice* utility is moved from the User Portability Utilities option to the Base. User Portability  
99268 Utilities is now an option for interactive utilities.



99269 **NAME**

99270 nl — line numbering filter

99271 **SYNOPSIS**

```
99272 xSI nl [-p] [-b type] [-d delim] [-f type] [-h type] [-i incr] [-l num]
99273 [-n format] [-s sep] [-v startnum] [-w width] [file]
```

99274 **DESCRIPTION**

99275 The *nl* utility shall read lines from the named *file* or the standard input if no *file* is named and  
 99276 shall reproduce the lines to standard output. Lines shall be numbered on the left. Additional  
 99277 functionality may be provided in accordance with the command options in effect.

99278 The *nl* utility views the text it reads in terms of logical pages. Line numbering shall be reset at  
 99279 the start of each logical page. A logical page consists of a header, a body, and a footer section.  
 99280 Empty sections are valid. Different line numbering options are independently available for  
 99281 header, body, and footer (for example, no numbering of header and footer lines while  
 99282 numbering blank lines only in the body).

99283 The starts of logical page sections shall be signaled by input lines containing nothing but the  
 99284 following delimiter characters:

| Line     | Start of |
|----------|----------|
| \: \: \: | Header   |
| \: \:    | Body     |
| \:       | Footer   |

99289 Unless otherwise specified, *nl* shall assume the text being read is in a single logical page body.

99290 **OPTIONS**

99291 The *nl* utility shall conform to XBD [Section 12.2](#) (on page 215). Only one file can be named.

99292 The following options shall be supported:

99293 **-b *type*** Specify which logical page body lines shall be numbered. Recognized *types* and  
 99294 their meaning are:

99295 **a** Number all lines.

99296 **t** Number only non-empty lines.

99297 **n** No line numbering.

99298 **pstring** Number only lines that contain the basic regular expression specified in  
 99299 *string*.

99300 The default *type* for logical page body shall be **t** (text lines numbered).

99301 **-d *delim*** Specify the delimiter characters that indicate the start of a logical page section.  
 99302 These can be changed from the default characters "\: " to two user-specified  
 99303 characters. If only one character is entered, the second character shall remain the  
 99304 default character ' : '.

99305 **-f *type*** Specify the same as **b *type*** except for footer. The default for logical page footer shall  
 99306 be **n** (no lines numbered).

99307 **-h *type*** Specify the same as **b *type*** except for header. The default *type* for logical page  
 99308 header shall be **n** (no lines numbered).

|       |                           |                                                                                                                                                                                                                                                                                         |
|-------|---------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 99309 | <b>-i</b> <i>incr</i>     | Specify the increment value used to number logical page lines. The default shall be 1.                                                                                                                                                                                                  |
| 99310 |                           |                                                                                                                                                                                                                                                                                         |
| 99311 | <b>-l</b> <i>num</i>      | Specify the number of blank lines to be considered as one. For example, <b>-l 2</b> results in only the second adjacent blank line being numbered (if the appropriate <b>-h a</b> , <b>-b a</b> , or <b>-f a</b> option is set). The default shall be 1.                                |
| 99312 |                           |                                                                                                                                                                                                                                                                                         |
| 99313 |                           |                                                                                                                                                                                                                                                                                         |
| 99314 | <b>-n</b> <i>format</i>   | Specify the line numbering format. Recognized values are: <b>ln</b> , left justified, leading zeros suppressed; <b>rn</b> , right justified, leading zeros suppressed; <b>rz</b> , right justified, leading zeros kept. The default <i>format</i> shall be <b>rn</b> (right justified). |
| 99315 |                           |                                                                                                                                                                                                                                                                                         |
| 99316 |                           |                                                                                                                                                                                                                                                                                         |
| 99317 | <b>-p</b>                 | Specify that numbering should not be restarted at logical page delimiters.                                                                                                                                                                                                              |
| 99318 | <b>-s</b> <i>sep</i>      | Specify the characters used in separating the line number and the corresponding text line. The default <i>sep</i> shall be a <tab>.                                                                                                                                                     |
| 99319 |                           |                                                                                                                                                                                                                                                                                         |
| 99320 | <b>-v</b> <i>startnum</i> | Specify the initial value used to number logical page lines. The default shall be 1.                                                                                                                                                                                                    |
| 99321 | <b>-w</b> <i>width</i>    | Specify the number of characters to be used for the line number. The default <i>width</i> shall be 6.                                                                                                                                                                                   |
| 99322 |                           |                                                                                                                                                                                                                                                                                         |

## 99323 OPERANDS

99324 The following operand shall be supported:

|       |             |                                                |
|-------|-------------|------------------------------------------------|
| 99325 | <i>file</i> | A pathname of a text file to be line-numbered. |
|-------|-------------|------------------------------------------------|

## 99326 STDIN

99327 The standard input shall be used if no *file* operand is specified, and shall be used if the *file*  
 99328 operand is '-' and the implementation treats the '-' as meaning standard input. Otherwise,  
 99329 the standard input shall not be used. See the INPUT FILES section.

## 99330 INPUT FILES

99331 The input file shall be a text file.

## 99332 ENVIRONMENT VARIABLES

99333 The following environment variables shall affect the execution of *nl*:

|       |                    |                                                                                                                                                                                                                                                                                                                                                                                              |
|-------|--------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 99334 | <b>LANG</b>        | Provide a default value for the internationalization variables that are unset or null. (See XBD <a href="#">Section 8.2</a> (on page 174) for the precedence of internationalization variables used to determine the values of locale categories.)                                                                                                                                           |
| 99335 |                    |                                                                                                                                                                                                                                                                                                                                                                                              |
| 99336 |                    |                                                                                                                                                                                                                                                                                                                                                                                              |
| 99337 | <b>LC_ALL</b>      | If set to a non-empty string value, override the values of all the other internationalization variables.                                                                                                                                                                                                                                                                                     |
| 99338 |                    |                                                                                                                                                                                                                                                                                                                                                                                              |
| 99339 | <b>LC_COLLATE</b>  | Determine the locale for the behavior of ranges, equivalence classes, and multi-character collating elements within regular expressions.                                                                                                                                                                                                                                                     |
| 99340 |                    |                                                                                                                                                                                                                                                                                                                                                                                              |
| 99341 |                    |                                                                                                                                                                                                                                                                                                                                                                                              |
| 99342 | <b>LC_CTYPE</b>    | Determine the locale for the interpretation of sequences of bytes of text data as characters (for example, single-byte as opposed to multi-byte characters in arguments and input files), the behavior of character classes within regular expressions, and for deciding which characters are in character class <b>graph</b> (for the <b>-b t</b> , <b>-f t</b> , and <b>-h t</b> options). |
| 99343 |                    |                                                                                                                                                                                                                                                                                                                                                                                              |
| 99344 |                    |                                                                                                                                                                                                                                                                                                                                                                                              |
| 99345 |                    |                                                                                                                                                                                                                                                                                                                                                                                              |
| 99346 |                    |                                                                                                                                                                                                                                                                                                                                                                                              |
| 99347 | <b>LC_MESSAGES</b> | Determine the locale that should be used to affect the format and contents of diagnostic messages written to standard error.                                                                                                                                                                                                                                                                 |
| 99348 |                    |                                                                                                                                                                                                                                                                                                                                                                                              |
| 99349 |                    |                                                                                                                                                                                                                                                                                                                                                                                              |

99350 *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

99351 **ASYNCHRONOUS EVENTS**

99352 Default.

99353 **STDOUT**

99354 The standard output shall be a text file in the following format:

99355 "%s%s%s", *<line number>*, *<separator>*, *<input line>*

99356 where *<line number>* is one of the following numeric formats:

99357 %6d When the **rn** format is used (the default; see **-n**).

99358 %06d When the **rz** format is used.

99359 %-6d When the **ln** format is used.

99360 <empty> When line numbers are suppressed for a portion of the page; the *<separator>* is also suppressed.

99361

99362 In the preceding list, the number 6 is the default width; the **-w** option can change this value.

99363 **STDERR**

99364 The standard error shall be used only for diagnostic messages.

99365 **OUTPUT FILES**

99366 None.

99367 **EXTENDED DESCRIPTION**

99368 None.

99369 **EXIT STATUS**

99370 The following exit values shall be returned:

99371 0 Successful completion.

99372 >0 An error occurred.

99373 **CONSEQUENCES OF ERRORS**

99374 Default.

99375 **APPLICATION USAGE**

99376 In using the **-d delim** option, care should be taken to escape characters that have special meaning to the command interpreter.

99377

99378 **EXAMPLES**

99379 The command:

99380 `nl -v 10 -i 10 -d \!+ file1`

99381 numbers *file1* starting at line number 10 with an increment of 10. The logical page delimiter is

99382 "!+". Note that the '!' has to be escaped when using *cs*h as a command interpreter because of

99383 its history substitution syntax. For *ks*h and *sh* the escape is not necessary, but does not do any

99384 harm.

99385 **RATIONALE**

99386 None.

99387 **FUTURE DIRECTIONS**

99388 None.

99389 **SEE ALSO**99390 *pr*99391 XBD [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 215)99392 **CHANGE HISTORY**

99393 First released in Issue 2.

99394 **Issue 5**99395 The option `[-f type]` is added to the SYNOPSIS. The option descriptions are presented in  
99396 alphabetic order. The description of `-bt` is changed to “Number only non-empty lines”.99397 **Issue 6**99398 The obsolescent behavior allowing the options to be intermingled with the optional *file* operand  
99399 is removed.99400 **Issue 7**

99401 Austin Group Interpretation 1003.1-2001 #092 is applied.

99402 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

99403 **NAME**

99404 nm — write the name list of an object file (DEVELOPMENT)

99405 **SYNOPSIS**

99406 SD nm [-APv] [-g|-u] [-t *format*] *file*...

99407 XSI nm [-APv] [-efox] [-g|-u] [-t *format*] *file*...

99408 **DESCRIPTION**

99409 The *nm* utility shall display symbolic information appearing in the object file, executable file, or

99410 object-file library named by *file*. If no symbolic information is available for a valid input file, the

99411 *nm* utility shall report that fact, but not consider it an error condition.

99412 XSI The default base used when numeric values are written is unspecified. On XSI-conformant

99413 systems, it shall be decimal.

99414 **OPTIONS**99415 The *nm* utility shall conform to XBD [Section 12.2](#) (on page 215).

99416 The following options shall be supported:

99417 **-A** Write the full pathname or library name of an object on each line.99418 XSI **-e** Write only external (global) and static symbol information.

99419 XSI **-f** Produce full output. Write redundant symbols (**.text**, **.data**, and **.bss**), normally

99420 suppressed.

99421 **-g** Write only external (global) symbol information.99422 XSI **-o** Write numeric values in octal (equivalent to **-t o**).99423 **-P** Write information in a portable output format, as specified in the STDOUT section.

99424 **-t *format*** Write each numeric value in the specified format. The format shall be dependent

99425 on the single character used as the *format* option-argument:

99426 XSI d The offset is written in decimal (default).

99427 o The offset is written in octal.

99428 x The offset is written in hexadecimal.

99429 **-u** Write only undefined symbols.99430 **-v** Sort output by value instead of by symbol name.99431 XSI **-x** Write numeric values in hexadecimal (equivalent to **-t x**).99432 **OPERANDS**

99433 The following operand shall be supported:

99434 *file* A pathname of an object file, executable file, or object-file library.99435 **STDIN**

99436 See the INPUT FILES section.

99437 **INPUT FILES**

99438 The input file shall be an object file, an object-file library whose format is the same as those

99439 produced by the *ar* utility for link editing, or an executable file. The *nm* utility may accept

99440 additional implementation-defined object library formats for the input file.

**ENVIRONMENT VARIABLES**

The following environment variables shall affect the execution of *nm*:

**LANG** Provide a default value for the internationalization variables that are unset or null. (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization variables used to determine the values of locale categories.)

**LC\_ALL** If set to a non-empty string value, override the values of all the other internationalization variables.

**LC\_COLLATE**

Determine the locale for character collation information for the symbol-name and symbol-value collation sequences.

**LC\_CTYPE** Determine the locale for the interpretation of sequences of bytes of text data as characters (for example, single-byte as opposed to multi-byte characters in arguments).

**LC\_MESSAGES**

Determine the locale that should be used to affect the format and contents of diagnostic messages written to standard error.

**NSI** **NLSPATH** Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

**ASYNCHRONOUS EVENTS**

Default.

**STDOUT**

If symbolic information is present in the input files, then for each file or for each member of an archive, the *nm* utility shall write the following information to standard output. By default, the format is unspecified, but the output shall be sorted by symbol name according to the collation sequence in the current locale.

- Library or object name, if **-A** is specified
- Symbol name
- Symbol type, which shall either be one of the following single characters or an implementation-defined type represented by a single character:
  - A** Global absolute symbol.
  - a** Local absolute symbol.
  - B** Global “bss” (that is, uninitialized data space) symbol.
  - b** Local bss symbol.
  - D** Global data symbol.
  - d** Local data symbol.
  - T** Global text symbol.
  - t** Local text symbol.
  - U** Undefined symbol.
- Value of the symbol
- The size associated with the symbol, if applicable

This information may be supplemented by additional information specific to the

implementation.

If the **-P** option is specified, the previous information shall be displayed using the following portable format. The three versions differ depending on whether **-t d**, **-t o**, or **-t x** was specified, respectively:

```
"%s%s %s %d %d\n", <library/object name>, <name>, <type>,
 <value>, <size>
```

```
"%s%s %s %o %o\n", <library/object name>, <name>, <type>,
 <value>, <size>
```

```
"%s%s %s %x %x\n", <library/object name>, <name>, <type>,
 <value>, <size>
```

where *<library/object name>* shall be formatted as follows:

- If **-A** is not specified, *<library/object name>* shall be an empty string.
- If **-A** is specified and the corresponding *file* operand does not name a library:

```
"%s: ", <file>
```

- If **-A** is specified and the corresponding *file* operand names a library. In this case, *<object file>* shall name the object file in the library containing the symbol being described:

```
"%s[%s]: ", <file>, <object file>
```

If **-A** is not specified, then if more than one *file* operand is specified or if only one *file* operand is specified and it names a library, *nm* shall write a line identifying the object containing the following symbols before the lines containing those symbols, in the form:

- If the corresponding *file* operand does not name a library:

```
"%s:\n", <file>
```

- If the corresponding *file* operand names a library; in this case, *<object file>* shall be the name of the file in the library containing the following symbols:

```
"%s[%s]:\n", <file>, <object file>
```

If **-P** is specified, but **-t** is not, the format shall be as if **-t x** had been specified.

## STDERR

The standard error shall be used only for diagnostic messages.

## OUTPUT FILES

None.

## EXTENDED DESCRIPTION

None.

## EXIT STATUS

The following exit values shall be returned:

0 Successful completion.

>0 An error occurred.

## CONSEQUENCES OF ERRORS

Default.

**APPLICATION USAGE**

Mechanisms for dynamic linking make this utility less meaningful when applied to an executable file because a dynamically linked executable may omit numerous library routines that would be found in a statically linked executable.

**EXAMPLES**

None.

**RATIONALE**

Historical implementations of *nm* have used different bases for numeric output and supplied different default types of symbols that were reported. The *-t format* option, similar to that used in *od* and *strings*, can be used to specify the numeric base; *-g* and *-u* can be used to restrict the amount of output or the types of symbols included in the output.

The compromise of using *-t format* versus using *-d*, *-o*, and other similar options was necessary because of differences in the meaning of *-o* between implementations. The *-o* option from BSD has been provided here as *-A* to avoid confusion with the *-o* from System V (which has been provided here as *-t* and as *-o* on XSI-conformant systems).

The option list was significantly reduced from that provided by historical implementations.

The *nm* description is a subset of both the System V and BSD *nm* utilities with no specified default output.

It was recognized that mechanisms for dynamic linking make this utility less meaningful when applied to an executable file (because a dynamically linked executable file may omit numerous library routines that would be found in a statically linked executable file), but the value of *nm* during software development was judged to outweigh other limitations.

The default output format of *nm* is not specified because of differences in historical implementations. The *-P* option was added to allow some type of portable output format. After a comparison of the different formats used in SunOS, BSD, SVR3, and SVR4, it was decided to create one that did not match the current format of any of these four systems. The format devised is easy to parse by humans, easy to parse in shell scripts, and does not need to vary depending on locale (because no English descriptions are included). All of the systems currently have the information available to use this format.

The format given in *nm* STDOUT uses <space> characters between the fields, which may be any number of <blank> characters required to align the columns. The single-character types were selected to match historical practice, and the requirement that implementation additions also be single characters made parsing the information easier for shell scripts.

**FUTURE DIRECTIONS**

None.

**SEE ALSO**

*ar*, *c99*

XBD [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 215)

**CHANGE HISTORY**

First released in Issue 2.

**Issue 6**

This utility is marked as supported when both the User Portability Utilities option and the Software Development Utilities option are supported.



99562 **Issue 7**

99563 The *nm* utility is removed from the User Portability Utilities option. User Portability Utilities is  
99564 now an option for interactive utilities.

99565 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

99566 POSIX.1-2008, Technical Corrigendum 1, XCU/TC1-2008/0125 [263] and XCU/TC1-2008/0126 +  
99567 [263] are applied.

99568 **NAME**

99569       nohup — invoke a utility immune to hangups

99570 **SYNOPSIS**99571       nohup *utility* [*argument...*]99572 **DESCRIPTION**

99573       The *nohup* utility shall invoke the utility named by the *utility* operand with arguments supplied  
 99574       as the *argument* operands. At the time the named *utility* is invoked, the SIGHUP signal shall be  
 99575       set to be ignored.

99576       If standard input is associated with a terminal, the *nohup* utility may redirect standard input  
 99577       from an unspecified file.

99578       If the standard output is a terminal, all output written by the named *utility* to its standard output  
 99579       shall be appended to the end of the file **nohup.out** in the current directory. If **nohup.out** cannot  
 99580       be created or opened for appending, the output shall be appended to the end of the file  
 99581       **nohup.out** in the directory specified by the *HOME* environment variable. If neither file can be  
 99582       created or opened for appending, *utility* shall not be invoked. If a file is created, the file's  
 99583       permission bits shall be set to S\_IRUSR | S\_IWUSR.

99584       If standard error is a terminal and standard output is open but is not a terminal, all output  
 99585       written by the named utility to its standard error shall be redirected to the same open file  
 99586       description as the standard output. If standard error is a terminal and standard output either is a  
 99587       terminal or is closed, the same output shall instead be appended to the end of the **nohup.out** file  
 99588       as described above.

99589 **OPTIONS**

99590       None.

99591 **OPERANDS**

99592       The following operands shall be supported:

99593       *utility*       The name of a utility that is to be invoked. If the *utility* operand names any of the  
 99594       special built-in utilities in [Section 2.14](#) (on page 2356), the results are undefined.

99595       *argument*    Any string to be supplied as an argument when invoking the utility named by the  
 99596       *utility* operand.

99597 **STDIN**

99598       Not used.

99599 **INPUT FILES**

99600       None.

99601 **ENVIRONMENT VARIABLES**99602       The following environment variables shall affect the execution of *nohup*:

99603       *HOME*       Determine the pathname of the user's home directory: if the output file **nohup.out**  
 99604       cannot be created in the current directory, the *nohup* utility shall use the directory  
 99605       named by *HOME* to create the file.

99606       *LANG*       Provide a default value for the internationalization variables that are unset or null.  
 99607       (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 99608       variables used to determine the values of locale categories.)

99609       *LC\_ALL*      If set to a non-empty string value, override the values of all the other  
 99610       internationalization variables.

99611 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
 99612 characters (for example, single-byte as opposed to multi-byte characters in  
 99613 arguments).

99614 *LC\_MESSAGES*  
 99615 Determine the locale that should be used to affect the format and contents of  
 99616 diagnostic messages written to standard error.

99617 XSI *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.  
 99618 *PATH* Determine the search path that is used to locate the utility to be invoked. See XBD  
 99619 Chapter 8 (on page 173).

## 99620 ASYNCHRONOUS EVENTS

99621 The *nohup* utility shall take the standard action for all signals except that *SIGHUP* shall be  
 99622 ignored.

## 99623 STDOUT

99624 If the standard output is not a terminal, the standard output of *nohup* shall be the standard  
 99625 output generated by the execution of the *utility* specified by the operands. Otherwise, nothing  
 99626 shall be written to the standard output.

## 99627 STDERR

99628 If the standard output is a terminal, a message shall be written to the standard error, indicating  
 99629 the name of the file to which the output is being appended. The name of the file shall be either  
 99630 **nohup.out** or **\$HOME/nohup.out**.

## 99631 OUTPUT FILES

99632 Output written by the named utility is appended to the file **nohup.out** (or **\$HOME/nohup.out**),  
 99633 if the conditions hold as described in the DESCRIPTION.

## 99634 EXTENDED DESCRIPTION

99635 None.

## 99636 EXIT STATUS

99637 The following exit values shall be returned:

99638 126 The utility specified by *utility* was found but could not be invoked.  
 99639 127 An error occurred in the *nohup* utility or the utility specified by *utility* could not be  
 99640 found.

99641 Otherwise, the exit status of *nohup* shall be that of the utility specified by the *utility* operand.

## 99642 CONSEQUENCES OF ERRORS

99643 Default.

## 99644 APPLICATION USAGE

99645 The *command*, *env*, *nice*, *nohup*, *time*, and *xargs* utilities have been specified to use exit code 127 if  
 99646 an error occurs so that applications can distinguish “failure to find a utility” from “invoked  
 99647 utility exited with an error indication”. The value 127 was chosen because it is not commonly  
 99648 used for other meanings; most utilities use small values for “normal error conditions” and the  
 99649 values above 128 can be confused with termination due to receipt of a signal. The value 126 was  
 99650 chosen in a similar manner to indicate that the utility could be found, but not invoked. Some  
 99651 scripts produce meaningful error messages differentiating the 126 and 127 cases. The distinction  
 99652 between exit codes 126 and 127 is based on KornShell practice that uses 127 when all attempts to  
 99653 *exec* the utility fail with [ENOENT], and uses 126 when any attempt to *exec* the utility fails for  
 99654 any other reason.

**EXAMPLES**

It is frequently desirable to apply *nohup* to pipelines or lists of commands. This can be done by placing pipelines and command lists in a single file; this file can then be invoked as a utility, and the *nohup* applies to everything in the file.

Alternatively, the following command can be used to apply *nohup* to a complex command:

```
nohup sh -c 'complex-command-line' </dev/null
```

**RATIONALE**

The 4.3 BSD version ignores SIGTERM and SIGHUP, and if *./nohup.out* cannot be used, it fails instead of trying to use *\$HOME/nohup.out*.

The *cs*h utility has a built-in version of *nohup* that acts differently from the *nohup* defined in this volume of POSIX.1-2008.

The term *utility* is used, rather than *command*, to highlight the fact that shell compound commands, pipelines, special built-ins, and so on, cannot be used directly. However, *utility* includes user application programs and shell scripts, not just the standard utilities.

Historical versions of the *nohup* utility use default file creation semantics. Some more recent versions use the permissions specified here as an added security precaution.

Some historical implementations ignore SIGQUIT in addition to SIGHUP; others ignore SIGTERM. An early proposal allowed, but did not require, SIGQUIT to be ignored. Several reviewers objected that *nohup* should only modify the handling of SIGHUP as required by this volume of POSIX.1-2008.

Historical versions of *nohup* did not affect standard input, but that causes problems in the common scenario where the user logs into a system, types the command:

```
nohup make &
```

at the prompt, and then logs out. If standard input is not affected by *nohup*, the login session may not terminate for quite some time, since standard input remains open until *make* exits. To avoid this problem, POSIX.1-2008 allows implementations to redirect standard input if it is a terminal. Since the behavior is implementation-defined, portable applications that may run into the problem should redirect standard input themselves. For example, instead of:

```
nohup make &
```

an application can invoke:

```
nohup make </dev/null &
```

**FUTURE DIRECTIONS**

None.

**SEE ALSO**

[Chapter 2](#) (on page 2319), *sh*

XBD [Chapter 8](#) (on page 173)

XSH [signal\(\)](#)

**CHANGE HISTORY**

First released in Issue 2.

99694 **Issue 7**  
99695

Austin Group Interpretations 1003.1-2001 #104, #105, and #106 are applied.

## 99696 NAME

99697 od — dump files in various formats

## 99698 SYNOPSIS

99699 od [-v] [-A *address\_base*] [-j *skip*] [-N *count*] [-t *type\_string*]...  
99700 [*file*...]99701 XSI od [-bcdosx] [*file*] [[+] *offset*][.][b]]

## 99702 DESCRIPTION

99703 The *od* utility shall write the contents of its input files to standard output in a user-specified  
99704 format.

## 99705 OPTIONS

99706 The *od* utility shall conform to XBD [Section 12.2](#) (on page 215), except that the order of  
99707 XSI presentation of the **-t** options and the **-bcdosx** options is significant.

99708 The following options shall be supported:

99709 **-A** *address\_base*99710 Specify the input offset base. See the EXTENDED DESCRIPTION section. The  
99711 application shall ensure that the *address\_base* option-argument is a character. The  
99712 characters 'd', 'o', and 'x' specify that the offset base shall be written in  
99713 decimal, octal, or hexadecimal, respectively. The character 'n' specifies that the  
99714 offset shall not be written.99715 XSI **-b** Interpret bytes in octal. This shall be equivalent to **-t o1**.99716 XSI **-c** Interpret bytes as characters specified by the current setting of the *LC\_CTYPE*  
99717 category. Certain non-graphic characters appear as C escapes: "NUL=\0",  
99718 "BS=\b", "FF=\f", "NL=\n", "CR=\r", "HT=\t"; others appear as 3-digit octal  
99719 numbers.99720 XSI **-d** Interpret *words* (two-byte units) in unsigned decimal. This shall be equivalent to  
99721 **-t u2**.99722 **-j** *skip* Jump over *skip* bytes from the beginning of the input. The *od* utility shall read or  
99723 seek past the first *skip* bytes in the concatenated input files. If the combined input is  
99724 not at least *skip* bytes long, the *od* utility shall write a diagnostic message to  
99725 standard error and exit with a non-zero exit status.99726 By default, the *skip* option-argument shall be interpreted as a decimal number.  
99727 With a leading 0x or 0X, the offset shall be interpreted as a hexadecimal number;  
99728 otherwise, with a leading '0', the offset shall be interpreted as an octal number.  
99729 Appending the character 'b', 'k', or 'm' to offset shall cause it to be interpreted  
99730 as a multiple of 512, 1024, or 1048576 bytes, respectively. If the *skip* number is  
99731 hexadecimal, any appended 'b' shall be considered to be the final hexadecimal  
99732 digit.99733 **-N** *count* Format no more than *count* bytes of input. By default, *count* shall be interpreted as  
99734 a decimal number. With a leading 0x or 0X, *count* shall be interpreted as a  
99735 hexadecimal number; otherwise, with a leading '0', it shall be interpreted as an  
99736 octal number. If *count* bytes of input (after successfully skipping, if **-j** *skip* is  
99737 specified) are not available, it shall not be considered an error; the *od* utility shall  
99738 format the input that is available.

|       |     |                             |                                                                                                                             |
|-------|-----|-----------------------------|-----------------------------------------------------------------------------------------------------------------------------|
| 99739 | XSI | <b>-o</b>                   | Interpret <i>words</i> (two-byte units) in octal. This shall be equivalent to <b>-t o2</b> .                                |
| 99740 | XSI | <b>-s</b>                   | Interpret <i>words</i> (two-byte units) in signed decimal. This shall be equivalent to                                      |
| 99741 |     | <b>-t d2</b> .              |                                                                                                                             |
| 99742 |     | <b>-t type_string</b>       |                                                                                                                             |
| 99743 |     |                             | Specify one or more output types. See the EXTENDED DESCRIPTION section. The                                                 |
| 99744 |     |                             | application shall ensure that the <i>type_string</i> option-argument is a string specifying                                 |
| 99745 |     |                             | the types to be used when writing the input data. The string shall consist of the                                           |
| 99746 |     |                             | type specification characters a, c, d, f, o, u, and x, specifying named character,                                          |
| 99747 |     |                             | character, signed decimal, floating point, octal, unsigned decimal, and                                                     |
| 99748 |     |                             | hexadecimal, respectively. The type specification characters d, f, o, u, and x can be                                       |
| 99749 |     |                             | followed by an optional unsigned decimal integer that specifies the number of                                               |
| 99750 |     |                             | bytes to be transformed by each instance of the output type. The type specification                                         |
| 99751 |     |                             | character f can be followed by an optional F, D, or L indicating that the conversion                                        |
| 99752 |     |                             | should be applied to an item of type <b>float</b> , <b>double</b> , or <b>long double</b> , respectively.                   |
| 99753 |     |                             | The type specification characters d, o, u, and x can be followed by an optional C, S,                                       |
| 99754 |     |                             | I, or L indicating that the conversion should be applied to an item of type <b>char</b> ,                                   |
| 99755 |     |                             | <b>short</b> , <b>int</b> , or <b>long</b> , respectively. Multiple types can be concatenated within the                    |
| 99756 |     |                             | same <i>type_string</i> and multiple <b>-t</b> options can be specified. Output lines shall be                              |
| 99757 |     |                             | written for each type specified in the order in which the type specification                                                |
| 99758 |     |                             | characters are specified.                                                                                                   |
| 99759 |     | <b>-v</b>                   | Write all input data. Without the <b>-v</b> option, any number of groups of output lines,                                   |
| 99760 |     |                             | which would be identical to the immediately preceding group of output lines                                                 |
| 99761 |     |                             | (except for the byte offsets), shall be replaced with a line containing only an                                             |
| 99762 |     |                             | <asterisk> (' *').                                                                                                          |
| 99763 | XSI | <b>-x</b>                   | Interpret <i>words</i> (two-byte units) in hexadecimal. This shall be equivalent to <b>-t x2</b> .                          |
| 99764 | XSI |                             | Multiple types can be specified by using multiple <b>-bcdostx</b> options. Output lines are written for                     |
| 99765 |     |                             | each type specified in the order in which the types are specified.                                                          |
| 99766 |     | <b>OPERANDS</b>             |                                                                                                                             |
| 99767 |     |                             | The following operands shall be supported:                                                                                  |
| 99768 |     | <i>file</i>                 | A pathname of a file to be read. If no <i>file</i> operands are specified, the standard input                               |
| 99769 |     |                             | shall be used.                                                                                                              |
| 99770 |     |                             | If there are no more than two operands, none of the <b>-A</b> , <b>-j</b> , <b>-N</b> , <b>-t</b> , or <b>-v</b> options is |
| 99771 |     |                             | specified, and either of the following is true: the first character of the last operand                                     |
| 99772 |     |                             | is a <plus-sign> (' + '), or there are two operands and the first character of the last                                     |
| 99773 | XSI |                             | operand is numeric; the last operand shall be interpreted as an offset operand on                                           |
| 99774 |     |                             | XSI-conformant systems. Under these conditions, the results are unspecified on                                              |
| 99775 |     |                             | systems that are not XSI-conformant systems.                                                                                |
| 99776 | XSI | <b>[+]<i>offset</i>[[b]</b> | The <i>offset</i> operand specifies the offset in the file where dumping is to commence.                                    |
| 99777 |     |                             | This operand is normally interpreted as octal bytes. If ' . ' is appended, the offset                                       |
| 99778 |     |                             | shall be interpreted in decimal. If ' b ' is appended, the offset shall be interpreted                                      |
| 99779 |     |                             | in units of 512 bytes.                                                                                                      |
| 99780 |     | <b>STDIN</b>                |                                                                                                                             |
| 99781 |     |                             | The standard input shall be used if no <i>file</i> operands are specified, and shall be used if a <i>file</i>               |
| 99782 |     |                             | operand is ' - ' and the implementation treats the ' - ' as meaning standard input. Otherwise,                              |
| 99783 |     |                             | the standard input shall not be used. See the INPUT FILES section.                                                          |

99784 **INPUT FILES**

99785 The input files can be any file type.

99786 **ENVIRONMENT VARIABLES**99787 The following environment variables shall affect the execution of *od*:

99788 **LANG** Provide a default value for the internationalization variables that are unset or null.  
 99789 (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 99790 variables used to determine the values of locale categories.)

99791 **LC\_ALL** If set to a non-empty string value, override the values of all the other  
 99792 internationalization variables.

99793 **LC\_CTYPE** Determine the locale for the interpretation of sequences of bytes of text data as  
 99794 characters (for example, single-byte as opposed to multi-byte characters in  
 99795 arguments and input files).

99796 **LC\_MESSAGES**  
 99797 Determine the locale that should be used to affect the format and contents of  
 99798 diagnostic messages written to standard error.

99799 **LC\_NUMERIC**  
 99800 Determine the locale for selecting the radix character used when writing floating-  
 99801 point formatted output.

99802 XSI **NLSPATH** Determine the location of message catalogs for the processing of **LC\_MESSAGES**.

99803 **ASYNCHRONOUS EVENTS**

99804 Default.

99805 **STDOUT**

99806 See the EXTENDED DESCRIPTION section.

99807 **STDERR**

99808 The standard error shall be used only for diagnostic messages.

99809 **OUTPUT FILES**

99810 None.

99811 **EXTENDED DESCRIPTION**

99812 The *od* utility shall copy sequentially each input file to standard output, transforming the input  
 99813 XSI data according to the output types specified by the **-t** option or the **-bcdosx** options. If no  
 99814 output type is specified, the default output shall be as if **-t oS** had been specified.

99815 The number of bytes transformed by the output type specifier *c* may be variable depending on  
 99816 the **LC\_CTYPE** category.

99817 The default number of bytes transformed by output type specifiers *d*, *f*, *o*, *u*, and *x* corresponds  
 99818 to the various C-language types as follows. If the *c99* compiler is present on the system, these  
 99819 specifiers shall correspond to the sizes used by default in that compiler. Otherwise, these sizes  
 99820 may vary among systems that conform to POSIX.1-2008.

- 99821 • For the type specifier characters *d*, *o*, *u*, and *x*, the default number of bytes shall
- 99822 correspond to the size of the underlying implementation's basic integer type. For these
- 99823 specifier characters, the implementation shall support values of the optional number of
- 99824 bytes to be converted corresponding to the number of bytes in the C-language types **char**,
- 99825 **short**, **int**, and **long**. These numbers can also be specified by an application as the
- 99826 characters '*C*', '*S*', '*I*', and '*L*', respectively. The implementation shall also support
- 99827 the values 1, 2, 4, and 8, even if it provides no C-Language types of those sizes. The



implementation shall support the decimal value corresponding to the C-language type **long long**. The byte order used when interpreting numeric values is implementation-defined, but shall correspond to the order in which a constant of the corresponding type is stored in memory on the system.

- For the type specifier character `f`, the default number of bytes shall correspond to the number of bytes in the underlying implementation's basic double precision floating-point data type. The implementation shall support values of the optional number of bytes to be converted corresponding to the number of bytes in the C-language types **float**, **double**, and **long double**. These numbers can also be specified by an application as the characters `'F'`, `'D'`, and `'L'`, respectively.

The type specifier character `a` specifies that bytes shall be interpreted as named characters from the International Reference Version (IRV) of the ISO/IEC 646:1991 standard. Only the least significant seven bits of each byte shall be used for this type specification. Bytes with the values listed in the following table shall be written using the corresponding names for those characters.

**Table 4-13** Named Characters in *od*

| Value | Name | Value | Name | Value | Name      | Value | Name |
|-------|------|-------|------|-------|-----------|-------|------|
| \000  | nul  | \001  | soh  | \002  | stx       | \003  | etx  |
| \004  | eot  | \005  | enq  | \006  | ack       | \007  | bel  |
| \010  | bs   | \011  | ht   | \012  | lf or nl* | \013  | vt   |
| \014  | ff   | \015  | cr   | \016  | so        | \017  | si   |
| \020  | dle  | \021  | dc1  | \022  | dc2       | \023  | dc3  |
| \024  | dc4  | \025  | nak  | \026  | syn       | \027  | etb  |
| \030  | can  | \031  | em   | \032  | sub       | \033  | esc  |
| \034  | fs   | \035  | gs   | \036  | rs        | \037  | us   |
| \040  | sp   | \177  | del  |       |           |       |      |

**Note:** The `"\012"` value may be written either as `lf` or `nl`.

The type specifier character `c` specifies that bytes shall be interpreted as characters specified by the current setting of the `LC_CTYPE` locale category. Characters listed in the table in XBD Chapter 5 (on page 121) (`'\'`, `'\a'`, `'\b'`, `'\f'`, `'\n'`, `'\r'`, `'\t'`, `'\v'`) shall be written as the corresponding escape sequences, except that `<backslash>` shall be written as a single `<backslash>` and a NUL shall be written as `'\0'`. Other non-printable characters shall be written as one three-digit octal number for each byte in the character. Printable multi-byte characters shall be written in the area corresponding to the first byte of the character; the two-character sequence `"**"` shall be written in the area corresponding to each remaining byte in the character, as an indication that the character is continued. When either the `-j skip` or `-N count` option is specified along with the `c` type specifier, and this results in an attempt to start or finish in the middle of a multi-byte character, the result is implementation-defined.

The input data shall be manipulated in blocks, where a block is defined as a multiple of the least common multiple of the number of bytes transformed by the specified output types. If the least common multiple is greater than 16, the results are unspecified. Each input block shall be written as transformed by each output type, one per written line, in the order that the output types were specified. If the input block size is larger than the number of bytes transformed by the output type, the output type shall sequentially transform the parts of the input block, and the output from each of the transformations shall be separated by one or more `<blank>` characters.

If, as a result of the specification of the `-N` option or end-of-file being reached on the last input file, input data only partially satisfies an output type, the input shall be extended sufficiently

99875 with null bytes to write the last byte of the input.

99876 Unless **-A n** is specified, the first output line produced for each input block shall be preceded by  
 99877 the input offset, cumulative across input files, of the next byte to be written. The format of the  
 99878 input offset is unspecified; however, it shall not contain any <blank> characters, shall start at the  
 99879 first character of the output line, and shall be followed by one or more <blank> characters. In  
 99880 addition, the offset of the byte following the last byte written shall be written after all the input  
 99881 data has been processed, but shall not be followed by any <blank> characters.

99882 If no **-A** option is specified, the input offset base is unspecified.

## 99883 EXIT STATUS

99884 The following exit values shall be returned:

99885 0 All input files were processed successfully.

99886 >0 An error occurred.

## 99887 CONSEQUENCES OF ERRORS

99888 Default.

## 99889 APPLICATION USAGE

99890 XSI-conformant applications are warned not to use filenames starting with '+' or a first  
 99891 operand starting with a numeric character so that the old functionality can be maintained by  
 99892 implementations, unless they specify one of the **-A**, **-j**, or **-N** options. To guarantee that one of  
 99893 these filenames is always interpreted as a filename, an application could always specify the  
 99894 address base format with the **-A** option.

## 99895 EXAMPLES

99896 If a file containing 128 bytes with decimal values zero to 127, in increasing order, is supplied as  
 99897 standard input to the command:

99898 `od -A d -t a`

99899 on an implementation using an input block size of 16 bytes, the standard output, independent of  
 99900 the current locale setting, would be similar to:

```

99901 0000000 nul soh stx etx eot enq ack bel bs ht nl vt ff cr so si
99902 0000016 dle dc1 dc2 dc3 dc4 nak syn etb can em sub esc fs gs rs us
99903 0000032 sp ! " # $ % & ' () * + , - . /
99904 0000048 0 1 2 3 4 5 6 7 8 9 : ; < = > ?
99905 0000064 @ A B C D E F G H I J K L M N O
99906 0000080 P Q R S T U V W X Y Z [\] ^ _
99907 0000096 ` a b c d e f g h i j k l m n o
99908 0000112 p q r s t u v w x y z { | } ~ del
99909 0000128
```

99910 Note that this volume of POSIX.1-2008 allows **nl** or **lf** to be used as the name for the  
 99911 ISO/IEC 646:1991 standard IRV character with decimal value 10. The IRV names this character  
 99912 **lf** (line feed), but traditional implementations have referred to this character as newline (**nl**) and  
 99913 the POSIX locale character set symbolic name for the corresponding character is a <newline>.

99914 The command:

99915 `od -A o -t o2x2x -N 18`

99916 on a system with 32-bit words and an implementation using an input block size of 16 bytes  
 99917 could write 18 bytes in approximately the following format:

99918 `0000000 032056 031440 041123 042040 052516 044530 020043 031464`

```

99919 342e 3320 4253 4420 554e 4958 2023 3334
99920 342e3320 42534420 554e4958 20233334
99921 0000020 032472
99922 353a
99923 353a0000
99924 0000022

```

The command:

```

99926 od -A d -t f -t o4 -t x4 -N 24 -j 0x15

```

on a system with 64-bit doubles (for example, IEEE Std 754-1985 double precision floating-point format) would skip 21 bytes of input data and then write 24 bytes in approximately the following format:

```

99930 0000000 1.000000000000000e+00 1.573500000000000e+01
99931 07774000000 00000000000 10013674121 35341217270
99932 3ff00000 00000000 402f3851 eb851eb8
99933 0000016 1.406682300000000e+02
99934 10030312542 04370303230
99935 40619562 23e18698
99936 0000024

```

## RATIONALE

The *od* utility went through several names in early proposals, including *hd*, *xd*, and most recently *hexdump*. There were several objections to all of these based on the following reasons:

- The *hd* and *xd* names conflicted with historical utilities that behaved differently.
- The *hexdump* description was much more complex than needed for a simple dump utility.
- The *od* utility has been available on all historical implementations and there was no need to create a new name for a utility so similar to the historical *od* utility.

The original reasons for not standardizing historical *od* were also fairly widespread. Those reasons are given below along with rationale explaining why the standard developers believe that this version does not suffer from the indicated problem:

- The BSD and System V versions of *od* have diverged, and the intersection of features provided by both does not meet the needs of the user community. In fact, the System V version only provides a mechanism for dumping octal bytes and **shorts**, signed and unsigned decimal **shorts**, hexadecimal **shorts**, and ASCII characters. BSD added the ability to dump **floats**, **doubles**, named ASCII characters, and octal, signed decimal, unsigned decimal, and hexadecimal **longs**. The version presented here provides more normalized forms for dumping bytes, **shorts**, **ints**, and **longs** in octal, signed decimal, unsigned decimal, and hexadecimal; **float**, **double**, and **long double**; and named ASCII as well as current locale characters.
- It would not be possible to come up with a compatible superset of the BSD and System V flags that met the requirements of the standard developers. The historical default *od* output is the specified default output of this utility. None of the option letters chosen for this version of *od* conflict with any of the options to historical versions of *od*.
- On systems with different sizes for **short**, **int**, and **long**, there was no way to ask for dumps of **ints**, even in the BSD version. Because of the way options are named, the name space could not be extended to solve these problems. This is why the **-t** option was added (with type specifiers more closely matched to the *printf()* formats used in the rest of this volume of POSIX.1-2008) and the optional field sizes were added to the **d**, **f**, **o**, **u**, and **x** type

specifiers. It is also one of the reasons why the historical practice was not mandated as a required obsolescent form of *od*. (Although the old versions of *od* are not listed as an obsolescent form, implementations are urged to continue to recognize the older forms for several more years.) The *a*, *c*, *f*, *o*, and *x* types match the meaning of the corresponding format characters in the historical implementations of *od* except for the default sizes of the fields converted. The *d* format is signed in this volume of POSIX.1-2008 to match the *printf()* notation. (Historical versions of *od* used *d* as a synonym for *u* in this version. The System V implementation uses *s* for signed decimal; BSD uses *i* for signed decimal and *s* for null-terminated strings.) Other than *d* and *u*, all of the type specifiers match format characters in the historical BSD version of *od*.

The sizes of the C-language types **char**, **short**, **int**, **long**, **float**, **double**, and **long double** are used even though it is recognized that there may be zero or more than one compiler for the C language on an implementation and that they may use different sizes for some of these types. (For example, one compiler might use 2 bytes **shorts**, 2 bytes **ints**, and 4 bytes **longs**, while another compiler (or an option to the same compiler) uses 2 bytes **shorts**, 4 bytes **ints**, and 4 bytes **longs**.) Nonetheless, there has to be a basic size known by the implementation for these types, corresponding to the values reported by invocations of the *getconf* utility when called with *system\_var* operands {UCHAR\_MAX}, {USHORT\_MAX}, {UINT\_MAX}, and {ULONG\_MAX} for the types **char**, **short**, **int**, and **long**, respectively. There are similar constants required by the ISO C standard, but not required by the System Interfaces volume of POSIX.1-2008 or this volume of POSIX.1-2008. They are {FLT\_MANT\_DIG}, {DBL\_MANT\_DIG}, and {LDBL\_MANT\_DIG} for the types **float**, **double**, and **long double**, respectively. If the optional *c99* utility is provided by the implementation and used as specified by this volume of POSIX.1-2008, these are the sizes that would be provided. If an option is used that specifies different sizes for these types, there is no guarantee that the *od* utility is able to interpret binary data output by such a program correctly.

This volume of POSIX.1-2008 requires that the numeric values of these lengths be recognized by the *od* utility and that symbolic forms also be recognized. Thus, a conforming application can always look at an array of **unsigned long** data elements using *od -t uL*.

- The method of specifying the format for the address field based on specifying a starting offset in a file unnecessarily tied the two together. The **-A** option now specifies the address base and the **-S** option specifies a starting offset.
- It would be difficult to break the dependence on US ASCII to achieve an internationalized utility. It does not seem to be any harder for *od* to dump characters in the current locale than it is for the *ed* or *sed* **l** commands. The **c** type specifier does this without difficulty and is completely compatible with the historical implementations of the **c** format character when the current locale uses a superset of the ISO/IEC 646:1991 standard as a codeset. The **a** type specifier (from the BSD **a** format character) was left as a portable means to dump ASCII (or more correctly ISO/IEC 646:1991 standard (IRV)) so that headers produced by *pax* could be deciphered even on systems that do not use the ISO/IEC 646:1991 standard as a subset of their base codeset.

The use of **"\*\*"** as an indication of continuation of a multi-byte character in **c** specifier output was chosen based on seeing an implementation that uses this method. The continuation bytes have to be marked in a way that is not ambiguous with another single-byte or multi-byte character.

An early proposal used **-S** and **-n**, respectively, for the **-j** and **-N** options eventually selected. These were changed to avoid conflicts with historical implementations.

- 100014 The original standard specified **-t o2** as the default when no output type was given. This was  
100015 changed to **-t oS** (the length of a **short**) to accommodate a supercomputer implementation that  
100016 historically used 64 bits as its default (and that defined shorts as 64 bits). This change should not  
100017 affect conforming applications. The requirement to support lengths of 1, 2, and 4 was added at  
100018 the same time to address an historical implementation that had no two-byte data types in its C  
100019 compiler.
- 100020 The use of a basic integer data type is intended to allow the implementation to choose a word  
100021 size commonly used by applications on that architecture.
- 100022 Earlier versions of this standard allowed for implementations with bytes other than eight bits,  
100023 but this has been modified in this version.
- 100024 **FUTURE DIRECTIONS**
- 100025 All option and operand interfaces marked XSI may be removed in a future version.
- 100026 **SEE ALSO**
- 100027 *c99, sed*
- 100028 XBD [Chapter 5](#) (on page 121), [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 215)
- 100029 **CHANGE HISTORY**
- 100030 First released in Issue 2.
- 100031 **Issue 5**
- 100032 In the description of the **-c** option, the phrase “This is equivalent to **-t c.**” is deleted.
- 100033 The FUTURE DIRECTIONS section is modified.
- 100034 **Issue 6**
- 100035 The *od* utility is changed to remove the assumption that **short** was a two-byte entity, as per the  
100036 revisions in the IEEE P1003.2b draft standard.
- 100037 The normative text is reworded to avoid use of the term “must” for application requirements.
- 100038 IEEE Std 1003.1-2001/Cor 1-2002, item XCU/TC1/D6/33 is applied, correcting the examples  
100039 which used an undefined **-n** option, which should have been **-N**.
- 100040 IEEE Std 1003.1-2001/Cor 2-2004, item XCU/TC2/D6/19 is applied, removing text describing  
100041 behavior on systems with bytes consisting of more than eight bits.
- 100042 **Issue 7**
- 100043 Austin Group Interpretation 1003.1-2001 #092 is applied.
- 100044 SD5-XCU-ERN-37 is applied, updating the OPERANDS section.
- 100045 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

100046 **NAME**100047 `paste` — merge corresponding or subsequent lines of files100048 **SYNOPSIS**100049 `paste [-s] [-d list] file...`100050 **DESCRIPTION**100051 The *paste* utility shall concatenate the corresponding lines of the given input files, and write the  
100052 resulting lines to standard output.100053 The default operation of *paste* shall concatenate the corresponding lines of the input files. The  
100054 <newline> of every line except the line from the last input file shall be replaced with a <tab>.100055 If an end-of-file condition is detected on one or more input files, but not all input files, *paste* shall  
100056 behave as though empty lines were read from the files on which end-of-file was detected, unless  
100057 the **-s** option is specified.100058 **OPTIONS**100059 The *paste* utility shall conform to XBD [Section 12.2](#) (on page 215).

100060 The following options shall be supported:

100061 **-d *list*** Unless a <backslash> character appears in *list*, each character in *list* is an element  
100062 specifying a delimiter character. If a <backslash> character appears in *list*, the  
100063 <backslash> character and one or more characters following it are an element  
100064 specifying a delimiter character as described below. These elements specify one or  
100065 more delimiters to use, instead of the default <tab>, to replace the <newline> of  
100066 the input lines. The elements in *list* shall be used circularly; that is, when the list is  
100067 exhausted the first element from the list is reused. When the **-s** option is specified:

- 100068
- The last <newline> in a file shall not be modified.
  - The delimiter shall be reset to the first element of *list* after each *file* operand is  
100069 processed.
- 100070

100071 When the **-s** option is not specified:

- 100072
- The <newline> characters in the file specified by the last *file* operand shall  
100073 not be modified.
  - The delimiter shall be reset to the first element of *list* each time a line is  
100074 processed from each file.
- 100075

100076 If a <backslash> character appears in *list*, it and the character following it shall be  
100077 used to represent the following delimiter characters:100078 `\n` <newline>.100079 `\t` <tab>.100080 `\\` <backslash> character.100081 `\0` Empty string (not a null character). If '`\0`' is immediately followed by the  
100082 character '`x`', the character '`X`', or any character defined by the *LC\_CTYPE*  
100083 **digit** keyword (see XBD [Chapter 7](#), on page 135), the results are unspecified.

100084 If any other characters follow the &lt;backslash&gt;, the results are unspecified.

100085 **-s** Concatenate all of the lines of each separate input file in command line order. The  
100086 <newline> of every line except the last line in each input file shall be replaced with  
100087 the <tab>, unless otherwise specified by the **-d** option.

**100088 OPERANDS**

100089 The following operand shall be supported:

100090 *file* A pathname of an input file. If '-' is specified for one or more of the *files*, the  
 100091 standard input shall be used; the standard input shall be read one line at a time,  
 100092 circularly, for each instance of '-'. Implementations shall support pasting of at  
 100093 least 12 *file* operands.

**100094 STDIN**

100095 The standard input shall be used only if one or more *file* operands is '-'. See the INPUT FILES  
 100096 section.

**100097 INPUT FILES**

100098 The input files shall be text files, except that line lengths shall be unlimited.

**100099 ENVIRONMENT VARIABLES**

100100 The following environment variables shall affect the execution of *paste*:

100101 *LANG* Provide a default value for the internationalization variables that are unset or null.  
 100102 (See XBD Section 8.2 (on page 174) the precedence of internationalization variables  
 100103 used to determine the values of locale categories.)

100104 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
 100105 internationalization variables.

100106 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
 100107 characters (for example, single-byte as opposed to multi-byte characters in  
 100108 arguments and input files).

100109 *LC\_MESSAGES*

100110 Determine the locale that should be used to affect the format and contents of  
 100111 diagnostic messages written to standard error.

100112 XSI *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

**100113 ASYNCHRONOUS EVENTS**

100114 Default.

**100115 STDOUT**

100116 Concatenated lines of input files shall be separated by the <tab> (or other characters under the  
 100117 control of the -d option) and terminated by a <newline>.

**100118 STDERR**

100119 The standard error shall be used only for diagnostic messages.

**100120 OUTPUT FILES**

100121 None.

**100122 EXTENDED DESCRIPTION**

100123 None.

**100124 EXIT STATUS**

100125 The following exit values shall be returned:

100126 0 Successful completion.

100127 >0 An error occurred.

## 100128 CONSEQUENCES OF ERRORS

100129 If one or more input files cannot be opened when the `-s` option is not specified, a diagnostic  
 100130 message shall be written to standard error, but no output is written to standard output. If the `-s`  
 100131 option is specified, the *paste* utility shall provide the default behavior described in [Section 1.4](#) (on  
 100132 page 2310).

## 100133 APPLICATION USAGE

100134 When the escape sequences of the *list* option-argument are used in a shell script, they must be  
 100135 quoted; otherwise, the shell treats the `<backslash>` as a special character.

100136 Conforming applications should only use the specific `<backslash>`-escaped delimiters presented  
 100137 in this volume of POSIX.1-2008. Historical implementations treat `'\x'`, where `'x'` is not in this  
 100138 list, as `'x'`, but future implementations are free to expand this list to recognize other common  
 100139 escapes similar to those accepted by *printf* and other standard utilities.

100140 Most of the standard utilities work on text files. The *cut* utility can be used to turn files with  
 100141 arbitrary line lengths into a set of text files containing the same data. The *paste* utility can be used  
 100142 to create (or recreate) files with arbitrary line lengths. For example, if *file* contains long lines:

```
100143 cut -b 1-500 -n file > file1
100144 cut -b 501- -n file > file2
```

100145 creates **file1** (a text file) with lines no longer than 500 bytes (plus the `<newline>`) and **file2** that  
 100146 contains the remainder of the data from *file*. Note that **file2** is not a text file if there are lines in  
 100147 *file* that are longer than 500 + `{LINE_MAX}` bytes. The original file can be recreated from **file1**  
 100148 and **file2** using the command:

```
100149 paste -d "\0" file1 file2 > file
```

100150 The commands:

```
100151 paste -d "\0" ...
100152 paste -d "" ...
```

100153 are not necessarily equivalent; the latter is not specified by this volume of POSIX.1-2008 and  
 100154 may result in an error. The construct `'\0'` is used to mean “no separator” because historical  
 100155 versions of *paste* did not follow the syntax guidelines, and the command:

```
100156 paste -d"" ...
```

100157 could not be handled properly by *getopt()*.

## 100158 EXAMPLES

100159 1. Write out a directory in four columns:

```
100160 ls | paste - - - -
```

100161 2. Combine pairs of lines from a file into single lines:

```
100162 paste -s -d "\t\n" file
```

## 100163 RATIONALE

100164 None.

## 100165 FUTURE DIRECTIONS

100166 None.



100167 **SEE ALSO**100168 [Section 1.4](#) (on page 2310), *cut*, *grep*, *pr*100169 XBD [Chapter 7](#) (on page 135), [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 215)100170 **CHANGE HISTORY**

100171 First released in Issue 2.

100172 **Issue 6**

100173 The normative text is reworded to avoid use of the term “must” for application requirements.

100174 **Issue 7**

100175 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

## 100176 NAME

100177 patch — apply changes to files

## 100178 SYNOPSIS

100179 patch [-blNR] [-c|-e|-n|-u] [-d *dir*] [-D *define*] [-i *patchfile*]  
 100180 [-o *outfile*] [-p *num*] [-r *rejectfile*] [*file*]

## 100181 DESCRIPTION

100182 The *patch* utility shall read a source (patch) file containing any of four forms of difference (diff)  
 100183 listings produced by the *diff* utility (normal, copied context, unified context, or in the style of *ed*)  
 100184 and apply those differences to a file. By default, *patch* shall read from the standard input.

100185 The *patch* utility shall attempt to determine the type of the *diff* listing, unless overruled by a *-c*,  
 100186 *-e*, *-n*, or *-u* option.

100187 If the patch file contains more than one patch, *patch* shall attempt to apply each of them as if they  
 100188 came from separate patch files. (In this case, the application shall ensure that the name of the  
 100189 patch file is determinable for each *diff* listing.)

## 100190 OPTIONS

100191 The *patch* utility shall conform to XBD [Section 12.2](#) (on page 215).

100192 The following options shall be supported:

100193 *-b* Save a copy of the original contents of each modified file, before the differences are  
 100194 applied, in a file of the same name with the suffix **.orig** appended to it. If the file  
 100195 already exists, it shall be overwritten; if multiple patches are applied to the same  
 100196 file, the **.orig** file shall be written only for the first patch. When the *-o outfile* option  
 100197 is also specified, *file.orig* shall not be created but, if *outfile* already exists, *outfile.orig*  
 100198 shall be created.

100199 *-c* Interpret the patch file as a copied context difference (the output of the utility *diff*  
 100200 when the *-c* or *-C* options are specified).

100201 *-d dir* Change the current directory to *dir* before processing as described in the  
 100202 EXTENDED DESCRIPTION section.

100203 *-D define* Mark changes with one of the following C preprocessor constructs:

```
100204 #ifdef define
100205 ...
100206 #endif

100207 #ifndef define
100208 ...
100209 #endif
```

100210 optionally combined with the C preprocessor construct **#else**. If the patched file is  
 100211 processed with the C preprocessor, where the macro *define* is defined, the output  
 100212 shall contain the changes from the patch file; otherwise, the output shall not  
 100213 contain the patches specified in the patch file.

100214 *-e* Interpret the patch file as an *ed* script, rather than a *diff* script.

100215 *-i patchfile* Read the patch information from the file named by the pathname *patchfile*, rather  
 100216 than the standard input.

100217 *-l* (The letter ell.) Cause any sequence of <blank> characters in the difference script to  
 100218 match any sequence of <blank> characters in the input file. Other characters shall  
 100219 be matched exactly.

|        |                      |                                                                                                      |
|--------|----------------------|------------------------------------------------------------------------------------------------------|
| 100220 | <b>-n</b>            | Interpret the script as a normal difference.                                                         |
| 100221 | <b>-N</b>            | Ignore patches where the differences have already been applied to the file; by                       |
| 100222 |                      | default, already-applied patches shall be rejected.                                                  |
| 100223 | <b>-o outfile</b>    | Instead of modifying the files (specified by the <i>file</i> operand or the difference               |
| 100224 |                      | listings) directly, write a copy of the file referenced by each patch, with the                      |
| 100225 |                      | appropriate differences applied, to <i>outfile</i> . Multiple patches for a single file shall be     |
| 100226 |                      | applied to the intermediate versions of the file created by any previous patches,                    |
| 100227 |                      | and shall result in multiple, concatenated versions of the file being written to                     |
| 100228 |                      | <i>outfile</i> .                                                                                     |
| 100229 | <b>-p num</b>        | For all pathnames in the patch file that indicate the names of files to be patched,                  |
| 100230 |                      | delete <i>num</i> pathname components from the beginning of each pathname. If the                    |
| 100231 |                      | pathname in the patch file is absolute, any leading <slash> characters shall be                      |
| 100232 |                      | considered the first component (that is, <b>-p 1</b> shall remove the leading <slash>                |
| 100233 |                      | characters). Specifying <b>-p 0</b> shall cause the full pathname to be used. If <b>-p</b> is not    |
| 100234 |                      | specified, only the basename (the final pathname component) shall be used.                           |
| 100235 | <b>-R</b>            | Reverse the sense of the patch script; that is, assume that the difference script was                |
| 100236 |                      | created from the new version to the old version. The <b>-R</b> option cannot be used                 |
| 100237 |                      | with <i>ed</i> scripts. The <i>patch</i> utility shall attempt to reverse each portion of the script |
| 100238 |                      | before applying it. Rejected differences shall be saved in swapped format. If this                   |
| 100239 |                      | option is not specified, and until a portion of the patch file is successfully applied,              |
| 100240 |                      | <i>patch</i> attempts to apply each portion in its reversed sense as well as in its normal           |
| 100241 |                      | sense. If the attempt is successful, the user shall be prompted to determine whether                 |
| 100242 |                      | the <b>-R</b> option should be set.                                                                  |
| 100243 | <b>-r rejectfile</b> | Override the default reject filename. In the default case, the reject file shall have the            |
| 100244 |                      | same name as the output file, with the suffix <b>.rej</b> appended to it; see <a href="#">Patch</a>  |
| 100245 |                      | <a href="#">Application</a> (on page 3023).                                                          |
| 100246 | <b>-u</b>            | Interpret the patch file as a unified context difference (the output of the <i>diff</i> utility      |
| 100247 |                      | when the <b>-u</b> or <b>-U</b> options are specified).                                              |

#### 100248 OPERANDS

100249 The following operand shall be supported:

100250 *file* A pathname of a file to patch.

#### 100251 STDIN

100252 See the INPUT FILES section.

#### 100253 INPUT FILES

100254 Input files shall be text files.

#### 100255 ENVIRONMENT VARIABLES

100256 The following environment variables shall affect the execution of *patch*:

100257 *LANG* Provide a default value for the internationalization variables that are unset or null.  
 100258 (See XBD [Section 8.2](#) (on page 174) the precedence of internationalization variables  
 100259 used to determine the values of locale categories.)

100260 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
 100261 internationalization variables.

100262 *LC\_COLLATE*  
 100263 Determine the locale for the behavior of ranges, equivalence classes, and multi-  
 100264 character collating elements used in the extended regular expression defined for

|        |                                                                                                                                                                                                                                                                                                                                                                                          |                                                                                                                                                                                                                                                                                                                                                           |
|--------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 100265 |                                                                                                                                                                                                                                                                                                                                                                                          | the <b>yesexpr</b> locale keyword in the <i>LC_MESSAGES</i> category.                                                                                                                                                                                                                                                                                     |
| 100266 | <i>LC_CTYPE</i>                                                                                                                                                                                                                                                                                                                                                                          | Determine the locale for the interpretation of sequences of bytes of text data as characters (for example, single-byte as opposed to multi-byte characters in arguments and input files), and the behavior of character classes used in the extended regular expression defined for the <b>yesexpr</b> locale keyword in the <i>LC_MESSAGES</i> category. |
| 100267 |                                                                                                                                                                                                                                                                                                                                                                                          |                                                                                                                                                                                                                                                                                                                                                           |
| 100268 |                                                                                                                                                                                                                                                                                                                                                                                          |                                                                                                                                                                                                                                                                                                                                                           |
| 100269 |                                                                                                                                                                                                                                                                                                                                                                                          |                                                                                                                                                                                                                                                                                                                                                           |
| 100270 |                                                                                                                                                                                                                                                                                                                                                                                          |                                                                                                                                                                                                                                                                                                                                                           |
| 100271 | <i>LC_MESSAGES</i>                                                                                                                                                                                                                                                                                                                                                                       |                                                                                                                                                                                                                                                                                                                                                           |
| 100272 |                                                                                                                                                                                                                                                                                                                                                                                          | Determine the locale used to process affirmative responses, and the locale used to affect the format and contents of diagnostic messages and prompts written to standard error.                                                                                                                                                                           |
| 100273 |                                                                                                                                                                                                                                                                                                                                                                                          |                                                                                                                                                                                                                                                                                                                                                           |
| 100274 |                                                                                                                                                                                                                                                                                                                                                                                          |                                                                                                                                                                                                                                                                                                                                                           |
| 100275 | XSI <i>NLSPATH</i>                                                                                                                                                                                                                                                                                                                                                                       | Determine the location of message catalogs for the processing of <i>LC_MESSAGES</i> .                                                                                                                                                                                                                                                                     |
| 100276 | <i>LC_TIME</i>                                                                                                                                                                                                                                                                                                                                                                           | Determine the locale for recognizing the format of file timestamps written by the <i>diff</i> utility in a context-difference input file.                                                                                                                                                                                                                 |
| 100277 |                                                                                                                                                                                                                                                                                                                                                                                          |                                                                                                                                                                                                                                                                                                                                                           |
| 100278 | <b>ASYNCHRONOUS EVENTS</b>                                                                                                                                                                                                                                                                                                                                                               |                                                                                                                                                                                                                                                                                                                                                           |
| 100279 | Default.                                                                                                                                                                                                                                                                                                                                                                                 |                                                                                                                                                                                                                                                                                                                                                           |
| 100280 | <b>STDOUT</b>                                                                                                                                                                                                                                                                                                                                                                            |                                                                                                                                                                                                                                                                                                                                                           |
| 100281 | Not used.                                                                                                                                                                                                                                                                                                                                                                                |                                                                                                                                                                                                                                                                                                                                                           |
| 100282 | <b>STDERR</b>                                                                                                                                                                                                                                                                                                                                                                            |                                                                                                                                                                                                                                                                                                                                                           |
| 100283 | The standard error shall be used for diagnostic and informational messages.                                                                                                                                                                                                                                                                                                              |                                                                                                                                                                                                                                                                                                                                                           |
| 100284 | <b>OUTPUT FILES</b>                                                                                                                                                                                                                                                                                                                                                                      |                                                                                                                                                                                                                                                                                                                                                           |
| 100285 | The output of the <i>patch</i> utility, the save files ( <b>.orig</b> suffixes), and the reject files ( <b>.rej</b> suffixes) shall be text files.                                                                                                                                                                                                                                       |                                                                                                                                                                                                                                                                                                                                                           |
| 100286 |                                                                                                                                                                                                                                                                                                                                                                                          |                                                                                                                                                                                                                                                                                                                                                           |
| 100287 | <b>EXTENDED DESCRIPTION</b>                                                                                                                                                                                                                                                                                                                                                              |                                                                                                                                                                                                                                                                                                                                                           |
| 100288 | A patch file may contain patching instructions for more than one file; filenames shall be determined as specified in <a href="#">Filename Determination</a> (on page 3023). When the <b>-b</b> option is specified, for each patched file, the original shall be saved in a file of the same name with the suffix <b>.orig</b> appended to it.                                           |                                                                                                                                                                                                                                                                                                                                                           |
| 100289 |                                                                                                                                                                                                                                                                                                                                                                                          |                                                                                                                                                                                                                                                                                                                                                           |
| 100290 |                                                                                                                                                                                                                                                                                                                                                                                          |                                                                                                                                                                                                                                                                                                                                                           |
| 100291 |                                                                                                                                                                                                                                                                                                                                                                                          |                                                                                                                                                                                                                                                                                                                                                           |
| 100292 | For each patched file, a reject file may also be created as noted in <a href="#">Patch Application</a> (on page 3023). In the absence of a <b>-r</b> option, the name of this file shall be formed by appending the suffix <b>.rej</b> to the original filename.                                                                                                                         |                                                                                                                                                                                                                                                                                                                                                           |
| 100293 |                                                                                                                                                                                                                                                                                                                                                                                          |                                                                                                                                                                                                                                                                                                                                                           |
| 100294 |                                                                                                                                                                                                                                                                                                                                                                                          |                                                                                                                                                                                                                                                                                                                                                           |
| 100295 | <b>Patch File Format</b>                                                                                                                                                                                                                                                                                                                                                                 |                                                                                                                                                                                                                                                                                                                                                           |
| 100296 | The patch file shall contain zero or more lines of header information followed by one or more patches. Each patch shall contain zero or more lines of filename identification in the format produced by the <b>-c</b> , <b>-C</b> , <b>-u</b> , or <b>-U</b> options of the <i>diff</i> utility, and one or more sets of <i>diff</i> output, which are customarily called <i>hunks</i> . |                                                                                                                                                                                                                                                                                                                                                           |
| 100297 |                                                                                                                                                                                                                                                                                                                                                                                          |                                                                                                                                                                                                                                                                                                                                                           |
| 100298 |                                                                                                                                                                                                                                                                                                                                                                                          |                                                                                                                                                                                                                                                                                                                                                           |
| 100299 |                                                                                                                                                                                                                                                                                                                                                                                          |                                                                                                                                                                                                                                                                                                                                                           |
| 100300 | The <i>patch</i> utility shall recognize the following expression in the header information:                                                                                                                                                                                                                                                                                             |                                                                                                                                                                                                                                                                                                                                                           |
| 100301 | <b>Index:</b> <i>pathname</i>                                                                                                                                                                                                                                                                                                                                                            |                                                                                                                                                                                                                                                                                                                                                           |
| 100302 | The file to be patched is named <i>pathname</i> .                                                                                                                                                                                                                                                                                                                                        |                                                                                                                                                                                                                                                                                                                                                           |
| 100303 | If all lines (including headers) within a patch begin with the same leading sequence of <blank> characters, the <i>patch</i> utility shall remove this sequence before proceeding. Within each patch, if the type of difference is common context, the <i>patch</i> utility shall recognize the following expressions:                                                                   |                                                                                                                                                                                                                                                                                                                                                           |
| 100304 |                                                                                                                                                                                                                                                                                                                                                                                          |                                                                                                                                                                                                                                                                                                                                                           |
| 100305 |                                                                                                                                                                                                                                                                                                                                                                                          |                                                                                                                                                                                                                                                                                                                                                           |
| 100306 |                                                                                                                                                                                                                                                                                                                                                                                          |                                                                                                                                                                                                                                                                                                                                                           |

100307 \*\*\* *filename timestamp*  
 100308 The patches arose from *filename*.

100309 --- *filename timestamp*  
 100310 The patches should be applied to *filename*.

100311 If the type of difference is unified context, the *patch* utility shall recognize the following  
 100312 expressions:

100313 --- *filename timestamp*  
 100314 The patches arose from *filename*.

100315 +++ *filename timestamp*  
 100316 The patches should be applied to *filename*.

100317 Each hunk within a patch shall be the *diff* output to change a line range within the original file.  
 100318 The line numbers for successive hunks within a patch shall occur in ascending order.

### 100319 Filename Determination

100320 If no *file* operand is specified, *patch* shall perform the following steps to determine the filename  
 100321 to use:

- 100322 1. If the type of *diff* is context, the *patch* utility shall delete pathname components (as  
 100323 specified by the **-p** option) from the filename on the line beginning with "\*\*\*\*" (if copied  
 100324 context) or "---" (if unified context), then test for the existence of this file relative to the  
 100325 current directory (or the directory specified with the **-d** option). If the file exists, the *patch*  
 100326 utility shall use this filename.
- 100327 2. If the type of *diff* is context, the *patch* utility shall delete the pathname components (as  
 100328 specified by the **-p** option) from the filename on the line beginning with "---" (if copied  
 100329 context) or "+++" (if unified context), then test for the existence of this file relative to the  
 100330 current directory (or the directory specified with the **-d** option). If the file exists, the *patch*  
 100331 utility shall use this filename.
- 100332 3. If the header information contains a line beginning with the string **Index:**, the *patch* utility  
 100333 shall delete pathname components (as specified by the **-p** option) from this line, then test  
 100334 for the existence of this file relative to the current directory (or the directory specified  
 100335 with the **-d** option). If the file exists, the *patch* utility shall use this filename.
- 100336 XSI 4. If an **SCCS** directory exists in the current directory, *patch* shall attempt to perform a **get -e**  
 100337 **SCCS/s.filename** command to retrieve an editable version of the file. If the file exists, the  
 100338 *patch* utility shall use this filename.
- 100339 5. The *patch* utility shall write a prompt to standard output and request a filename  
 100340 interactively from the controlling terminal (for example, **/dev/tty**).

### 100341 Patch Application

100342 If the **-c**, **-e**, **-n**, or **-u** option is present, the *patch* utility shall interpret information within each  
 100343 hunk as a copied context difference, an *ed* difference, a normal difference, or a unified context  
 100344 difference, respectively. In the absence of any of these options, the *patch* utility shall determine  
 100345 the type of difference based on the format of information within the hunk.

100346 For each hunk, the *patch* utility shall begin to search for the place to apply the patch at the line  
 100347 number at the beginning of the hunk, plus or minus any offset used in applying the previous  
 100348 hunk. If lines matching the hunk context are not found, *patch* shall scan both forwards and  
 100349 backwards at least 1 000 bytes for a set of lines that match the hunk context.

100350 If no such place is found and it is a context difference, then another scan shall take place,  
100351 ignoring the first and last line of context. If that fails, the first two and last two lines of context  
100352 shall be ignored and another scan shall be made. Implementations may search more extensively  
100353 for installation locations.

100354 If no location can be found, the *patch* utility shall append the hunk to the reject file. A rejected  
100355 hunk that is a copied context difference, an *ed* difference, or a normal difference shall be written  
100356 in copied-context-difference format regardless of the format of the patch file. It is  
100357 implementation-defined whether a rejected hunk that is a unified context difference is written in  
100358 copied-context-difference format or in unified-context-difference format. If the input was a  
100359 normal or *ed*-style difference, the reject file may contain differences with zero lines of context.  
100360 The line numbers on the hunks in the reject file may be different from the line numbers in the  
100361 patch file since they shall reflect the approximate locations for the failed hunks in the new file  
100362 rather than the old one.

100363 If the type of patch is an *ed* diff, the implementation may accomplish the patching by invoking  
100364 the *ed* utility.

#### 100365 EXIT STATUS

100366 The following exit values shall be returned:

- 100367 0 Successful completion.
- 100368 1 One or more lines were written to a reject file.
- 100369 >1 An error occurred.

#### 100370 CONSEQUENCES OF ERRORS

100371 Patches that cannot be correctly placed in the file shall be written to a reject file.

#### 100372 APPLICATION USAGE

100373 The **-R** option does not work with *ed* scripts because there is too little information to reconstruct  
100374 the reverse operation.

100375 The **-p** option makes it possible to customize a patch file to local user directory structures  
100376 without manually editing the patch file. For example, if the filename in the patch file was:

100377 `/curds/whey/src/blurfl/blurfl.c`

100378 Setting **-p 0** gives the entire pathname unmodified; **-p 1** gives:

100379 `curds/whey/src/blurfl/blurfl.c`

100380 without the leading <slash>, **-p 4** gives:

100381 `blurfl/blurfl.c`

100382 and not specifying **-p** at all gives:

100383 `blurfl.c` .

#### 100384 EXAMPLES

100385 None.

#### 100386 RATIONALE

100387 Some of the functionality in historical *patch* implementations was not specified. The following  
100388 documents those features present in historical implementations that have not been specified.

100389 A deleted piece of functionality was the '+' pseudo-option allowing an additional set of options  
100390 and a patch file operand to be given. This was seen as being insufficiently useful to standardize.

100391 In historical implementations, if the string "Prereq:" appeared in the header, the *patch* utility

would search for the corresponding version information (the string specified in the header, delimited by <blank> characters or the beginning or end of a line or the file) anywhere in the original file. This was deleted as too simplistic and insufficiently trustworthy a mechanism to standardize. For example, if:

Prereq: 1.2

were in the header, the presence of a delimited 1.2 anywhere in the file would satisfy the prerequisite.

The following options were dropped from historical implementations of *patch* as insufficiently useful to standardize:

**-b** The **-b** option historically provided a method for changing the name extension of the backup file from the default **.orig**. This option has been modified and retained in this volume of POSIX.1-2008.

**-F** The **-F** option specified the number of lines of a context diff to ignore when searching for a place to install a patch.

**-f** The **-f** option historically caused *patch* not to request additional information from the user.

**-r** The **-r** option historically provided a method of overriding the extension of the reject file from the default **.rej**.

**-s** The **-s** option historically caused *patch* to work silently unless an error occurred.

**-x** The **-x** option historically set internal debugging flags.

In some file system implementations, the saving of a **.orig** file may produce unwanted results. In the case of 12, 13, or 14-character filenames (on file systems supporting 14-character maximum filenames), the **.orig** file overwrites the new file. The reject file may also exceed this filename limit. It was suggested, due to some historical practice, that a <tilde> ('~') suffix be used instead of **.orig** and some other character instead of the **.rej** suffix. This was rejected because it is not obvious to the user which file is which. The suffixes **.orig** and **.rej** are clearer and more understandable.

The **-b** option has the opposite sense in some historical implementations—do not save the **.orig** file. The default case here is not to save the files, making *patch* behave more consistently with the other standard utilities.

The **-w** option in early proposals was changed to **-l** to match historical practice.

The **-N** option was included because without it, a non-interactive application cannot reject previously applied patches. For example, if a user is piping the output of *diff* into the *patch* utility, and the user only wants to patch a file to a newer version non-interactively, the **-N** option is required.

Changes to the **-l** option description were proposed to allow matching across <newline> characters in addition to just <blank> characters. Since this is not historical practice, and since some ambiguities could result, it is suggested that future developments in this area utilize another option letter, such as **-L**.

The **-u** option of GNU *patch* has been added, along with support for unified context formats.

100432 **FUTURE DIRECTIONS**

100433 None.

100434 **SEE ALSO**100435 *diff, ed*100436 XBD [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 215)100437 **CHANGE HISTORY**

100438 First released in Issue 4.

100439 **Issue 5**

100440 The FUTURE DIRECTIONS section is added.

100441 **Issue 6**

100442 This utility is marked as part of the User Portability Utilities option.

100443 The description of the **-D** option and the steps in [Filename Determination](#) (on page 3023) are  
100444 changed to match historical practice as defined in the IEEE P1003.2b draft standard.

100445 The normative text is reworded to avoid use of the term “must” for application requirements.

100446 IEEE Std 1003.1-2001/Cor 1-2002, item XCU/TC1/D6/34 is applied, clarifying the way that the  
100447 *patch* utility performs **ifdef** selection for the **-D** option.100448 **Issue 7**100449 The *patch* utility is moved from the User Portability Utilities option to the Base. User Portability  
100450 Utilities is now an option for interactive utilities.

100451 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

100452 SD5-XCU-ERN-103 and SD5-XCU-ERN-120 are applied, adding the **-u** option.100453 Austin Group Interpretation 1003.1-2001 #126 is applied, changing the description of the  
100454 *LC\_MESSAGES* and *LC\_CTYPE* environment variables and adding the *LC\_COLLATE*  
100455 environment variable.



100456 **NAME**

100457 pathchk — check pathnames

100458 **SYNOPSIS**100459 pathchk [-p] [-P] *pathname*...100460 **DESCRIPTION**

100461 The *pathchk* utility shall check that one or more pathnames are valid (that is, they could be used  
 100462 to access or create a file without causing syntax errors) and portable (that is, no filename  
 100463 truncation results). More extensive portability checks are provided by the **-p** and **-P** options.

100464 By default, the *pathchk* utility shall check each component of each *pathname* operand based on the  
 100465 underlying file system. A diagnostic shall be written for each *pathname* operand that:

- 100466 • Is longer than {PATH\_MAX} bytes (see **Pathname Variable Values** in XBD [<limits.h>](#))
- 100467 • Contains any component longer than {NAME\_MAX} bytes in its containing directory
- 100468 • Contains any component in a directory that is not searchable
- 100469 • Contains any byte sequence that is not valid in its containing directory

100470 The format of the diagnostic message is not specified, but shall indicate the error detected and  
 100471 the corresponding *pathname* operand.

100472 It shall not be considered an error if one or more components of a *pathname* operand do not exist  
 100473 as long as a file matching the pathname specified by the missing components could be created  
 100474 that does not violate any of the checks specified above.

100475 **OPTIONS**100476 The *pathchk* utility shall conform to XBD [Section 12.2](#) (on page 215).

100477 The following option shall be supported:

100478 **-p** Instead of performing checks based on the underlying file system, write a  
 100479 diagnostic for each *pathname* operand that:

- 100480 • Is longer than {\_POSIX\_PATH\_MAX} bytes (see **Minimum Values** in XBD  
 100481 [<limits.h>](#))
- 100482 • Contains any component longer than {\_POSIX\_NAME\_MAX} bytes
- 100483 • Contains any character in any component that is not in the portable filename  
 100484 character set

100485 **-P** Write a diagnostic for each *pathname* operand that:

- 100486 • Contains a component whose first character is the <hyphen> character
- 100487 • Is empty

100488 **OPERANDS**

100489 The following operand shall be supported:

100490 *pathname* A pathname to be checked.100491 **STDIN**

100492 Not used.

100493 **INPUT FILES**

100494 None.

100495 **ENVIRONMENT VARIABLES**100496 The following environment variables shall affect the execution of *pathchk*:

100497 *LANG* Provide a default value for the internationalization variables that are unset or null.  
 100498 (See XBD [Section 8.2](#) (on page 174) the precedence of internationalization variables  
 100499 used to determine the values of locale categories.)

100500 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
 100501 internationalization variables.

100502 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
 100503 characters (for example, single-byte as opposed to multi-byte characters in  
 100504 arguments).

100505 *LC\_MESSAGES*

100506 Determine the locale that should be used to affect the format and contents of  
 100507 diagnostic messages written to standard error.

100508 *XSI NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

100509 **ASYNCHRONOUS EVENTS**

100510 Default.

100511 **STDOUT**

100512 Not used.

100513 **STDERR**

100514 The standard error shall be used only for diagnostic messages.

100515 **OUTPUT FILES**

100516 None.

100517 **EXTENDED DESCRIPTION**

100518 None.

100519 **EXIT STATUS**

100520 The following exit values shall be returned:

100521 0 All *pathname* operands passed all of the checks.

100522 &gt;0 An error occurred.

100523 **CONSEQUENCES OF ERRORS**

100524 Default.

100525 **APPLICATION USAGE**

100526 The *test* utility can be used to determine whether a given pathname names an existing file; it  
 100527 does not, however, give any indication of whether or not any component of the pathname was  
 100528 truncated in a directory where the `_POSIX_NO_TRUNC` feature is not in effect. The *pathchk*  
 100529 utility does not check for file existence; it performs checks to determine whether a pathname  
 100530 does exist or could be created with no pathname component truncation.

100531 The *noclobber* option in the shell (see the [set](#) special built-in) can be used to atomically create a  
 100532 file. As with all file creation semantics in the System Interfaces volume of POSIX.1-2008, it  
 100533 guarantees atomic creation, but still depends on applications to agree on conventions and  
 100534 cooperate on the use of files after they have been created.

100535 To verify that a pathname meets the requirements of filename portability, applications should  
 100536 use both the `-p` and `-P` options together.

**EXAMPLES**

To verify that all pathnames in an imported data interchange archive are legitimate and unambiguous on the current system:

```
This example assumes that no pathnames in the archive
contain <newline> characters.
pax -f archive | sed -e 's/^[[:alnum:]]/\\&/g' | xargs pathchk --
if [$? -eq 0]
then
 pax -r -f archive
else
 echo Investigate problems before importing files.
 exit 1
fi
```

To verify that all files in the current directory hierarchy could be moved to any system conforming to the System Interfaces volume of POSIX.1-2008 that also supports the *pax* utility:

```
find . -exec pathchk -p -P {} +
if [$? -eq 0]
then
 pax -w -f ../archive .
else
 echo Portable archive cannot be created.
 exit 1
fi
```

To verify that a user-supplied pathname names a readable file and that the application can create a file extending the given path without truncation and without overwriting any existing file:

```
case $- in
 C) reset="";;
 *) reset="set +C"
 set -C;;
esac
test -r "$path" && pathchk "$path.out" &&
rm "$path.out" > "$path.out"
if [$? -ne 0]; then
 printf "%s: %s not found or %s.out fails \
creation checks.\n" $0 "$path" "$path"
 $reset # Reset the noclobber option in case a trap
 # on EXIT depends on it.
 exit 1
fi
$reset
PROCESSING < "$path" > "$path.out"
```

The following assumptions are made in this example:

1. **PROCESSING** represents the code that is used by the application to use **\$path** once it is verified that **\$path.out** works as intended.
2. The state of the *noclobber* option is unknown when this code is invoked and should be set on exit to the state it was in when this code was invoked. (The **reset** variable is used in this example to restore the initial state.)

## 3. Note the usage of:

```
rm "$path.out" > "$path.out"
```

- a. The *pathchk* command has already verified, at this point, that **\$path.out** is not truncated.
- b. With the *noclobber* option set, the shell verifies that **\$path.out** does not already exist before invoking *rm*.
- c. If the shell succeeded in creating **\$path.out**, *rm* removes it so that the application can create the file again in the **PROCESSING** step.
- d. If the **PROCESSING** step wants the file to exist already when it is invoked, the:

```
rm "$path.out" > "$path.out"
```

should be replaced with:

```
> "$path.out"
```

which verifies that the file did not already exist, but leaves **\$path.out** in place for use by **PROCESSING**.

**RATIONALE**

The *pathchk* utility was new for the ISO POSIX-2:1993 standard. It, along with the *set -C(noclobber)* option added to the shell, replaces the *mktemp*, *validfnam*, and *create* utilities that appeared in early proposals. All of these utilities were attempts to solve several common problems:

- Verify the validity (for several different definitions of “valid”) of a pathname supplied by a user, generated by an application, or imported from an external source.
- Atomically create a file.
- Perform various string handling functions to generate a temporary filename.

The *create* utility, included in an early proposal, provided checking and atomic creation in a single invocation of the utility; these are orthogonal issues and need not be grouped into a single utility. Note that the *noclobber* option also provides a way of creating a lock for process synchronization; since it provides an atomic *create*, there is no race between a test for existence and the following creation if it did not exist.

Having a function like *tmpnam()* in the ISO C standard is important in many high-level languages. The shell programming language, however, has built-in string manipulation facilities, making it very easy to construct temporary filenames. The names needed obviously depend on the application, but are frequently of a form similar to:

```
$TMPDIR/application_abbreviation$$suffix
```

In cases where there is likely to be contention for a given suffix, a simple shell **for** or **while** loop can be used with the shell *noclobber* option to create a file without risk of collisions, as long as applications trying to use the same filename name space are cooperating on the use of files after they have been created.

For historical purposes, **-p** does not check for the use of the <hyphen> character as the first character in a component of the pathname, or for an empty *pathname* operand.

100623 **FUTURE DIRECTIONS**

100624 None.

100625 **SEE ALSO**100626 [Section 2.7](#) (on page 2334), [set](#) (on page 2380), [test](#)100627 XBD [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 215), [<limits.h>](#)100628 **CHANGE HISTORY**

100629 First released in Issue 4.

100630 **Issue 7**

100631 Austin Group Interpretations 1003.1-2001 #039, #040, and #094 are applied.

100632 SD5-XCU-ERN-121 is applied, updating the EXAMPLES section.

100633 POSIX.1-2008, Technical Corrigendum 1, XCU/TC1-2008/0127 [291] is applied.

+

## 100634 NAME

100635 pax — portable archive interchange

## 100636 SYNOPSIS

100637 pax [-dv] [-c|-n] [-H|-L] [-o options] [-f archive] [-s replstr]...  
 100638 [pattern...]

100639 pax -r[-c|-n] [-dikuv] [-H|-L] [-f archive] [-o options]... [-p string]...  
 100640 [-s replstr]... [pattern...]

100641 pax -w [-dituvX] [-H|-L] [-b blocksize] [[-a] [-f archive]] [-o options]...  
 100642 [-s replstr]... [-x format] [file...]

100643 pax -r -w [-diklntuvX] [-H|-L] [-o options]... [-p string]...  
 100644 [-s replstr]... [file...] directory

## 100645 DESCRIPTION

100646 The *pax* utility shall read, write, and write lists of the members of archive files and copy  
 100647 directory hierarchies. A variety of archive formats shall be supported; see the *-x format* option.

100648 The action to be taken depends on the presence of the *-r* and *-w* options. The four combinations  
 100649 of *-r* and *-w* are referred to as the four modes of operation: **list**, **read**, **write**, and **copy** modes,  
 100650 corresponding respectively to the four forms shown in the SYNOPSIS section.

100651 **list** In **list** mode (when neither *-r* nor *-w* are specified), *pax* shall write the names of  
 100652 the members of the archive file read from the standard input, with pathnames  
 100653 matching the specified patterns, to standard output. If a named file is of type  
 100654 directory, the file hierarchy rooted at that file shall be listed as well.

100655 **read** In **read** mode (when *-r* is specified, but *-w* is not), *pax* shall extract the members of  
 100656 the archive file read from the standard input, with pathnames matching the  
 100657 specified patterns. If an extracted file is of type directory, the file hierarchy rooted  
 100658 at that file shall be extracted as well. The extracted files shall be created performing  
 100659 pathname resolution with the directory in which *pax* was invoked as the current  
 100660 working directory.

100661 If an attempt is made to extract a directory when the directory already exists, this  
 100662 shall not be considered an error. If an attempt is made to extract a FIFO when the  
 100663 FIFO already exists, this shall not be considered an error.

100664 The ownership, access, and modification times, and file mode of the restored files  
 100665 are discussed under the *-p* option.

100666 **write** In **write** mode (when *-w* is specified, but *-r* is not), *pax* shall write the contents of  
 100667 the *file* operands to the standard output in an archive format. If no *file* operands are  
 100668 specified, a list of files to copy, one per line, shall be read from the standard input  
 100669 and each entry in this list shall be processed as if it had been a *file* operand on the  
 100670 command line. A file of type directory shall include all of the files in the file  
 100671 hierarchy rooted at the file.

100672 **copy** In **copy** mode (when both *-r* and *-w* are specified), *pax* shall copy the *file* operands  
 100673 to the destination directory.

100674 If no *file* operands are specified, a list of files to copy, one per line, shall be read  
 100675 from the standard input. A file of type directory shall include all of the files in the  
 100676 file hierarchy rooted at the file.

100677 The effect of the **copy** shall be as if the copied files were written to a *pax* format  
 100678 archive file and then subsequently extracted, except that there may be hard links

100679                   between the original and the copied files. If the destination directory is a  
 100680                   subdirectory of one of the files to be copied, the results are unspecified. If the  
 100681                   destination directory is a file of a type not defined by the System Interfaces volume  
 100682                   of POSIX.1-2008, the results are implementation-defined; otherwise, it shall be an  
 100683                   error for the file named by the *directory* operand not to exist, not be writable by the  
 100684                   user, or not be a file of type directory.

100685           In **read** or **copy** modes, if intermediate directories are necessary to extract an archive member,  
 100686           *pax* shall perform actions equivalent to the *mkdir()* function defined in the System Interfaces  
 100687           volume of POSIX.1-2008, called with the following arguments:

- 100688           • The intermediate directory used as the *path* argument
- 100689           • The value of the bitwise-inclusive OR of S\_IRWXU, S\_IRWXG, and S\_IRWXO as the *mode*  
 100690           argument

100691           If any specified *pattern* or *file* operands are not matched by at least one file or archive member,  
 100692           *pax* shall write a diagnostic message to standard error for each one that did not match and exit  
 100693           with a non-zero exit status.

100694           The archive formats described in the EXTENDED DESCRIPTION section shall be automatically  
 100695           detected on input. The default output archive format shall be implementation-defined.

100696           A single archive can span multiple files. The *pax* utility shall determine, in an implementation-  
 100697           defined manner, what file to read or write as the next file.

100698           If the selected archive format supports the specification of linked files, it shall be an error if these  
 100699           files cannot be linked when the archive is extracted. For archive formats that do not store file  
 100700           contents with each name that causes a hard link, if the file that contains the data is not extracted  
 100701           during this *pax* session, either the data shall be restored from the original file, or a diagnostic  
 100702           message shall be displayed with the name of a file that can be used to extract the data. In  
 100703           traversing directories, *pax* shall detect infinite loops; that is, entering a previously visited  
 100704           directory that is an ancestor of the last file visited. When it detects an infinite loop, *pax* shall  
 100705           write a diagnostic message to standard error and shall terminate.

## 100706 OPTIONS

100707           The *pax* utility shall conform to XBD [Section 12.2](#) (on page 215), except that the order of  
 100708           presentation of the **-o**, **-p**, and **-s** options is significant.

100709           The following options shall be supported:

- 100710           **-r**           Read an archive file from standard input.
- 100711           **-w**           Write files to the standard output in the specified archive format.
- 100712           **-a**           Append files to the end of the archive. It is implementation-defined which devices  
 100713           on the system support appending. Additional file formats unspecified by this  
 100714           volume of POSIX.1-2008 may impose restrictions on appending.
- 100715           **-b** *blocksize*   Block the output at a positive decimal integer number of bytes per write to the  
 100716           archive file. Devices and archive formats may impose restrictions on blocking.  
 100717           Blocking shall be automatically determined on input. Conforming applications  
 100718           shall not specify a *blocksize* value larger than 32256. Default blocking when  
 100719           creating archives depends on the archive format. (See the **-x** option below.)
- 100720           **-c**           Match all file or archive members except those specified by the *pattern* or *file*  
 100721           operands.

|        |                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
|--------|-------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 100722 | -d                | Cause files of type directory being copied or archived or archive members of type directory being extracted or listed to match only the file or archive member itself and not the file hierarchy rooted at the file.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 100723 |                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 100724 |                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 100725 | -f <i>archive</i> | Specify the pathname of the input or output archive, overriding the default standard input (in <b>list</b> or <b>read</b> modes) or standard output ( <b>write</b> mode).                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 100726 |                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 100727 | -H                | If a symbolic link referencing a file of type directory is specified on the command line, <i>pax</i> shall archive the file hierarchy rooted in the file referenced by the link, using the name of the link as the root of the file hierarchy. Otherwise, if a symbolic link referencing a file of any other file type which <i>pax</i> can normally archive is specified on the command line, then <i>pax</i> shall archive the file referenced by the link, using the name of the link. The default behavior, when neither -H or -L are specified, shall be to archive the symbolic link itself.                                                                                                                                                                                                                                     |
| 100728 |                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 100729 |                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 100730 |                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 100731 |                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 100732 |                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 100733 |                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 100734 | -i                | Interactively rename files or archive members. For each archive member matching a <i>pattern</i> operand or file matching a <i>file</i> operand, a prompt shall be written to the file <b>/dev/tty</b> . The prompt shall contain the name of the file or archive member, but the format is otherwise unspecified. A line shall then be read from <b>/dev/tty</b> . If this line is blank, the file or archive member shall be skipped. If this line consists of a single period, the file or archive member shall be processed with no modification to its name. Otherwise, its name shall be replaced with the contents of the line. The <i>pax</i> utility shall immediately exit with a non-zero exit status if end-of-file is encountered when reading a response or if <b>/dev/tty</b> cannot be opened for reading and writing. |
| 100735 |                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 100736 |                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 100737 |                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 100738 |                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 100739 |                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 100740 |                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 100741 |                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 100742 |                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 100743 |                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 100744 |                   | The results of extracting a hard link to a file that has been renamed during extraction are unspecified.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| 100745 |                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 100746 | -k                | Prevent the overwriting of existing files.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| 100747 | -l                | (The letter ell.) In <b>copy</b> mode, hard links shall be made between the source and destination file hierarchies whenever possible. If specified in conjunction with -H or -L, when a symbolic link is encountered, the hard link created in the destination file hierarchy shall be to the file referenced by the symbolic link. If specified when neither -H nor -L is specified, when a symbolic link is encountered, the implementation shall create a hard link to the symbolic link in the source file hierarchy or copy the symbolic link to the destination.                                                                                                                                                                                                                                                                |
| 100748 |                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 100749 |                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 100750 |                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 100751 |                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 100752 |                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 100753 |                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 100754 | -L                | If a symbolic link referencing a file of type directory is specified on the command line or encountered during the traversal of a file hierarchy, <i>pax</i> shall archive the file hierarchy rooted in the file referenced by the link, using the name of the link as the root of the file hierarchy. Otherwise, if a symbolic link referencing a file of any other file type which <i>pax</i> can normally archive is specified on the command line or encountered during the traversal of a file hierarchy, <i>pax</i> shall archive the file referenced by the link, using the name of the link. The default behavior, when neither -H or -L are specified, shall be to archive the symbolic link itself.                                                                                                                          |
| 100755 |                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 100756 |                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 100757 |                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 100758 |                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 100759 |                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 100760 |                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 100761 | -n                | Select the first archive member that matches each <i>pattern</i> operand. No more than one archive member shall be matched for each pattern (although members of type directory shall still match the file hierarchy rooted at that file).                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| 100762 |                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 100763 |                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 100764 |                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 100765 | -o <i>options</i> | Provide information to the implementation to modify the algorithm for extracting or writing files. The value of <i>options</i> shall consist of one or more <comma>-separated keywords of the form:                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 100766 |                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 100767 |                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 100768 |                   | <i>keyword</i> [[:]= <i>value</i> ][, <i>keyword</i> [[:]= <i>value</i> ], ...]                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |



Some keywords apply only to certain file formats, as indicated with each description. Use of keywords that are inapplicable to the file format being processed produces undefined results.

Keywords in the *options* argument shall be a string that would be a valid portable filename as described in XBD [Section 3.278](#) (on page 78).

**Note:** Keywords are not expected to be filenames, merely to follow the same character composition rules as portable filenames.

Keywords can be preceded with white space. The *value* field shall consist of zero or more characters; within *value*, the application shall precede any literal <comma> with a <backslash>, which shall be ignored, but preserves the <comma> as part of *value*. A <comma> as the final character, or a <comma> followed solely by white space as the final characters, in *options* shall be ignored. Multiple **-o** options can be specified; if keywords given to these multiple **-o** options conflict, the keywords and values appearing later in command line sequence shall take precedence and the earlier shall be silently ignored. The following keyword values of *options* shall be supported for the file formats as indicated:

**delete=pattern**

(Applicable only to the **-x pax** format.) When used in **write** or **copy** mode, *pax* shall omit from extended header records that it produces any keywords matching the string pattern. When used in **read** or **list** mode, *pax* shall ignore any keywords matching the string pattern in the extended header records. In both cases, matching shall be performed using the pattern matching notation described in [Section 2.13.1](#) (on page 2354) and [Section 2.13.2](#) (on page 2355). For example:

**-o delete=security.\***

would suppress security-related information. See [pax Extended Header](#) (on page 3045) for extended header record keyword usage.

When multiple **-odelete=pattern** options are specified, the patterns shall be additive; all keywords matching the specified string patterns shall be omitted from extended header records that *pax* produces.

**exthdr.name=string**

(Applicable only to the **-x pax** format.) This keyword allows user control over the name that is written into the **ustar** header blocks for the extended header produced under the circumstances described in [pax Header Block](#) (on page 3045). The name shall be the contents of *string*, after the following character substitutions have been made:

| <i>string</i><br>Includes: | Replaced by:                                                                                                       |
|----------------------------|--------------------------------------------------------------------------------------------------------------------|
| %d                         | The directory name of the file, equivalent to the result of the <i>dirname</i> utility on the translated pathname. |
| %f                         | The filename of the file, equivalent to the result of the <i>basename</i> utility on the translated pathname.      |
| %p                         | The process ID of the <i>pax</i> process.                                                                          |
| %%                         | A ' % ' character.                                                                                                 |

Any other ' % ' characters in *string* produce undefined results.

If no **-o exthdr.name=string** is specified, *pax* shall use the following default

value:

```
%d/PaxHeaders.%p/%f
```

#### **globexthdr.name=string**

(Applicable only to the **-x pax** format.) When used in **write** or **copy** mode with the appropriate options, *pax* shall create global extended header records with **ustar** header blocks that will be treated as regular files by previous versions of *pax*. This keyword allows user control over the name that is written into the **ustar** header blocks for global extended header records. The name shall be the contents of *string*, after the following character substitutions have been made:

| <i>string</i><br>Includes: | Replaced by:                                                                                                       |
|----------------------------|--------------------------------------------------------------------------------------------------------------------|
| %n                         | An integer that represents the sequence number of the global extended header record in the archive, starting at 1. |
| %p                         | The process ID of the <i>pax</i> process.                                                                          |
| %%                         | A '%' character.                                                                                                   |

Any other '%' characters in *string* produce undefined results.

If no **-o globexthdr.name=string** is specified, *pax* shall use the following default value:

```
$TMPDIR/GlobalHead.%p.%n
```

where *\$TMPDIR* represents the value of the *TMPDIR* environment variable. If *TMPDIR* is not set, *pax* shall use **/tmp**.

#### **invalid=action**

(Applicable only to the **-x pax** format.) This keyword allows user control over the action *pax* takes upon encountering values in an extended header record that, in **read** or **copy** mode, are invalid in the destination hierarchy or, in **list** mode, cannot be written in the codeset and current locale of the implementation. The following are invalid values that shall be recognized by *pax*:

- In **read** or **copy** mode, a filename or link name that contains character encodings invalid in the destination hierarchy. (For example, the name may contain embedded NULs.)
- In **read** or **copy** mode, a filename or link name that is longer than the maximum allowed in the destination hierarchy (for either a pathname component or the entire pathname).
- In **list** mode, any character string value (filename, link name, user name, and so on) that cannot be written in the codeset and current locale of the implementation.

The following mutually-exclusive values of the *action* argument are supported:

**binary** In **write** mode, *pax* shall generate a **hdrcharset=BINARY** extended header record for each file with a filename, link name, group name, owner name, or any other field in an extended header record that cannot be translated to the UTF-8 codeset, allowing the archive to contain the files with unencoded extended header record values. In **read** or **copy** mode, *pax* shall

|        |                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|--------|-----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 100860 |                       | use the values specified in the header without translation, regardless of whether this may overwrite an existing file with a valid name. In <b>list</b> mode, <i>pax</i> shall behave identically to the <b>bypass</b> action.                                                                                                                                                                                                                                                                                                                                                                                          |
| 100861 |                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 100862 |                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 100863 |                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 100864 | <b>bypass</b>         | In <b>read</b> or <b>copy</b> mode, <i>pax</i> shall bypass the file, causing no change to the destination hierarchy. In <b>list</b> mode, <i>pax</i> shall write all requested valid values for the file, but its method for writing invalid values is unspecified.                                                                                                                                                                                                                                                                                                                                                    |
| 100865 |                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 100866 |                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 100867 |                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 100868 | <b>rename</b>         | In <b>read</b> or <b>copy</b> mode, <i>pax</i> shall act as if the <b>-i</b> option were in effect for each file with invalid filename or link name values, allowing the user to provide a replacement name interactively. In <b>list</b> mode, <i>pax</i> shall behave identically to the <b>bypass</b> action.                                                                                                                                                                                                                                                                                                        |
| 100869 |                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 100870 |                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 100871 |                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 100872 | <b>UTF-8</b>          | When used in <b>read</b> , <b>copy</b> , or <b>list</b> mode and a filename, link name, owner name, or any other field in an extended header record cannot be translated from the <b>pax</b> UTF-8 codeset format to the codeset and current locale of the implementation, <i>pax</i> shall use the actual UTF-8 encoding for the name. If a <b>hdrcharset</b> extended header record is in effect for this file, the character set specified by that record shall be used instead of UTF-8. If a <b>hdrcharset=BINARY</b> extended header record is in effect for this file, no translation shall be performed.        |
| 100873 |                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 100874 |                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 100875 |                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 100876 |                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 100877 |                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 100878 |                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 100879 |                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 100880 |                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 100881 | <b>write</b>          | In <b>read</b> or <b>copy</b> mode, <i>pax</i> shall write the file, translating the name, regardless of whether this may overwrite an existing file with a valid name. In <b>list</b> mode, <i>pax</i> shall behave identically to the <b>bypass</b> action.                                                                                                                                                                                                                                                                                                                                                           |
| 100882 |                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 100883 |                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 100884 |                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 100885 |                       | If no <b>-o invalid=option</b> is specified, <i>pax</i> shall act as if <b>-o invalid=bypass</b> were specified. Any overwriting of existing files that may be allowed by the <b>-o invalid=</b> actions shall be subject to permission ( <b>-p</b> ) and modification time ( <b>-u</b> ) restrictions, and shall be suppressed if the <b>-k</b> option is also specified.                                                                                                                                                                                                                                              |
| 100886 |                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 100887 |                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 100888 |                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 100889 | <b>linkdata</b>       | (Applicable only to the <b>-x pax</b> format.) In <b>write</b> mode, <i>pax</i> shall write the contents of a file to the archive even when that file is merely a hard link to a file whose contents have already been written to the archive.                                                                                                                                                                                                                                                                                                                                                                          |
| 100890 |                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 100891 |                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 100892 |                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 100893 | <b>listopt=format</b> | This keyword specifies the output format of the table of contents produced when the <b>-v</b> option is specified in <b>list</b> mode. See <a href="#">List Mode Format Specifications</a> (on page 3040). To avoid ambiguity, the <b>listopt=format</b> shall be the only or final <b>keyword=value</b> pair in a <b>-o</b> option-argument; all characters in the remainder of the option-argument shall be considered part of the format string. When multiple <b>-olistopt=format</b> options are specified, the format strings shall be considered a single, concatenated string, evaluated in command line order. |
| 100894 |                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 100895 |                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 100896 |                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 100897 |                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 100898 |                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 100899 |                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 100900 |                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 100901 |                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 100902 | <b>times</b>          | (Applicable only to the <b>-x pax</b> format.) When used in <b>write</b> or <b>copy</b> mode, <i>pax</i> shall include <b>atime</b> and <b>mtime</b> extended header records for each file. See <a href="#">pax Extended Header File Times</a> (on page 3049).                                                                                                                                                                                                                                                                                                                                                          |
| 100903 |                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 100904 |                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 100905 |                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 100906 |                       | In addition to these keywords, if the <b>-x pax</b> format is specified, any of the                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |

keywords and values defined in [pax Extended Header](#) (on page 3045), including implementation extensions, can be used in **-o** option-arguments, in either of two modes:

**keyword=***value*

When used in **write** or **copy** mode, these keyword/value pairs shall be included at the beginning of the archive as **typeflag g** global extended header records. When used in **read** or **list** mode, these keyword/value pairs shall act as if they had been at the beginning of the archive as **typeflag g** global extended header records.

**keyword:=***value*

When used in **write** or **copy** mode, these keyword/value pairs shall be included as records at the beginning of a **typeflag x** extended header for each file. (This shall be equivalent to the `<equals-sign>` form except that it creates no **typeflag g** global extended header records.) When used in **read** or **list** mode, these keyword/value pairs shall act as if they were included as records at the end of each extended header; thus, they shall override any global or file-specific extended header record keywords of the same names. For example, in the command:

```
pax -r -o "
 gname:=mygroup,
 " <archive
```

the group name will be forced to a new value for all files read from the archive.

The precedence of **-o** keywords over various fields in the archive is described in [pax Extended Header Keyword Precedence](#) (on page 3048).

**-p** *string*

Specify one or more file characteristic options (privileges). The *string* option-argument shall be a string specifying file characteristics to be retained or discarded on extraction. The string shall consist of the specification characters *a*, *e*, *m*, *o*, and *p*. Other implementation-defined characters can be included. Multiple characteristics can be concatenated within the same string and multiple **-p** options can be specified. The meaning of the specification characters are as follows:

- a* Do not preserve file access times.
- e* Preserve the user ID, group ID, file mode bits (see [XBD Section 3.169](#), on page 60), access time, modification time, and any other implementation-defined file characteristics.
- m* Do not preserve file modification times.
- o* Preserve the user ID and group ID.
- p* Preserve the file mode bits. Other implementation-defined file mode attributes may be preserved.

In the preceding list, “preserve” indicates that an attribute stored in the archive shall be given to the extracted file, subject to the permissions of the invoking process. The access and modification times of the file shall be preserved unless otherwise specified with the **-p** option or not stored in the archive. All attributes that are not preserved shall be determined as part of the normal file creation action (see [Section 1.1.1.4](#), on page 2302).

|        |                         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
|--------|-------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 100952 |                         | If neither the <code>e</code> nor the <code>o</code> specification character is specified, or the user ID and group ID are not preserved for any reason, <i>pax</i> shall not set the <code>S_ISUID</code> and <code>S_ISGID</code> bits of the file mode.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| 100953 |                         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 100954 |                         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 100955 |                         | If the preservation of any of these items fails for any reason, <i>pax</i> shall write a diagnostic message to standard error. Failure to preserve these items shall affect the final exit status, but shall not cause the extracted file to be deleted.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| 100956 |                         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 100957 |                         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 100958 |                         | If file characteristic letters in any of the <i>string</i> option-arguments are duplicated or conflict with each other, the ones given last shall take precedence. For example, if <code>-p eme</code> is specified, file modification times are preserved.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 100959 |                         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 100960 |                         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 100961 | <code>-s replstr</code> | Modify file or archive member names named by <i>pattern</i> or <i>file</i> operands according to the substitution expression <i>replstr</i> , using the syntax of the <i>ed</i> utility. The concepts of “address” and “line” are meaningless in the context of the <i>pax</i> utility, and shall not be supplied. The format shall be:                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 100962 |                         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 100963 |                         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 100964 |                         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 100965 |                         | <code>-s /old/new/[gp]</code>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 100966 |                         | where as in <i>ed</i> , <i>old</i> is a basic regular expression and <i>new</i> can contain an <code>&amp;</code> and <code>'\n'</code> (where <i>n</i> is a digit) back-references, or subexpression matching. The <i>old</i> string shall also be permitted to contain <code>&lt;newline&gt;</code> characters.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 100967 |                         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 100968 |                         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 100969 |                         | Any non-null character can be used as a delimiter ( <code>'/'</code> shown here). Multiple <code>-s</code> expressions can be specified; the expressions shall be applied in the order specified, terminating with the first successful substitution. The optional trailing <code>'g'</code> is as defined in the <i>ed</i> utility. The optional trailing <code>'p'</code> shall cause successful substitutions to be written to standard error. File or archive member names that substitute to the empty string shall be ignored when reading and writing archives.                                                                                                                                                                                                                                                                                                                                 |
| 100970 |                         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 100971 |                         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 100972 |                         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 100973 |                         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 100974 |                         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 100975 | <code>-t</code>         | When reading files from the file system, and if the user has the permissions required by <i>utime()</i> to do so, set the access time of each file read to the access time that it had before being read by <i>pax</i> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| 100976 |                         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 100977 |                         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 100978 | <code>-u</code>         | Ignore files that are older (having a less recent file modification time) than a pre-existing file or archive member with the same name. In <b>read</b> mode, an archive member with the same name as a file in the file system shall be extracted if the archive member is newer than the file. In <b>write</b> mode, an archive file member with the same name as a file in the file system shall be superseded if the file is newer than the archive member. If <code>-a</code> is also specified, this is accomplished by appending to the archive; otherwise, it is unspecified whether this is accomplished by actual replacement in the archive or by appending to the archive. In <b>copy</b> mode, the file in the destination hierarchy shall be replaced by the file in the source hierarchy or by a link to the file in the source hierarchy if the file in the source hierarchy is newer. |
| 100979 |                         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 100980 |                         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 100981 |                         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 100982 |                         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 100983 |                         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 100984 |                         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 100985 |                         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 100986 |                         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 100987 |                         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 100988 |                         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 100989 | <code>-v</code>         | In <b>list</b> mode, produce a verbose table of contents (see the STDOUT section). Otherwise, write archive member pathnames to standard error (see the STDERR section).                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| 100990 |                         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 100991 |                         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 100992 | <code>-x format</code>  | Specify the output archive format. The <i>pax</i> utility shall support the following formats:                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 100993 |                         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 100994 | <b>cpio</b>             | The <b>cpio</b> interchange format; see the EXTENDED DESCRIPTION section. The default <i>blocksize</i> for this format for character special archive files shall be 5120. Implementations shall support all <i>blocksize</i> values less than or equal to 32768 that are multiples of 512.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| 100995 |                         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 100996 |                         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 100997 |                         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |

|        |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|--------|--------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 100998 | <b>pax</b>   | The <b>pax</b> interchange format; see the EXTENDED DESCRIPTION section. The default <i>blocksize</i> for this format for character special archive files shall be 5120. Implementations shall support all <i>blocksize</i> values less than or equal to 32768 that are multiples of 512.                                                                                                                                                                                                                                                      |
| 100999 |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 101000 |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 101001 |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 101002 | <b>ustar</b> | The <b>tar</b> interchange format; see the EXTENDED DESCRIPTION section. The default <i>blocksize</i> for this format for character special archive files shall be 10240. Implementations shall support all <i>blocksize</i> values less than or equal to 32768 that are multiples of 512.                                                                                                                                                                                                                                                     |
| 101003 |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 101004 |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 101005 |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 101006 |              | Implementation-defined formats shall specify a default block size as well as any other block sizes supported for character special archive files.                                                                                                                                                                                                                                                                                                                                                                                              |
| 101007 |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 101008 |              | Any attempt to append to an archive file in a format different from the existing archive format shall cause <i>pax</i> to exit immediately with a non-zero exit status.                                                                                                                                                                                                                                                                                                                                                                        |
| 101009 |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 101010 | <b>-X</b>    | When traversing the file hierarchy specified by a pathname, <i>pax</i> shall not descend into directories that have a different device ID ( <i>st_dev</i> ; see the System Interfaces volume of POSIX.1-2008, <i>stat()</i> ).                                                                                                                                                                                                                                                                                                                 |
| 101011 |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 101012 |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 101013 |              | Specifying more than one of the mutually-exclusive options <b>-H</b> and <b>-L</b> shall not be considered an error and the last option specified shall determine the behavior of the utility.                                                                                                                                                                                                                                                                                                                                                 |
| 101014 |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 101015 |              | The options that operate on the names of files or archive members ( <b>-c</b> , <b>-i</b> , <b>-n</b> , <b>-s</b> , <b>-u</b> , and <b>-v</b> ) shall interact as follows. In <b>read</b> mode, the archive members shall be selected based on the user-specified <i>pattern</i> operands as modified by the <b>-c</b> , <b>-n</b> , and <b>-u</b> options. Then, any <b>-s</b> and <b>-i</b> options shall modify, in that order, the names of the selected files. The <b>-v</b> option shall write names resulting from these modifications. |
| 101016 |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 101017 |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 101018 |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 101019 |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 101020 |              | In <b>write</b> mode, the files shall be selected based on the user-specified pathnames as modified by the <b>-n</b> and <b>-u</b> options. Then, any <b>-s</b> and <b>-i</b> options shall modify, in that order, the names of these selected files. The <b>-v</b> option shall write names resulting from these modifications.                                                                                                                                                                                                               |
| 101021 |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 101022 |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 101023 |              | If both the <b>-u</b> and <b>-n</b> options are specified, <i>pax</i> shall not consider a file selected unless it is newer than the file to which it is compared.                                                                                                                                                                                                                                                                                                                                                                             |
| 101024 |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 101025 |              | <b>List Mode Format Specifications</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 101026 |              | In <b>list</b> mode with the <b>-o listopt=format</b> option, the <i>format</i> argument shall be applied for each selected file. The <i>pax</i> utility shall append a <newline> to the <b>listopt</b> output for each selected file.                                                                                                                                                                                                                                                                                                         |
| 101027 |              | The <i>format</i> argument shall be used as the <i>format</i> string described in XBD Chapter 5 (on page 121), with the exceptions 1. through 6. defined in the EXTENDED DESCRIPTION section of <i>printf</i> , plus the following exceptions:                                                                                                                                                                                                                                                                                                 |
| 101028 |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 101029 |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 101030 |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 101031 |              | 7. The sequence ( <i>keyword</i> ) can occur before a format conversion specifier. The conversion argument is defined by the value of <i>keyword</i> . The implementation shall support the following keywords:                                                                                                                                                                                                                                                                                                                                |
| 101032 |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 101033 |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 101034 |              | — Any of the Field Name entries in Table 4-14 (on page 3050) and Table 4-16 (on page 3053). The implementation may support the <i>cpio</i> keywords without the leading <b>c_</b> in addition to the form required by Table 4-16 (on page 3053).                                                                                                                                                                                                                                                                                               |
| 101035 |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 101036 |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 101037 |              | — Any keyword defined for the extended header in pax Extended Header (on page 3045).                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| 101038 |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 101039 |              | — Any keyword provided as an implementation-defined extension within the extended header defined in pax Extended Header (on page 3045).                                                                                                                                                                                                                                                                                                                                                                                                        |
| 101040 |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 101041 |              | For example, the sequence "%(charset)s" is the string value of the name of the character                                                                                                                                                                                                                                                                                                                                                                                                                                                       |

- 101042 set in the extended header.
- 101043 The result of the keyword conversion argument shall be the value from the applicable  
101044 header field or extended header, without any trailing NULs.
- 101045 All keyword values used as conversion arguments shall be translated from the UTF-8  
101046 encoding (or alternative encoding specified by any **hdrcharset** extended header record) to  
101047 the character set appropriate for the local file system, user database, and so on, as  
101048 applicable.
- 101049 8. An additional conversion specifier character, **T**, shall be used to specify time formats. The **T**  
101050 conversion specifier character can be preceded by the sequence (*keyword=subformat*), where  
101051 *subformat* is a date format as defined by *date* operands. The default *keyword* shall be **mtime**  
101052 and the default subformat shall be:
- 101053 %b %e %H:%M %Y
- 101054 9. An additional conversion specifier character, **M**, shall be used to specify the file mode string  
101055 as defined in *ls* Standard Output. If (*keyword*) is omitted, the **mode** keyword shall be used.  
101056 For example, % .1M writes the single character corresponding to the <entry type> field of the  
101057 *ls -l* command.
- 101058 10. An additional conversion specifier character, **D**, shall be used to specify the device for block  
101059 or special files, if applicable, in an implementation-defined format. If not applicable, and  
101060 (*keyword*) is specified, then this conversion shall be equivalent to %(*keyword*)u. If not  
101061 applicable, and (*keyword*) is omitted, then this conversion shall be equivalent to <space>.
- 101062 11. An additional conversion specifier character, **F**, shall be used to specify a pathname. The **F**  
101063 conversion character can be preceded by a sequence of <comma>-separated keywords:
- 101064 (*keyword*[ , *keyword*] . . . )
- 101065 The values for all the keywords that are non-null shall be concatenated together, each  
101066 separated by a ' / '. The default shall be (**path**) if the keyword **path** is defined; otherwise,  
101067 the default shall be (**prefix,name**).
- 101068 12. An additional conversion specifier character, **L**, shall be used to specify a symbolic link  
101069 expansion. If the current file is a symbolic link, then %L shall expand to:
- 101070 "%s -> %s", <value of keyword>, <contents of link>
- 101071 Otherwise, the %L conversion specification shall be the equivalent of %F.

## 101072 OPERANDS

101073 The following operands shall be supported:

- 101074 *directory* The destination directory pathname for **copy** mode.
- 101075 *file* A pathname of a file to be copied or archived.
- 101076 *pattern* A pattern matching one or more pathnames of archive members. A pattern must  
101077 be given in the name-generating notation of the pattern matching notation in  
101078 [Section 2.13](#) (on page 2354), including the filename expansion rules in [Section](#)  
101079 [2.13.3](#) (on page 2355). The default, if no *pattern* is specified, is to select all members  
101080 in the archive.

## 101081 STDIN

- 101082 In **write** mode, the standard input shall be used only if no *file* operands are specified. It shall be a |  
101083 file containing a list of pathnames, each terminated by a <newline> character.
- 101084 In **list** and **read** modes, if **-f** is not specified, the standard input shall be an archive file.



101085           Otherwise, the standard input shall not be used.

## 101086 INPUT FILES

101087           The input file named by the *archive* option-argument, or standard input when the archive is read  
101088           from there, shall be a file formatted according to one of the specifications in the EXTENDED  
101089           DESCRIPTION section or some other implementation-defined format.

101090           The file */dev/tty* shall be used to write prompts and read responses.

## 101091 ENVIRONMENT VARIABLES

101092           The following environment variables shall affect the execution of *pax*:

101093           **LANG**           Provide a default value for the internationalization variables that are unset or null.  
101094                               (See XBD [Section 8.2](#) (on page 174) the precedence of internationalization variables  
101095                               used to determine the values of locale categories.)

101096           **LC\_ALL**       If set to a non-empty string value, override the values of all the other  
101097                               internationalization variables.

101098           **LC\_COLLATE**

101099                               Determine the locale for the behavior of ranges, equivalence classes, and multi-  
101100                               character collating elements used in the pattern matching expressions for the  
101101                               *pattern* operand, the basic regular expression for the *-s* option, and the extended  
101102                               regular expression defined for the **yesexpr** locale keyword in the *LC\_MESSAGES*  
101103                               category.

101104           **LC\_CTYPE**   Determine the locale for the interpretation of sequences of bytes of text data as  
101105                               characters (for example, single-byte as opposed to multi-byte characters in  
101106                               arguments and input files), the behavior of character classes used in the extended  
101107                               regular expression defined for the **yesexpr** locale keyword in the *LC\_MESSAGES*  
101108                               category, and pattern matching.

101109           **LC\_MESSAGES**

101110                               Determine the locale used to process affirmative responses, and the locale used to  
101111                               affect the format and contents of diagnostic messages and prompts written to  
101112                               standard error.

101113           **LC\_TIME**     Determine the format and contents of date and time strings when the *-v* option is  
101114                               specified.

101115 XSI       **NLSPATH**   Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

101116           **TMPDIR**     Determine the pathname that provides part of the default global extended header  
101117                               record file, as described for the **-o globexthdr=** keyword in the *OPTIONS* section.

101118           **TZ**           Determine the timezone used to calculate date and time strings when the *-v* option  
101119                               is specified. If *TZ* is unset or null, an unspecified default timezone shall be used.

## 101120 ASYNCHRONOUS EVENTS

101121           Default.

## 101122 STDOUT

101123           In **write** mode, if *-f* is not specified, the standard output shall be the archive formatted  
101124           according to one of the specifications in the EXTENDED DESCRIPTION section, or some other  
101125           implementation-defined format (see *-x format*).

101126           In **list** mode, when the **-olistopt=format** has been specified, the selected archive members shall  
101127           be written to standard output using the format described under [List Mode Format Specifications](#)  
101128           (on page 3040). In **list** mode without the **-olistopt=format** option, the table of contents of the



101129 selected archive members shall be written to standard output using the following format:

101130 "%s\n", <pathname>

101131 If the **-v** option is specified in **list** mode, the table of contents of the selected archive members  
101132 shall be written to standard output using the following formats.

101133 For pathnames representing hard links to previous members of the archive:

101134 "%sΔ==Δ%s\n", <ls -l listing>, <linkname>

101135 For all other pathnames:

101136 "%s\n", <ls -l listing>

101137 where <ls -l listing> shall be the format specified by the *ls* utility with the **-l** option. When  
101138 writing pathnames in this format, it is unspecified what is written for fields for which the  
101139 underlying archive format does not have the correct information, although the correct number of  
101140 <blank>-separated fields shall be written.

101141 In **list** mode, standard output shall not be buffered more than a pathname (plus any associated  
101142 information and a <newline> terminator) at a time.

#### 101143 STDERR

101144 If **-v** is specified in **read**, **write**, or **copy** modes, *pax* shall write the pathnames it processes to the  
101145 standard error output using the following format:

101146 "%s\n", <pathname>

101147 These pathnames shall be written as soon as processing is begun on the file or archive member,  
101148 and shall be flushed to standard error. The trailing <newline>, which shall not be buffered, is  
101149 written when the file has been read or written.

101150 If the **-s** option is specified, and the replacement string has a trailing 'p', substitutions shall be  
101151 written to standard error in the following format:

101152 "%sΔ>>Δ%s\n", <original pathname>, <new pathname>

101153 In all operating modes of *pax*, optional messages of unspecified format concerning the input  
101154 archive format and volume number, the number of files, blocks, volumes, and media parts as  
101155 well as other diagnostic messages may be written to standard error.

101156 In all formats, for both standard output and standard error, it is unspecified how non-printable  
101157 characters in pathnames or link names are written.

101158 When using the **-xpax** archive format, if a filename, link name, group name, owner name, or any  
101159 other field in an extended header record cannot be translated between the codeset in use for that  
101160 extended header record and the character set of the current locale, *pax* shall write a diagnostic  
101161 message to standard error, shall process the file as described for the **-o invalid=** option, and then  
101162 shall continue processing with the next file.

#### 101163 OUTPUT FILES

101164 In **read** mode, the extracted output files shall be of the archived file type. In **copy** mode, the  
101165 copied output files shall be the type of the file being copied. In either mode, existing files in the  
101166 destination hierarchy shall be overwritten only when all permission (**-p**), modification time (**-u**),  
101167 and invalid-value (**-o invalid=**) tests allow it.

101168 In **write** mode, the output file named by the **-f** option-argument shall be a file formatted  
101169 according to one of the specifications in the EXTENDED DESCRIPTION section, or some other  
101170 implementation-defined format.

## 101171 EXTENDED DESCRIPTION

101172 **pax Interchange Format**

101173 A *pax* archive tape or file produced in the **-xpax** format shall contain a series of blocks. The  
 101174 physical layout of the archive shall be identical to the **ustar** format described in **ustar**  
 101175 **Interchange Format** (on page 3049). Each file archived shall be represented by the following  
 101176 sequence:

- 101177 • An optional header block with extended header records. This header block is of the form  
 101178 described in **pax Header Block** (on page 3045), with a *typeflag* value of **x** or **g**. The  
 101179 extended header records, described in **pax Extended Header** (on page 3045), shall be  
 101180 included as the data for this header block.
- 101181 • A header block that describes the file. Any fields in the preceding optional extended  
 101182 header shall override the associated fields in this header block for this file.
- 101183 • Zero or more blocks that contain the contents of the file.

101184 At the end of the archive file there shall be two 512-byte blocks filled with binary zeros,  
 101185 interpreted as an end-of-archive indicator.

101186 A schematic of an example archive with global extended header records and two actual files is  
 101187 shown in **Figure 4-1**. In the example, the second file in the archive has no extended header  
 101188 preceding it, presumably because it has no need for extended attributes.

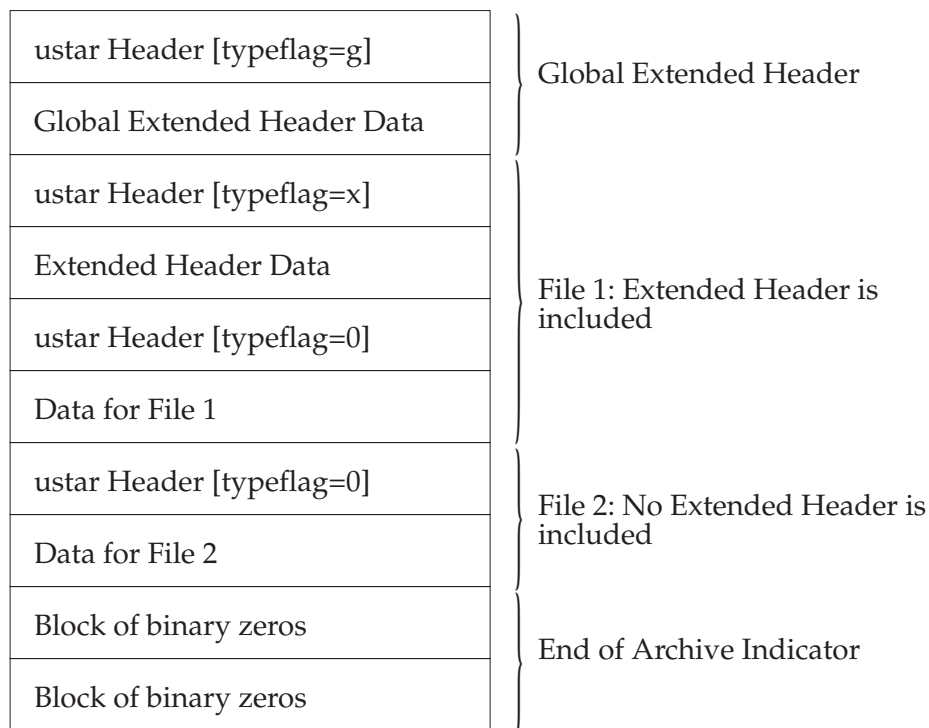


**Figure 4-1** pax Format Archive Example

101189

## pax Header Block

The **pax** header block shall be identical to the **ustar** header block described in **ustar Interchange Format** (on page 3049), except that two additional *typeflag* values are defined:

- x Represents extended header records for the following file in the archive (which shall have its own **ustar** header block). The format of these extended header records shall be as described in **pax Extended Header**.
- g Represents global extended header records for the following files in the archive. The format of these extended header records shall be as described in **pax Extended Header**. Each value shall affect all subsequent files that do not override that value in their own extended header record and until another global extended header record is reached that provides another value for the same field. The *typeflag* **g** global headers should not be used with interchange media that could suffer partial data loss in transporting the archive.

For both of these types, the *size* field shall be the size of the extended header records in octets. The other fields in the header block are not meaningful to this version of the *pax* utility. However, if this archive is read by a *pax* utility conforming to the ISO POSIX-2:1993 standard, the header block fields are used to create a regular file that contains the extended header records as data. Therefore, header block field values should be selected to provide reasonable file access to this regular file.

A further difference from the **ustar** header block is that data blocks for files of *typeflag* 1 (the digit one) (hard link) may be included, which means that the *size* field may be greater than zero. Archives created by *pax -o linkdata* shall include these data blocks with the hard links.

## pax Extended Header

A **pax** extended header contains values that are inappropriate for the **ustar** header block because of limitations in that format: fields requiring a character encoding other than that described in the ISO/IEC 646:1991 standard, fields representing file attributes not described in the **ustar** header, and fields whose format or length do not fit the requirements of the **ustar** header. The values in an extended header add attributes to the following file (or files; see the description of the *typeflag* **g** header block) or override values in the following header block(s), as indicated in the following list of keywords.

An extended header shall consist of one or more records, each constructed as follows:

```
"%d %s=%s\n", <length>, <keyword>, <value>
```

The extended header records shall be encoded according to the ISO/IEC 10646-1:2000 standard UTF-8 encoding. The *<length>* field, *<blank>*, *<equals-sign>*, and *<newline>* shown shall be limited to the portable character set, as encoded in UTF-8. The *<keyword>* fields can be any UTF-8 characters. The *<length>* field shall be the decimal length of the extended header record in octets, including the trailing *<newline>*. If there is a **hdrcharset** extended header in effect for a file, the *value* field for any **gname**, **linkpath**, **path**, and **uname** extended header records shall be encoded using the character set specified by the **hdrcharset** extended header record; otherwise, the *value* field shall be encoded using UTF-8. The *value* field for all other keywords specified by POSIX.1-2008 shall be encoded using UTF-8.

The *<keyword>* field shall be one of the entries from the following list or a keyword provided as an implementation extension. Keywords consisting entirely of lowercase letters, digits, and periods are reserved for future standardization. A keyword shall not include an *<equals-sign>*. (In the following list, the notations "file(s)" or "block(s)" is used to acknowledge that a keyword affects the following single file after a *typeflag* **x** extended header, but possibly multiple files after *typeflag* **g**. Any requirements in the list for *pax* to include a record when in **write** or **copy** mode

shall apply only when such a record has not already been provided through the use of the **-o** option. When used in **copy** mode, *pax* shall behave as if an archive had been created with applicable extended header records and then extracted.)

**atime** The file access time for the following file(s), equivalent to the value of the *st\_atime* member of the **stat** structure for a file, as described by the *stat()* function. The access time shall be restored if the process has appropriate privileges required to do so. The format of the *<value>* shall be as described in [pax Extended Header File Times](#) (on page 3049).

**charset** The name of the character set used to encode the data in the following file(s). The entries in the following table are defined to refer to known standards; additional names may be agreed on between the originator and recipient.

| <i>&lt;value&gt;</i>    | Formal Standard               |
|-------------------------|-------------------------------|
| ISO-IRΔ646Δ1990         | ISO/IEC 646:1990              |
| ISO-IRΔ8859Δ1Δ1998      | ISO/IEC 8859-1:1998           |
| ISO-IRΔ8859Δ2Δ1999      | ISO/IEC 8859-2:1999           |
| ISO-IRΔ8859Δ3Δ1999      | ISO/IEC 8859-3:1999           |
| ISO-IRΔ8859Δ4Δ1998      | ISO/IEC 8859-4:1998           |
| ISO-IRΔ8859Δ5Δ1999      | ISO/IEC 8859-5:1999           |
| ISO-IRΔ8859Δ6Δ1999      | ISO/IEC 8859-6:1999           |
| ISO-IRΔ8859Δ7Δ1987      | ISO/IEC 8859-7:1987           |
| ISO-IRΔ8859Δ8Δ1999      | ISO/IEC 8859-8:1999           |
| ISO-IRΔ8859Δ9Δ1999      | ISO/IEC 8859-9:1999           |
| ISO-IRΔ8859Δ10Δ1998     | ISO/IEC 8859-10:1998          |
| ISO-IRΔ8859Δ13Δ1998     | ISO/IEC 8859-13:1998          |
| ISO-IRΔ8859Δ14Δ1998     | ISO/IEC 8859-14:1998          |
| ISO-IRΔ8859Δ15Δ1999     | ISO/IEC 8859-15:1999          |
| ISO-IRΔ10646Δ2000       | ISO/IEC 10646:2000            |
| ISO-IRΔ10646Δ2000ΔUTF-8 | ISO/IEC 10646, UTF-8 encoding |
| BINARY                  | None.                         |

The encoding is included in an extended header for information only; when *pax* is used as described in POSIX.1-2008, it shall not translate the file data into any other encoding. The **BINARY** entry indicates unencoded binary data.

When used in **write** or **copy** mode, it is implementation-defined whether *pax* includes a **charset** extended header record for a file.

**comment** A series of characters used as a comment. All characters in the *<value>* field shall be ignored by *pax*.

**gid** The group ID of the group that owns the file, expressed as a decimal number using digits from the ISO/IEC 646:1991 standard. This record shall override the *gid* field in the following header block(s). When used in **write** or **copy** mode, *pax* shall include a *gid* extended header record for each file whose group ID is greater than 2 097 151 (octal 7 777 777).

**gname** The group of the file(s), formatted as a group name in the group database. This record shall override the *gid* and *gname* fields in the following header block(s), and any *gid* extended header record. When used in **read**, **copy**, or **list** mode, *pax* shall translate the name from the encoding in the header record to the character set appropriate for the group database on the receiving system. If any of the characters cannot be translated, and if neither the **-oinvalid=UTF-8** option nor the

101283 **—oinvalid=binary** option is specified, the results are implementation-defined.  
 101284 When used in **write** or **copy** mode, *pax* shall include a **gname** extended header  
 101285 record for each file whose group name cannot be represented entirely with the  
 101286 letters and digits of the portable character set.

101287 **hdrcharset** The name of the character set used to encode the value field of the **gname**,  
 101288 **linkpath**, **path**, and **uname** *pax* extended header records. The entries in the  
 101289 following table are defined to refer to known standards; additional names may be  
 101290 agreed between the originator and the recipient.

| <value>                 | Formal Standard               |
|-------------------------|-------------------------------|
| ISO-IRΔ10646Δ2000ΔUTF-8 | ISO/IEC 10646, UTF-8 encoding |
| BINARY                  | None.                         |

101294 If no **hdrcharset** extended header record is specified, the default character set used  
 101295 to encode all values in extended header records shall be the ISO/IEC 10646-1: 2000  
 101296 standard UTF-8 encoding.

101297 The **BINARY** entry indicates that all values recorded in extended headers for  
 101298 affected files are unencoded binary data from the underlying system.

101299 **linkpath** The pathname of a link being created to another file, of any type, previously  
 101300 archived. This record shall override the *linkname* field in the following **ustar** header  
 101301 block(s). The following **ustar** header block shall determine the type of link created.  
 101302 If *typeflag* of the following header block is 1, it shall be a hard link. If *typeflag* is 2, it  
 101303 shall be a symbolic link and the **linkpath** value shall be the contents of the  
 101304 symbolic link. The *pax* utility shall translate the name of the link (contents of the  
 101305 symbolic link) from the encoding in the header to the character set appropriate for  
 101306 the local file system. When used in **write** or **copy** mode, *pax* shall include a  
 101307 **linkpath** extended header record for each link whose pathname cannot be  
 101308 represented entirely with the members of the portable character set other than  
 101309 NUL.

101310 **mtime** The file modification time of the following file(s), equivalent to the value of the  
 101311 *st\_mtime* member of the **stat** structure for a file, as described in the *stat()* function.  
 101312 This record shall override the *mtime* field in the following header block(s). The  
 101313 modification time shall be restored if the process has appropriate privileges  
 101314 required to do so. The format of the <value> shall be as described in [pax Extended](#)  
 101315 [Header File Times](#) (on page 3049).

101316 **path** The pathname of the following file(s). This record shall override the *name* and  
 101317 *prefix* fields in the following header block(s). The *pax* utility shall translate the  
 101318 pathname of the file from the encoding in the header to the character set  
 101319 appropriate for the local file system.

101320 When used in **write** or **copy** mode, *pax* shall include a *path* extended header record  
 101321 for each file whose pathname cannot be represented entirely with the members of  
 101322 the portable character set other than NUL.

101323 **realtime.any** The keywords prefixed by “realtime.” are reserved for future standardization.

101324 **security.any** The keywords prefixed by “security.” are reserved for future standardization.

101325 **size** The size of the file in octets, expressed as a decimal number using digits from the  
 101326 ISO/IEC 646:1991 standard. This record shall override the *size* field in the  
 101327 following header block(s). When used in **write** or **copy** mode, *pax* shall include a  
 101328 *size* extended header record for each file with a size value greater than 8 589 934 591

101329

101330

101331

101332

101333

101334

101335

101336

101337

101338

101339

101340

101341

101342

101343

101344

101345

101346

101347

101348

101349

101350

101351

101352

101353

101354

101355

101356

101357

101358

101359

101360

101361

101362

101363

101364

101365

101366

101367

101368

**uid**

The user ID of the file owner, expressed as a decimal number using digits from the ISO/IEC 646:1991 standard. This record shall override the *uid* field in the following header block(s). When used in **write** or **copy** mode, *pax* shall include a *uid* extended header record for each file whose owner ID is greater than 2097151 (octal 7777777).

**uname**

The owner of the following file(s), formatted as a user name in the user database. This record shall override the *uid* and *uname* fields in the following header block(s), and any *uid* extended header record. When used in **read**, **copy**, or **list** mode, *pax* shall translate the name from the encoding in the header record to the character set appropriate for the user database on the receiving system. If any of the characters cannot be translated, and if neither the **-oinvalid=UTF-8** option nor the **-oinvalid=binary** option is specified, the results are implementation-defined. When used in **write** or **copy** mode, *pax* shall include a **uname** extended header record for each file whose user name cannot be represented entirely with the letters and digits of the portable character set.

If the *<value>* field is zero length, it shall delete any header block field, previously entered extended header value, or global extended header value of the same name.

If a keyword in an extended header record (or in a **-o** option-argument) overrides or deletes a corresponding field in the **ustar** header block, *pax* shall ignore the contents of that header block field.

Unlike the **ustar** header block fields, NULs shall not delimit *<value>*s; all characters within the *<value>* field shall be considered data for the field. None of the length limitations of the **ustar** header block fields in [Table 4-14](#) (on page 3050) shall apply to the extended header records.

**pax Extended Header Keyword Precedence**

This section describes the precedence in which the various header records and fields and command line options are selected to apply to a file in the archive. When *pax* is used in **read** or **list** modes, it shall determine a file attribute in the following sequence:

1. If **-odelete=keyword-prefix** is used, the affected attributes shall be determined from step 7., if applicable, or ignored otherwise.
2. If **-okeyword:=** is used, the affected attributes shall be ignored.
3. If **-okeyword:=value** is used, the affected attribute shall be assigned the value.
4. If there is a *typeflag* **x** extended header record, the affected attribute shall be assigned the *<value>*. When extended header records conflict, the last one given in the header shall take precedence.
5. If **-okeyword=value** is used, the affected attribute shall be assigned the value.
6. If there is a *typeflag* **g** global extended header record, the affected attribute shall be assigned the *<value>*. When global extended header records conflict, the last one given in the global header shall take precedence.
7. Otherwise, the attribute shall be determined from the **ustar** header block.

**pax Extended Header File Times**

The *pax* utility shall write an **mtime** record for each file in **write** or **copy** modes if the file's modification time cannot be represented exactly in the **ustar** header logical record described in **ustar Interchange Format**. This can occur if the time is out of **ustar** range, or if the file system of the underlying implementation supports non-integer time granularities and the time is not an integer. All of these time records shall be formatted as a decimal representation of the time in seconds since the Epoch. If a <period> ( ' . ' ) decimal point character is present, the digits to the right of the point shall represent the units of a subsecond timing granularity, where the first digit is tenths of a second and each subsequent digit is a tenth of the previous digit. In **read** or **copy** mode, the *pax* utility shall truncate the time of a file to the greatest value that is not greater than the input header file time. In **write** or **copy** mode, the *pax* utility shall output a time exactly if it can be represented exactly as a decimal number, and otherwise shall generate only enough digits so that the same time shall be recovered if the file is extracted on a system whose underlying implementation supports the same time granularity.

**ustar Interchange Format**

A **ustar** archive tape or file shall contain a series of logical records. Each logical record shall be a fixed-size logical record of 512 octets (see below). Although this format may be thought of as being stored on 9-track industry-standard 12.7 mm (0.5 in) magnetic tape, other types of transportable media are not excluded. Each file archived shall be represented by a header logical record that describes the file, followed by zero or more logical records that give the contents of the file. At the end of the archive file there shall be two 512-octet logical records filled with binary zeros, interpreted as an end-of-archive indicator.

The logical records may be grouped for physical I/O operations, as described under the **-bblocksize** and **-x ustar** options. Each group of logical records may be written with a single operation equivalent to the *write()* function. On magnetic tape, the result of this write shall be a single tape physical block. The last physical block shall always be the full size, so logical records after the two zero logical records may contain undefined data.

The header logical record shall be structured as shown in the following table. All lengths and offsets are in decimal.



Table 4-14 ustar Header Block

| Field Name      | Octet Offset | Length (in Octets) |
|-----------------|--------------|--------------------|
| <i>name</i>     | 0            | 100                |
| <i>mode</i>     | 100          | 8                  |
| <i>uid</i>      | 108          | 8                  |
| <i>gid</i>      | 116          | 8                  |
| <i>size</i>     | 124          | 12                 |
| <i>mtime</i>    | 136          | 12                 |
| <i>chksum</i>   | 148          | 8                  |
| <i>typeflag</i> | 156          | 1                  |
| <i>linkname</i> | 157          | 100                |
| <i>magic</i>    | 257          | 6                  |
| <i>version</i>  | 263          | 2                  |
| <i>uname</i>    | 265          | 32                 |
| <i>gname</i>    | 297          | 32                 |
| <i>devmajor</i> | 329          | 8                  |
| <i>devminor</i> | 337          | 8                  |
| <i>prefix</i>   | 345          | 155                |

All characters in the header logical record shall be represented in the coded character set of the ISO/IEC 646: 1991 standard. For maximum portability between implementations, names should be selected from characters represented by the portable filename character set as octets with the most significant bit zero. If an implementation supports the use of characters outside of <slash> and the portable filename character set in names for files, users, and groups, one or more implementation-defined encodings of these characters shall be provided for interchange purposes.

However, the *pax* utility shall never create filenames on the local system that cannot be accessed via the procedures described in POSIX.1-2008. If a filename is found on the medium that would create an invalid filename, it is implementation-defined whether the data from the file is stored on the file hierarchy and under what name it is stored. The *pax* utility may choose to ignore these files as long as it produces an error indicating that the file is being ignored.

Each field within the header logical record is contiguous; that is, there is no padding used. Each character on the archive medium shall be stored contiguously.

The fields *magic*, *uname*, and *gname* are character strings each terminated by a NUL character. The fields *name*, *linkname*, and *prefix* are NUL-terminated character strings except when all characters in the array contain non-NUL characters including the last character. The *version* field is two octets containing the characters "00" (zero-zero). The *typeflag* contains a single character. All other fields are leading zero-filled octal numbers using digits from the ISO/IEC 646: 1991 standard IRV. Each numeric field is terminated by one or more <space> or NUL characters.

The *name* and the *prefix* fields shall produce the pathname of the file. A new pathname shall be formed, if *prefix* is not an empty string (its first character is not NUL), by concatenating *prefix* (up to the first NUL character), a <slash> character, and *name*; otherwise, *name* is used alone. In either case, *name* is terminated at the first NUL character. If *prefix* begins with a NUL character, it shall be ignored. In this manner, pathnames of at most 256 characters can be supported. If a pathname does not fit in the space provided, *pax* shall notify the user of the error, and shall not store any part of the file—header or data—on the medium.

The *linkname* field, described below, shall not use the *prefix* to produce a pathname. As such, a *linkname* is limited to 100 characters. If the name does not fit in the space provided, *pax* shall



notify the user of the error, and shall not attempt to store the link on the medium.

The *mode* field provides 12 bits encoded in the ISO/IEC 646:1991 standard octal digit representation. The encoded bits shall represent the following values:

**Table 4-15** ustar *mode* Field

| Bit Value | POSIX.1-2008 Bit | Description                                     |
|-----------|------------------|-------------------------------------------------|
| 04 000    | S_ISUID          | Set UID on execution.                           |
| 02 000    | S_ISGID          | Set GID on execution.                           |
| 01 000    | <reserved>       | Reserved for future standardization.            |
| 00 400    | S_IRUSR          | Read permission for file owner class.           |
| 00 200    | S_IWUSR          | Write permission for file owner class.          |
| 00 100    | S_IXUSR          | Execute/search permission for file owner class. |
| 00 040    | S_IRGRP          | Read permission for file group class.           |
| 00 020    | S_IWGRP          | Write permission for file group class.          |
| 00 010    | S_IXGRP          | Execute/search permission for file group class. |
| 00 004    | S_IROTH          | Read permission for file other class.           |
| 00 002    | S_IWOTH          | Write permission for file other class.          |
| 00 001    | S_IXOTH          | Execute/search permission for file other class. |

When appropriate privileges are required to set one of these mode bits, and the user restoring the files from the archive does not have appropriate privileges, the mode bits for which the user does not have appropriate privileges shall be ignored. Some of the mode bits in the archive format are not mentioned elsewhere in this volume of POSIX.1-2008. If the implementation does not support those bits, they may be ignored.

The *uid* and *gid* fields are the user and group ID of the owner and group of the file, respectively.

The *size* field is the size of the file in octets. If the *typeflag* field is set to specify a file to be of type 1 (a link) or 2 (a symbolic link), the *size* field shall be specified as zero. If the *typeflag* field is set to specify a file of type 5 (directory), the *size* field shall be interpreted as described under the definition of that record type. No data logical records are stored for types 1, 2, or 5. If the *typeflag* field is set to 3 (character special file), 4 (block special file), or 6 (FIFO), the meaning of the *size* field is unspecified by this volume of POSIX.1-2008, and no data logical records shall be stored on the medium. Additionally, for type 6, the *size* field shall be ignored when reading. If the *typeflag* field is set to any other value, the number of logical records written following the header shall be  $(size+511)/512$ , ignoring any fraction in the result of the division.

The *mtime* field shall be the modification time of the file at the time it was archived. It is the ISO/IEC 646:1991 standard representation of the octal value of the modification time obtained from the *stat()* function.

The *chksum* field shall be the ISO/IEC 646:1991 standard IRV representation of the octal value of the simple sum of all octets in the header logical record. Each octet in the header shall be treated as an unsigned value. These values shall be added to an unsigned integer, initialized to zero, the precision of which is not less than 17 bits. When calculating the checksum, the *chksum* field is treated as if it were all <space> characters.

The *typeflag* field specifies the type of file archived. If a particular implementation does not recognize the type, or the user does not have appropriate privileges to create that type, the file shall be extracted as if it were a regular file if the file type is defined to have a meaning for the *size* field that could cause data logical records to be written on the medium (see the previous description for *size*). If conversion to a regular file occurs, the *pax* utility shall produce an error indicating that the conversion took place. All of the *typeflag* fields shall be coded in the

|        |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |                                                                                                                                                                                                                                                                                                                                                                                                                            |
|--------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 101491 | ISO/IEC 646: 1991 standard IRV:                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 101492 | 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       | Represents a regular file. For backwards-compatibility, a <i>typeflag</i> value of binary zero (' \0 ') should be recognized as meaning a regular file when extracting files from the archive. Archives written with this version of the archive file format create regular files with a <i>typeflag</i> value of the ISO/IEC 646: 1991 standard IRV ' 0 '.                                                                |
| 101493 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 101494 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 101495 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 101496 | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       | Represents a file linked to another file, of any type, previously archived. Such files are identified by having the same device and file serial numbers, and pathnames that refer to different directory entries. All such files shall be archived as linked files. The linked-to name is specified in the <i>linkname</i> field with a NUL-character terminator if it is less than 100 octets in length.                  |
| 101497 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 101498 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 101499 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 101500 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 101501 | 2                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       | Represents a symbolic link. The contents of the symbolic link shall be stored in the <i>linkname</i> field.                                                                                                                                                                                                                                                                                                                |
| 101502 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 101503 | 3, 4                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    | Represent character special files and block special files respectively. In this case the <i>devmajor</i> and <i>devminor</i> fields shall contain information defining the device, the format of which is unspecified by this volume of POSIX.1-2008. Implementations may map the device specifications to their own local specification or may ignore the entry.                                                          |
| 101504 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 101505 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 101506 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 101507 | 5                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       | Specifies a directory or subdirectory. On systems where disk allocation is performed on a directory basis, the <i>size</i> field shall contain the maximum number of octets (which may be rounded to the nearest disk block allocation unit) that the directory may hold. A <i>size</i> field of zero indicates no such limiting. Systems that do not support limiting in this manner should ignore the <i>size</i> field. |
| 101508 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 101509 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 101510 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 101511 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 101512 | 6                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       | Specifies a FIFO special file. Note that the archiving of a FIFO file archives the existence of this file and not its contents.                                                                                                                                                                                                                                                                                            |
| 101513 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 101514 | 7                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       | Reserved to represent a file to which an implementation has associated some high-performance attribute. Implementations without such extensions should treat this file as a regular file (type 0).                                                                                                                                                                                                                         |
| 101515 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 101516 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 101517 | A–Z                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     | The letters ' A ' to ' Z ', inclusive, are reserved for custom implementations. All other values are reserved for future versions of this standard.                                                                                                                                                                                                                                                                        |
| 101518 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 101519 | It is unspecified whether files with pathnames that refer to the same directory entry are archived as linked files or as separate files. If they are archived as linked files, this means that attempting to extract both pathnames from the resulting archive will always cause an error (unless the <i>-u</i> option is used) because the link cannot be created.                                                                                                                                                                                                                                                                                                                                                                     |                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 101520 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 101521 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 101522 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 101523 | It is unspecified whether files with the same device and file serial numbers being appended to an archive are treated as linked files to members that were in the archive before the append.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 101524 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 101525 | Attempts to archive a socket using <i>ustar</i> interchange format shall produce a diagnostic message. Handling of other file types is implementation-defined.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 101526 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 101527 | The <i>magic</i> field is the specification that this archive was output in this archive format. If this field contains <i>ustar</i> (the five characters from the ISO/IEC 646: 1991 standard IRV shown followed by NUL), the <i>uname</i> and <i>gname</i> fields shall contain the ISO/IEC 646: 1991 standard IRV representation of the owner and group of the file, respectively (truncated to fit, if necessary). When the file is restored by a privileged, protection-preserving version of the utility, the user and group databases shall be scanned for these names. If found, the user and group IDs contained within these files shall be used rather than the values contained within the <i>uid</i> and <i>gid</i> fields. |                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 101528 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 101529 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 101530 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 101531 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 101532 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 101533 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 101534 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |                                                                                                                                                                                                                                                                                                                                                                                                                            |

**cpio Interchange Format**

The octet-oriented **cpio** archive format shall be a series of entries, each comprising a header that describes the file, the name of the file, and then the contents of the file.

An archive may be recorded as a series of fixed-size blocks of octets. This blocking shall be used only to make physical I/O more efficient. The last group of blocks shall always be at the full size.

For the octet-oriented **cpio** archive format, the individual entry information shall be in the order indicated and described by the following table; see also the **<cpio.h>** header.

**Table 4-16** Octet-Oriented cpio Archive Entry

| Header Field Name    | Length (in Octets) | Interpreted as  |
|----------------------|--------------------|-----------------|
| <i>c_magic</i>       | 6                  | Octal number    |
| <i>c_dev</i>         | 6                  | Octal number    |
| <i>c_ino</i>         | 6                  | Octal number    |
| <i>c_mode</i>        | 6                  | Octal number    |
| <i>c_uid</i>         | 6                  | Octal number    |
| <i>c_gid</i>         | 6                  | Octal number    |
| <i>c_nlink</i>       | 6                  | Octal number    |
| <i>c_rdev</i>        | 6                  | Octal number    |
| <i>c_mtime</i>       | 11                 | Octal number    |
| <i>c_namesize</i>    | 6                  | Octal number    |
| <i>c_filesize</i>    | 11                 | Octal number    |
| Filename Field Name  | Length             | Interpreted as  |
| <i>c_name</i>        | <i>c_namesize</i>  | Pathname string |
| File Data Field Name | Length             | Interpreted as  |
| <i>c_filedata</i>    | <i>c_filesize</i>  | Data            |

**cpio Header**

For each file in the archive, a header as defined previously shall be written. The information in the header fields is written as streams of the ISO/IEC 646:1991 standard characters interpreted as octal numbers. The octal numbers shall be extended to the necessary length by appending the ISO/IEC 646:1991 standard IRV zeros at the most-significant-digit end of the number; the result is written to the most-significant digit of the stream of octets first. The fields shall be interpreted as follows:

*c\_magic* Identify the archive as being a transportable archive by containing the identifying value "070707".

*c\_dev, c\_ino* Contains values that uniquely identify the file within the archive (that is, no files contain the same pair of *c\_dev* and *c\_ino* values unless they are links to the same file). The values shall be determined in an unspecified manner.

*c\_mode* Contains the file type and access permissions as defined in the following table.

101573

**Table 4-17** Values for cpio c\_mode Field

101574

101575

101576

101577

101578

101579

101580

101581

101582

101583

101584

101585

101586

101587

101588

101589

101590

101591

101592

101593

101594

101595

| File Permissions Name | Value    | Indicates              |
|-----------------------|----------|------------------------|
| C_IRUSR               | 000 400  | Read by owner          |
| C_IWUSR               | 000 200  | Write by owner         |
| C_IXUSR               | 000 100  | Execute by owner       |
| C_IRGRP               | 000 040  | Read by group          |
| C_IWGRP               | 000 020  | Write by group         |
| C_IXGRP               | 000 010  | Execute by group       |
| C_IROTH               | 000 004  | Read by others         |
| C_IWOTH               | 000 002  | Write by others        |
| C_IXOTH               | 000 001  | Execute by others      |
| C_ISUID               | 004 000  | Set <i>uid</i>         |
| C_ISGID               | 002 000  | Set <i>gid</i>         |
| C_ISVTX               | 001 000  | Reserved               |
| File Type Name        | Value    | Indicates              |
| C_ISDIR               | 040 000  | Directory              |
| C_ISFIFO              | 010 000  | FIFO                   |
| C_ISREG               | 0100 000 | Regular file           |
| C_ISLNK               | 0120 000 | Symbolic link          |
| C_ISBLK               | 060 000  | Block special file     |
| C_ISCHR               | 020 000  | Character special file |
| C_ISSOCK              | 0140 000 | Socket                 |
| C_ISCTG               | 0110 000 | Reserved               |

101596

101597

101598

101599

101600

Directories, FIFOs, symbolic links, and regular files shall be supported on a system conforming to this volume of POSIX.1-2008; additional values defined previously are reserved for compatibility with existing systems. Additional file types may be supported; however, such files should not be written to archives intended to be transported to other systems.

101601

*c\_uid*

Contains the user ID of the owner.

101602

*c\_gid*

Contains the group ID of the group.

101603

*c\_nlink*

Contains a number greater than or equal to the number of links in the archive referencing the file. If the *-a* option is used to append to a *cpio* archive, then the *pax* utility need not account for the files in the existing part of the archive when calculating the *c\_nlink* values for the appended part of the archive, and need not alter the *c\_nlink* values in the existing part of the archive if additional files with the same *c\_dev* and *c\_ino* values are appended to the archive.

101604

101605

101606

101607

101608

101609

*c\_rdev*

Contains implementation-defined information for character or block special files.

101610

*c\_mtime*

Contains the latest time of modification of the file at the time the archive was created.

101611

101612

*c\_namesize*

Contains the length of the pathname, including the terminating NUL character.

101613

*c\_filesiz*

Contains the length in octets of the data section following the header structure.

**cpio Filename**

The *c\_name* field shall contain the pathname of the file. The length of this field in octets is the value of *c\_namesize*.

If a filename is found on the medium that would create an invalid pathname, it is implementation-defined whether the data from the file is stored on the file hierarchy and under what name it is stored.

All characters shall be represented in the ISO/IEC 646:1991 standard IRV. For maximum portability between implementations, names should be selected from characters represented by the portable filename character set as octets with the most significant bit zero. If an implementation supports the use of characters outside the portable filename character set in names for files, users, and groups, one or more implementation-defined encodings of these characters shall be provided for interchange purposes. However, the *pax* utility shall never create filenames on the local system that cannot be accessed via the procedures described previously in this volume of POSIX.1-2008. If a filename is found on the medium that would create an invalid filename, it is implementation-defined whether the data from the file is stored on the local file system and under what name it is stored. The *pax* utility may choose to ignore these files as long as it produces an error indicating that the file is being ignored.

**cpio File Data**

Following *c\_name*, there shall be *c\_filesize* octets of data. Interpretation of such data occurs in a manner dependent on the file. For regular files, the data shall consist of the contents of the file. For symbolic links, the data shall consist of the contents of the symbolic link. If *c\_filesize* is zero, no data shall be contained in *c\_filedata*.

When restoring from an archive:

- If the user does not have appropriate privileges to create a file of the specified type, *pax* shall ignore the entry and write an error message to standard error.
- Only regular files and symbolic links have data to be restored. Presuming a regular file meets any selection criteria that might be imposed on the format-reading utility by the user, such data shall be restored.
- If a user does not have appropriate privileges to set a particular mode flag, the flag shall be ignored. Some of the mode flags in the archive format are not mentioned elsewhere in this volume of POSIX.1-2008. If the implementation does not support those flags, they may be ignored.

**cpio Special Entries**

FIFO special files, directories, and the trailer shall be recorded with *c\_filesize* equal to zero. Symbolic links shall be recorded with *c\_filesize* equal to the length of the contents of the symbolic link. For other special files, *c\_filesize* is unspecified by this volume of POSIX.1-2008. The header for the next file entry in the archive shall be written directly after the last octet of the file entry preceding it. A header denoting the filename **TRAILER!!!** shall indicate the end of the archive; the contents of octets in the last block of the archive following such a header are undefined.

**EXIT STATUS**

The following exit values shall be returned:

- 0 All files were processed successfully.

101656 >0 An error occurred.

## 101657 CONSEQUENCES OF ERRORS

101658 If *pax* cannot create a file or a link when reading an archive or cannot find a file when writing an  
 101659 archive, or cannot preserve the user ID, group ID, or file mode when the **-p** option is specified, a  
 101660 diagnostic message shall be written to standard error and a non-zero exit status shall be  
 101661 returned, but processing shall continue. In the case where *pax* cannot create a link to a file, *pax*  
 101662 shall not, by default, create a second copy of the file.

101663 If the extraction of a file from an archive is prematurely terminated by a signal or error, *pax* may  
 101664 have only partially extracted the file or (if the **-n** option was not specified) may have extracted a  
 101665 file of the same name as that specified by the user, but which is not the file the user wanted.  
 101666 Additionally, the file modes of extracted directories may have additional bits from the S\_IRWXU  
 101667 mask set as well as incorrect modification and access times.

## 101668 APPLICATION USAGE

101669 Caution is advised when using the **-a** option to append to a *cpio* format archive. If any of the  
 101670 files being appended happen to be given the same *c\_dev* and *c\_ino* values as a file in the existing  
 101671 part of the archive, then they may be treated as links to that file on extraction. Thus, it is risky to  
 101672 use **-a** with *cpio* format except when it is done on the same system that the original archive was  
 101673 created on, and with the same *pax* utility, and in the knowledge that there has been little or no  
 101674 file system activity since the original archive was created that could lead to any of the files  
 101675 appended being given the same *c\_dev* and *c\_ino* values as an unrelated file in the existing part of  
 101676 the archive. Also, when (intentionally) appending additional links to a file in the existing part of  
 101677 the archive, the *c\_nlink* values in the modified archive can be smaller than the number of links to  
 101678 the file in the archive, which may mean that the links are not preserved on extraction.

101679 The **-p** (privileges) option was invented to reconcile differences between historical *tar* and *cpio*  
 101680 implementations. In particular, the two utilities use **-m** in diametrically opposed ways. The **-p**  
 101681 option also provides a consistent means of extending the ways in which future file attributes can  
 101682 be addressed, such as for enhanced security systems or high-performance files. Although it may  
 101683 seem complex, there are really two modes that are most commonly used:

101684 **-p e** “Preserve everything”. This would be used by the historical superuser, someone with  
 101685 all appropriate privileges, to preserve all aspects of the files as they are recorded in the  
 101686 archive. The **e** flag is the sum of **o** and **p**, and other implementation-defined attributes.

101687 **-p p** “Preserve” the file mode bits. This would be used by the user with regular privileges  
 101688 who wished to preserve aspects of the file other than the ownership. The file times are  
 101689 preserved by default, but two other flags are offered to disable these and use the time  
 101690 of extraction.

101691 The one pathname per line format of standard input precludes pathnames containing <newline>  
 101692 characters. Although such pathnames violate the portable filename guidelines, they may exist  
 101693 and their presence may inhibit usage of *pax* within shell scripts. This problem is inherited from  
 101694 historical archive programs. The problem can be avoided by listing filename arguments on the  
 101695 command line instead of on standard input.

101696 It is almost certain that appropriate privileges are required for *pax* to accomplish parts of this  
 101697 volume of POSIX.1-2008. Specifically, creating files of type block special or character special,  
 101698 restoring file access times unless the files are owned by the user (the **-t** option), or preserving file  
 101699 owner, group, and mode (the **-p** option) all probably require appropriate privileges.

101700 In **read** mode, implementations are permitted to overwrite files when the archive has multiple  
 101701 members with the same name. This may fail if permissions on the first version of the file do not  
 101702 permit it to be overwritten.

101703 The **cpio** and **ustar** formats can only support files up to 8 589 934 592 bytes ( $8 * 2^{30}$ ) in size.

101704 When archives containing binary header information are listed , the filenames printed may  
101705 cause strange behavior on some terminals.

101706 When all of the following are true: +

- 101707 1. A file of type directory is being placed into an archive. +
- 101708 2. The **ustar** archive format is being used. +
- 101709 3. The pathname of the directory is less than or equal to 155 bytes long (it will fit in the *prefix* +  
101710 field in the **ustar** header block). +
- 101711 4. The last component of the pathname of the directory is longer than 100 bytes long (it will +  
101712 not fit in the *name* field in the **ustar** header block). +

101713 some implementations of the *pax* utility will place the entire directory pathname in the *prefix* +  
101714 field, set the *name* field to an empty string, and place the directory in the archive. Other +  
101715 implementations of the *pax* utility will give an error under these conditions because the *name* +  
101716 field is not large enough to hold the last component of the directory name. This standard allows +  
101717 either behavior. However, when extracting a directory from a **ustar** format archive, this standard +  
101718 requires that all implementations be able to extract a directory even if the *name* field contains an +  
101719 empty string as long as the *prefix* field does not also contain an empty string.

101720 **EXAMPLES**

101721 The following command:

101722 `pax -w -f /dev/rmt/lm .`

101723 copies the contents of the current directory to tape drive 1, medium density (assuming historical  
101724 System V device naming procedures—the historical BSD device name would be **/dev/rmt9**).

101725 The following commands:

101726 `mkdir newdir`  
101727 `pax -rw olddir newdir`

101728 copy the *olddir* directory hierarchy to *newdir*.

101729 `pax -r -s ',^//*usr//*,,' -f a.pax`

101730 reads the archive **a.pax**, with all files rooted in **/usr** in the archive extracted relative to the current  
101731 directory.

101732 Using the option:

101733 `-o listopt="%M %(atime)T %(size)D %(name)s"`

101734 overrides the default output description in Standard Output and instead writes:

101735 `-rw-rw--- Jan 12 15:53 2003 1492 /usr/foo/bar`

101736 Using the options:

101737 `-o listopt='%L\t%(size)D\n%.7' \`  
101738 `-o listopt='(name)s\n%(atime)T\n%T'`

101739 overrides the default output description in Standard Output and instead writes:

101740 `/usr/foo/bar -> /tmp 1492`  
101741 `/usr/fo`  
101742 `Jan 12 15:53 1991`

101743 Jan 31 15:53 2003

#### 101744 RATIONALE

101745 The *pax* utility was new for the ISO POSIX-2:1993 standard. It represents a peaceful compromise  
101746 between advocates of the historical *tar* and *cpio* utilities.

101747 A fundamental difference between *cpio* and *tar* was in the way directories were treated. The *cpio*  
101748 utility did not treat directories differently from other files, and to select a directory and its  
101749 contents required that each file in the hierarchy be explicitly specified. For *tar*, a directory  
101750 matched every file in the file hierarchy it rooted.

101751 The *pax* utility offers both interfaces; by default, directories map into the file hierarchy they root.  
101752 The **-d** option causes *pax* to skip any file not explicitly referenced, as *cpio* historically did. The *tar*  
101753 *-style* behavior was chosen as the default because it was believed that this was the more  
101754 common usage and because *tar* is the more commonly available interface, as it was historically  
101755 provided on both System V and BSD implementations.

101756 The data interchange format specification in this volume of POSIX.1-2008 requires that processes  
101757 with “appropriate privileges” shall always restore the ownership and permissions of extracted  
101758 files exactly as archived. If viewed from the historic equivalence between superuser and  
101759 “appropriate privileges”, there are two problems with this requirement. First, users running as  
101760 superusers may unknowingly set dangerous permissions on extracted files. Second, it is  
101761 needlessly limiting, in that superusers cannot extract files and own them as superuser unless the  
101762 archive was created by the superuser. (It should be noted that restoration of ownerships and  
101763 permissions for the superuser, by default, is historical practice in *cpio*, but not in *tar*.) In order to  
101764 avoid these two problems, the *pax* specification has an additional “privilege” mechanism, the **-p**  
101765 option. Only a *pax* invocation with the privileges needed, and which has the **-p** option set using  
101766 the *e* specification character, has appropriate privileges to restore full ownership and permission  
101767 information.

101768 Note also that this volume of POSIX.1-2008 requires that the file ownership and access  
101769 permissions shall be set, on extraction, in the same fashion as the *creat()* function when provided  
101770 with the mode stored in the archive. This means that the file creation mask of the user is applied  
101771 to the file permissions.

101772 Users should note that directories may be created by *pax* while extracting files with permissions  
101773 that are different from those that existed at the time the archive was created. When extracting  
101774 sensitive information into a directory hierarchy that no longer exists, users are encouraged to set  
101775 their file creation mask appropriately to protect these files during extraction.

101776 The table of contents output is written to standard output to facilitate pipeline processing.

101777 An early proposal had hard links displaying for all pathnames. This was removed because it  
101778 complicates the output of the case where **-v** is not specified and does not match historical *cpio*  
101779 usage. The hard-link information is available in the **-v** display.

101780 The description of the **-l** option allows implementations to make hard links to symbolic links. |  
101781 Earlier versions of this standard did not specify any way to create a hard link to a symbolic link, |  
101782 but many implementations provided this capability as an extension. If there are hard links to |  
101783 symbolic links when an archive is created, the implementation is required to archive the hard |  
101784 link in the archive (unless **-H** or **-L** is specified). When in **read** mode and in **copy** mode,  
101785 implementations supporting hard links to symbolic links should use them when appropriate.

101786 The archive formats inherited from the POSIX.1-1990 standard have certain restrictions that have  
101787 been brought along from historical usage. For example, there are restrictions on the length of  
101788 pathnames stored in the archive. When *pax* is used in **copy(-rw)** mode (copying directory  
101789 hierarchies), the ability to use extensions from the **-xpax** format overcomes these restrictions.



The default *blocksize* value of 5120 bytes for *cpio* was selected because it is one of the standard block-size values for *cpio*, set when the **-B** option is specified. (The other default block-size value for *cpio* is 512 bytes, and this was considered to be too small.) The default block value of 10240 bytes for *tar* was selected because that is the standard block-size value for BSD *tar*. The maximum block size of 32256 bytes ( $2^{15}$ –512 bytes) is the largest multiple of 512 bytes that fits into a signed 16-bit tape controller transfer register. There are known limitations in some historical systems that would prevent larger blocks from being accepted. Historical values were chosen to improve compatibility with historical scripts using *dd* or similar utilities to manipulate archives. Also, default block sizes for any file type other than character special file has been deleted from this volume of POSIX.1-2008 as unimportant and not likely to affect the structure of the resulting archive.

Implementations are permitted to modify the block-size value based on the archive format or the device to which the archive is being written. This is to provide implementations with the opportunity to take advantage of special types of devices, and it should not be used without a great deal of consideration as it almost certainly decreases archive portability.

The intended use of the **-n** option was to permit extraction of one or more files from the archive without processing the entire archive. This was viewed by the standard developers as offering significant performance advantages over historical implementations. The **-n** option in early proposals had three effects; the first was to cause special characters in patterns to not be treated specially. The second was to cause only the first file that matched a pattern to be extracted. The third was to cause *pax* to write a diagnostic message to standard error when no file was found matching a specified pattern. Only the second behavior is retained by this volume of POSIX.1-2008, for many reasons. First, it is in general not acceptable for a single option to have multiple effects. Second, the ability to make pattern matching characters act as normal characters is useful for parts of *pax* other than file extraction. Third, a finer degree of control over the special characters is useful because users may wish to normalize only a single special character in a single filename. Fourth, given a more general escape mechanism, the previous behavior of the **-n** option can be easily obtained using the **-s** option or a *sed* script. Finally, writing a diagnostic message when a pattern specified by the user is unmatched by any file is useful behavior in all cases.

In this version, the **-n** was removed from the **copy** mode synopsis of *pax*; it is inapplicable because there are no pattern operands specified in this mode.

There is another method than *pax* for copying subtrees in POSIX.1-2008 described as part of the *cp* utility. Both methods are historical practice: *cp* provides a simpler, more intuitive interface, while *pax* offers a finer granularity of control. Each provides additional functionality to the other; in particular, *pax* maintains the hard-link structure of the hierarchy while *cp* does not. It is the intention of the standard developers that the results be similar (using appropriate option combinations in both utilities). The results are not required to be identical; there seemed insufficient gain to applications to balance the difficulty of implementations having to guarantee that the results would be exactly identical.

A single archive may span more than one file. It is suggested that implementations provide informative messages to the user on standard error whenever the archive file is changed.

The **-d** option (do not create intermediate directories not listed in the archive) found in early proposals was originally provided as a complement to the historic **-d** option of *cpio*. It has been deleted.

The **-s** option in early proposals specified a subset of the substitution command from the *ed* utility. As there was no reason for only a subset to be supported, the **-s** option is now compatible with the current *ed* specification. Since the delimiter can be any non-null character, the following

usage with single <space> characters is valid:

```
pax -s " foo bar " ...
```

The **-t** description is worded so as to note that this may cause the access time update caused by some other activity (which occurs while the file is being read) to be overwritten.

The default behavior of *pax* with regard to file modification times is the same as historical implementations of *tar*. It is not the historical behavior of *cpio*.

Because the **-i** option uses **/dev/tty**, utilities without a controlling terminal are not able to use this option.

The **-y** option, found in early proposals, has been deleted because a line containing a single <period> for the **-i** option has equivalent functionality. The special lines for the **-i** option (a single <period> and the empty line) are historical practice in *cpio*.

In early drafts, a **-echarmap** option was included to increase portability of files between systems using different coded character sets. This option was omitted because it was apparent that consensus could not be formed for it. In this version, the use of UTF-8 should be an adequate substitute.

The ISO POSIX-2:1993 standard and ISO POSIX-1 standard requirements for *pax*, however, made it very difficult to create a single archive containing files created using extended characters provided by different locales. This version adds the **hdrcharset** keyword to make it possible to archive files in these cases without dropping files due to translation errors.

Translating filenames and other attributes from a locale's encoding to UTF-8 and then back again can lose information, as the resulting filename might not be byte-for-byte equivalent to the original. To avoid this problem, users can specify the **-o hdrcharset=binary** option, which will cause the resulting archive to use binary format for all names and attributes. Such archives are not portable among hosts that use different native encodings (e.g., EBCDIC *versus* ASCII-based encodings), but they will allow interchange among the vast majority of POSIX file systems in practical use. Also, the **-o hdrcharset=binary** option will cause *pax* in **copy** mode to behave more like other standard utilities such as *cp*.

If the values specified by the **-o exthdr.name=value**, **-o globexthdr.name=value**, or by **\$TMPDIR** (if **-o globexthdr.name** is not specified) require a character encoding other than that described in the ISO/IEC 646:1991 standard, a **path** extended header record will have to be created for the file. If a **hdrcharset** extended header record is active for such headers, it will determine the codeset used for the value field in these extended **path** header records. These **path** extended header records always need to be created when writing an archive even if **hdrcharset=binary** has been specified and would contain the same (binary) data that appears in the **ustar** header record prefix and *name* fields. (In other words, an extended header **path** record is always required to be generated if the *prefix* or *name* fields contain non-ASCII characters even when **hdrcharset=binary** is also in effect for that file.)

The **-k** option was added to address international concerns about the dangers involved in the character set transformations of **-e** (if the target character set were different from the source, the filenames might be transformed into names matching existing files) and also was made more general to protect files transferred between file systems with different {NAME\_MAX} values (truncating a filename on a smaller system might also inadvertently overwrite existing files). As stated, it prevents any overwriting, even if the target file is older than the source. This version adds more granularity of options to solve this problem by introducing the **-oinvalid=option**—specifically the **UTF-8** and **binary** actions. (Note that an existing file is still subject to overwriting in this case. The **-k** option closes that loophole.)

Some of the file characteristics referenced in this volume of POSIX.1-2008 might not be supported by some archive formats. For example, neither the **tar** nor **cpio** formats contain the file access time. For this reason, the **e** specification character has been provided, intended to cause all file characteristics specified in the archive to be retained.

It is required that extracted directories, by default, have their access and modification times and permissions set to the values specified in the archive. This has obvious problems in that the directories are almost certainly modified after being extracted and that directory permissions may not permit file creation. One possible solution is to create directories with the mode specified in the archive, as modified by the *umask* of the user, with sufficient permissions to allow file creation. After all files have been extracted, *pax* would then reset the access and modification times and permissions as necessary.

The list-mode formatting description borrows heavily from the one defined by the *printf* utility. However, since there is no separate operand list to get conversion arguments, the format was extended to allow specifying the name of the conversion argument as part of the conversion specification.

The **T** conversion specifier allows time fields to be displayed in any of the date formats. Unlike the *ls* utility, *pax* does not adjust the format when the date is less than six months in the past. This makes parsing the output more predictable.

The **D** conversion specifier handles the ability to display the major/minor or file size, as with *ls*, by using `%-8(size)D`.

The **L** conversion specifier handles the *ls* display for symbolic links.

Conversion specifiers were added to generate existing known types used for *ls*.

### **pax Interchange Format**

The new POSIX data interchange format was developed primarily to satisfy international concerns that the **ustar** and **cpio** formats did not provide for file, user, and group names encoded in characters outside a subset of the ISO/IEC 646:1991 standard. The standard developers realized that this new POSIX data interchange format should be very extensible because there were other requirements they foresaw in the near future:

- Support international character encodings and locale information
- Support security information (ACLs, and so on)
- Support future file types, such as realtime or contiguous files
- Include data areas for implementation use
- Support systems with words larger than 32 bits and timers with subsecond granularity

The following were not goals for this format because these are better handled by separate utilities or are inappropriate for a portable format:

- Encryption
- Compression
- Data translation between locales and codesets
- *inode* storage

The format chosen to support the goals is an extension of the **ustar** format. Of the two formats previously available, only the **ustar** format was selected for extensions because:

- It was easier to extend in an upwards-compatible way. It offered version flags and header block type fields with room for future standardization. The **cpio** format, while possessing a more flexible file naming methodology, could not be extended without breaking some theoretical implementation or using a dummy filename that could be a legitimate filename.
- Industry experience since the original “tar wars” fought in developing the ISO POSIX-1 standard has clearly been in favor of the **ustar** format, which is generally the default output format selected for *pax* implementations on new systems.

The new format was designed with one additional goal in mind: reasonable behavior when an older *tar* or *pax* utility happened to read an archive. Since the POSIX.1-1990 standard mandated that a “format-reading utility” had to treat unrecognized *typeflag* values as regular files, this allowed the format to include all the extended information in a pseudo-regular file that preceded each real file. An option is given that allows the archive creator to set up reasonable names for these files on the older systems. Also, the normative text suggests that reasonable file access values be used for this **ustar** header block. Making these header files inaccessible for convenient reading and deleting would not be reasonable. File permissions of 600 or 700 are suggested.

The **ustar** *typeflag* field was used to accommodate the additional functionality of the new format rather than magic or version because the POSIX.1-1990 standard (and, by reference, the previous version of *pax*), mandated the behavior of the format-reading utility when it encountered an unknown *typeflag*, but was silent about the other two fields.

Early proposals for the first version of this standard contained a proposed archive format that was based on compatibility with the standard for tape files (ISO 1001, similar to the format used historically on many mainframes and minicomputers). This format was overly complex and required considerable overhead in volume and header records. Furthermore, the standard developers felt that it would not be acceptable to the community of POSIX developers, so it was later changed to be a format more closely related to historical practice on POSIX systems.

The prefix and name split of pathnames in **ustar** was replaced by the single path extended header record for simplicity.

The concept of a global extended header (*typeflag g*) was controversial. If this were applied to an archive being recorded on magnetic tape, a few unreadable blocks at the beginning of the tape could be a serious problem; a utility attempting to extract as many files as possible from a damaged archive could lose a large percentage of file header information in this case. However, if the archive were on a reliable medium, such as a CD-ROM, the global extended header offers considerable potential size reductions by eliminating redundant information. Thus, the text warns against using the global method for unreliable media and provides a method for implanting global information in the extended header for each file, rather than in the *typeflag g* records.

No facility for data translation or filtering on a per-file basis is included because the standard developers could not invent an interface that would allow this in an efficient manner. If a filter, such as encryption or compression, is to be applied to all the files, it is more efficient to apply the filter to the entire archive as a single file. The standard developers considered interfaces that would invoke a shell script for each file going into or out of the archive, but the system overhead in this approach was considered to be too high.

One such approach would be to have **filter=** records that give a pathname for an executable. When the program is invoked, the file and archive would be open for standard input/output and all the header fields would be available as environment variables or command-line arguments. The standard developers did discuss such schemes, but they were omitted from POSIX.1-2008 due to concerns about excessive overhead. Also, the program itself would need to

|        |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|--------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 101973 | be in the archive if it were to be used portably.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 101974 | There is currently no portable means of identifying the character set(s) used for a file in the file system. Therefore, <i>pax</i> has not been given a mechanism to generate charset records automatically. The only portable means of doing this is for the user to write the archive using the <b>-ocharset=string</b> command line option. This assumes that all of the files in the archive use the same encoding. The “implementation-defined” text is included to allow for a system that can identify the encodings used for each of its files.                                                                                               |
| 101975 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 101976 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 101977 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 101978 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 101979 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 101980 | The table of standards that accompanies the charset record description is acknowledged to be very limited. Only a limited number of character set standards is reasonable for maximal interchange. Any character set is, of course, possible by prior agreement. It was suggested that EBCDIC be listed, but it was omitted because it is not defined by a formal standard. Formal standards, and then only those with reasonably large followings, can be included here, simply as a matter of practicality. The <i>&lt;value&gt;s</i> represent names of officially registered character sets in the format required by the ISO 2375:1985 standard. |
| 101981 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 101982 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 101983 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 101984 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 101985 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 101986 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 101987 | The normal <i>&lt;comma&gt;</i> or <i>&lt;blank&gt;</i> -separated list rules are not followed in the case of keyword options to allow ease of argument parsing for <i>getopts</i> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 101988 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 101989 | Further information on character encodings is in <a href="#">pax Archive Character Set Encoding/Decoding</a> (on page 3065).                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 101990 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 101991 | The standard developers have reserved keyword name space for vendor extensions. It is suggested that the format to be used is:                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 101992 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 101993 | <i>VENDOR.keyword</i>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| 101994 | where <i>VENDOR</i> is the name of the vendor or organization in all uppercase letters. It is further suggested that the keyword following the <i>&lt;period&gt;</i> be named differently than any of the standard keywords so that it could be used for future standardization, if appropriate, by omitting the <i>VENDOR</i> prefix.                                                                                                                                                                                                                                                                                                                |
| 101995 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 101996 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 101997 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 101998 | The <i>&lt;length&gt;</i> field in the extended header record was included to make it simpler to step through the records, even if a record contains an unknown format (to a particular <i>pax</i> ) with complex interactions of special characters. It also provides a minor integrity checkpoint within the records to aid a program attempting to recover files from a damaged archive.                                                                                                                                                                                                                                                           |
| 101999 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 102000 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 102001 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 102002 | There are no extended header versions of the <i>devmajor</i> and <i>devminor</i> fields because the unspecified format <b>ustar</b> header field should be sufficient. If they are not, vendor-specific extended keywords (such as <i>VENDOR.devmajor</i> ) should be used.                                                                                                                                                                                                                                                                                                                                                                           |
| 102003 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 102004 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 102005 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 102006 | Device and <i>i</i> -number labeling of files was not adopted from <i>cpio</i> ; files are interchanged strictly on a symbolic name basis, as in <b>ustar</b> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 102007 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 102008 | Just as with the <b>ustar</b> format descriptions, the new format makes no special arrangements for multi-volume archives. Each of the <i>pax</i> archive types is assumed to be inside a single POSIX file and splitting that file over multiple volumes (diskettes, tape cartridges, and so on), processing their labels, and mounting each in the proper sequence are considered to be implementation details that cannot be described portably.                                                                                                                                                                                                   |
| 102009 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 102010 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 102011 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 102012 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 102013 | The <b>pax</b> format is intended for interchange, not only for backup on a single (family of) systems. It is not as densely packed as might be possible for backup:                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 102014 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 102015 | <ul style="list-style-type: none"> <li>• It contains information as coded characters that could be coded in binary.</li> <li>• It identifies extended records with name fields that could be omitted in favor of a fixed-field layout.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                     |
| 102016 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |

- It translates names into a portable character set and identifies locale-related information, both of which are probably unnecessary for backup.

The requirements on restoring from an archive are slightly different from the historical wording, allowing for non-monolithic privilege to bring forward as much as possible. In particular, attributes such as “high performance file” might be broadly but not universally granted while set-user-ID or *chown()* might be much more restricted. There is no implication in POSIX.1-2008 that the security information be honored after it is restored to the file hierarchy, in spite of what might be improperly inferred by the silence on that topic. That is a topic for another standard.

Links are recorded in the fashion described here because a link can be to any file type. It is desirable in general to be able to restore part of an archive selectively and restore all of those files completely. If the data is not associated with each link, it is not possible to do this. However, the data associated with a file can be large, and when selective restoration is not needed, this can be a significant burden. The archive is structured so that files that have no associated data can always be restored by the name of any link name of any link, and the user may choose whether data is recorded with each instance of a file that contains data. The format permits mixing of both types of links in a single archive; this can be done for special needs, and *pax* is expected to interpret such archives on input properly, despite the fact that there is no *pax* option that would force this mixed case on output. (When **-o linkdata** is used, the output must contain the duplicate data, but the implementation is free to include it or omit it when **-o linkdata** is not used.)

The time values are included as extended header records for those implementations needing more than the eleven octal digits allowed by the **ustar** format. Portable file timestamps cannot be negative. If *pax* encounters a file with a negative timestamp in **copy** or **write** mode, it can reject the file, substitute a non-negative timestamp, or generate a non-portable timestamp with a leading ‘-’. Even though some implementations can support finer file-time granularities than seconds, the normative text requires support only for seconds since the Epoch because the ISO POSIX-1 standard states them that way. The **ustar** format includes only *mtime*; the new format adds *atime* and *ctime* for symmetry. The *atime* access time restored to the file system will be affected by the **-p a** and **-p e** options. The *ctime* creation time (actually *inode* modification time) is described with appropriate privileges so that it can be ignored when writing to the file system. POSIX does not provide a portable means to change file creation time. Nothing is intended to prevent a non-portable implementation of *pax* from restoring the value.

The *gid*, *size*, and *uid* extended header records were included to allow expansion beyond the sizes specified in the regular *tar* header. New file system architectures are emerging that will exhaust the 12-digit size field. There are probably not many systems requiring more than 8 digits for user and group IDs, but the extended header values were included for completeness, allowing overrides for all of the decimal values in the *tar* header.

The standard developers intended to describe the effective results of *pax* with regard to file ownerships and permissions; implementations are not restricted in timing or sequencing the restoration of such, provided the results are as specified.

Much of the text describing the extended headers refers to use in “**write** or **copy** modes”. The **copy** mode references are due to the normative text: “The effect of the copy shall be as if the copied files were written to an archive file and then subsequently extracted ...”. There is certainly no way to test whether *pax* is actually generating the extended headers in **copy** mode, but the effects must be as if it had.

### pax Archive Character Set Encoding/Decoding

There is a need to exchange archives of files between systems of different native codesets. Filenames, group names, and user names must be preserved to the fullest extent possible when an archive is read on the receiving platform. Translation of the contents of files is not within the scope of the *pax* utility.

There will also be the need to represent characters that are not available on the receiving platform. These unsupported characters cannot be automatically folded to the local set of characters due to the chance of collisions. This could result in overwriting previous extracted files from the archive or pre-existing files on the system.

For these reasons, the codeset used to represent characters within the extended header records of the *pax* archive must be sufficiently rich to handle all commonly used character sets. The fields requiring translation include, at a minimum, filenames, user names, group names, and link pathnames. Implementations may wish to have localized extended keywords that use non-portable characters.

The standard developers considered the following options:

- The archive creator specifies the well-defined name of the source codeset. The receiver must then recognize the codeset name and perform the appropriate translations to the destination codeset.
- The archive creator includes within the archive the character mapping table for the source codeset used to encode extended header records. The receiver must then read the character mapping table and perform the appropriate translations to the destination codeset.
- The archive creator translates the extended header records in the source codeset into a canonical form. The receiver must then perform the appropriate translations to the destination codeset.

The approach that incorporates the name of the source codeset poses the problem of codeset name registration, and makes the archive useless to *pax* archive decoders that do not recognize that codeset.

Because parts of an archive may be corrupted, the standard developers felt that including the character map of the source codeset was too fragile. The loss of this one key component could result in making the entire archive useless. (The difference between this and the global extended header decision was that the latter has a workaround—duplicating extended header records on unreliable media—but this would be too burdensome for large character set maps.)

Both of the above approaches also put an undue burden on the *pax* archive receiver to handle the cross-product of all source and destination codesets.

To simplify the translation from the source codeset to the canonical form and from the canonical form to the destination codeset, the standard developers decided that the internal representation should be a stateless encoding. A stateless encoding is one where each codepoint has the same meaning, without regard to the decoder being in a specific state. An example of a stateful encoding would be the Japanese Shift-JIS; an example of a stateless encoding would be the ISO/IEC 646:1991 standard (equivalent to 7-bit ASCII).

For these reasons, the standard developers decided to adopt a canonical format for the representation of file information strings. The obvious, well-endorsed candidate is the ISO/IEC 10646-1:2000 standard (based in part on Unicode), which can be used to represent the characters of virtually all standardized character sets. The standard developers initially agreed upon using UCS2 (16-bit Unicode) as the internal representation. This repertoire of characters

provides a sufficiently rich set to represent all commonly-used codesets.

However, the standard developers found that the 16-bit Unicode representation had some problems. It forced the issue of standardizing byte ordering. The 2-byte length of each character made the extended header records twice as long for the case of strings coded entirely from historical 7-bit ASCII. For these reasons, the standard developers chose the UTF-8 defined in the ISO/IEC 10646-1:2000 standard. This multi-byte representation encodes UCS2 or UCS4 characters reliably and deterministically, eliminating the need for a canonical byte ordering. In addition, NUL octets and other characters possibly confusing to POSIX file systems do not appear, except to represent themselves. It was realized that certain national codesets take up more space after the encoding, due to their placement within the UCS range; it was felt that the usefulness of the encoding of the names outweighs the disadvantage of size increase for file, user, and group names.

The encoding of UTF-8 is as follows:

| UCS4 Hex Encoding   | UTF-8 Binary Encoding                                 |
|---------------------|-------------------------------------------------------|
| 00000000-0000007F   | 0xxxxxxx                                              |
| 00000080-000007FF   | 110xxxxx 10xxxxxx                                     |
| 00000800-0000FFFF   | 1110xxxx 10xxxxxx 10xxxxxx                            |
| 00010000-001FFFFFFF | 11110xxx 10xxxxxx 10xxxxxx 10xxxxxx                   |
| 00200000-03FFFFFFF  | 111110xx 10xxxxxx 10xxxxxx 10xxxxxx 10xxxxxx          |
| 04000000-7FFFFFFF   | 1111110x 10xxxxxx 10xxxxxx 10xxxxxx 10xxxxxx 10xxxxxx |

where each 'x' represents a bit value from the character being translated.

### ustar Interchange Format

The description of the **ustar** format reflects numerous enhancements over pre-1988 versions of the historical *tar* utility. The goal of these changes was not only to provide the functional enhancements desired, but also to retain compatibility between new and old versions. This compatibility has been retained. Archives written using the old archive format are compatible with the new format.

Implementors should be aware that the previous file format did not include a mechanism to archive directory type files. For this reason, the convention of using a filename ending with <slash> was adopted to specify a directory on the archive.

The total size of the *name* and *prefix* fields have been set to meet the minimum requirements for {PATH\_MAX}. If a pathname will fit within the *name* field, it is recommended that the pathname be stored there without the use of the *prefix* field. Although the name field is known to be too small to contain {PATH\_MAX} characters, the value was not changed in this version of the archive file format to retain backwards-compatibility, and instead the prefix was introduced. Also, because of the earlier version of the format, there is no way to remove the restriction on the *linkname* field being limited in size to just that of the *name* field.

The *size* field is required to be meaningful in all implementation extensions, although it could be zero. This is required so that the data blocks can always be properly counted.

It is suggested that if device special files need to be represented that cannot be represented in the standard format, that one of the extension types (A-Z) be used, and that the additional information for the special file be represented as data and be reflected in the *size* field.

Attempting to restore a special file type, where it is converted to ordinary data and conflicts with an existing filename, need not be specially detected by the utility. If run as an ordinary user, *pax* should not be able to overwrite the entries in, for example, */dev* in any case (whether the file is



converted to another type or not). If run as a privileged user, it should be able to do so, and it would be considered a bug if it did not. The same is true of ordinary data files and similarly named special files; it is impossible to anticipate the needs of the user (who could really intend to overwrite the file), so the behavior should be predictable (and thus regular) and rely on the protection system as required.

The value 7 in the *typeflag* field is intended to define how contiguous files can be stored in a **ustar** archive. POSIX.1-2008 does not require the contiguous file extension, but does define a standard way of archiving such files so that all conforming systems can interpret these file types in a meaningful and consistent manner. On a system that does not support extended file types, the *pax* utility should do the best it can with the file and go on to the next.

The file protection modes are those conventionally used by the *ls* utility. This is extended beyond the usage in the ISO POSIX-2 standard to support the “shared text” or “sticky” bit. It is intended that the conformance document should not document anything beyond the existence of and support of such a mode. Further extensions are expected to these bits, particularly with overloading the set-user-ID and set-group-ID flags.

### cpio Interchange Format

The reference to appropriate privileges in the **cpio** format refers to an error on standard output; the **ustar** format does not make comparable statements.

The model for this format was the historical System V *cpio-c* data interchange format. This model documents the portable version of the **cpio** format and not the binary version. It has the flexibility to transfer data of any type described within POSIX.1-2008, yet is extensible to transfer data types specific to extensions beyond POSIX.1-2008 (for example, contiguous files). Because it describes existing practice, there is no question of maintaining upwards-compatibility.

### cpio Header

There has been some concern that the size of the *c\_ino* field of the header is too small to handle those systems that have very large *inode* numbers. However, the *c\_ino* field in the header is used strictly as a hard-link resolution mechanism for archives. It is not necessarily the same value as the *inode* number of the file in the location from which that file is extracted.

The name *c\_magic* is based on historical usage.

### cpio Filename

For most historical implementations of the *cpio* utility, {PATH\_MAX} octets can be used to describe the pathname without the addition of any other header fields (the NUL character would be included in this count). {PATH\_MAX} is the minimum value for pathname size, documented as 256 bytes. However, an implementation may use *c\_namesize* to determine the exact length of the pathname. With the current description of the **<cpio.h>** header, this pathname size can be as large as a number that is described in six octal digits.

Two values are documented under the *c\_mode* field values to provide for extensibility for known file types:

**0110 000** Reserved for contiguous files. The implementation may treat the rest of the information for this archive like a regular file. If this file type is undefined, the implementation may create the file as a regular file.

This provides for extensibility of the **cpio** format while allowing for the ability to read old archives. Files of an unknown type may be read as “regular files” on some implementations. On a system that does not support extended file types, the *pax* utility should do the best it can with

102197 the file and go on to the next.

## 102198 FUTURE DIRECTIONS

102199 None.

## 102200 SEE ALSO

102201 [Chapter 2](#) (on page 2319), *cp*, *ed*, *getopts*, *ls*, *printf*

102202 XBD [Section 3.169](#) (on page 60), [Chapter 5](#) (on page 121), [Chapter 8](#) (on page 173), [Section 12.2](#)  
102203 (on page 215), [<cpio.h>](#)

102204 XSH *chown()*, *creat()*, *fstatat()*, *mkdir()*, *mkfifo()*, *utime()*, *write()*

## 102205 CHANGE HISTORY

102206 First released in Issue 4.

### 102207 Issue 5

102208 A note is added to the APPLICATION USAGE indicating that the **cpio** and **tar** formats can only  
102209 support files up to 8 gigabytes in size.

### 102210 Issue 6

102211 The *pax* utility is aligned with the IEEE P1003.2b draft standard:

- 102212 • Support has been added for symbolic links in the options and interchange formats.
- 102213 • A new format has been devised, based on extensions to **ustar**.
- 102214 • References to the “extended” **tar** and **cpio** formats derived from the POSIX.1-1990  
102215 standard have been changed to remove the “extended” adjective because this could cause  
102216 confusion with the extended *tar* header added in this version. (All references to *tar* are  
102217 actually to **ustar**.)

102218 The *TZ* entry is added to the ENVIRONMENT VARIABLES section.

102219 IEEE PASC Interpretation 1003.2 #168 is applied, clarifying that *mkdir()* and *mkfifo()* calls can  
102220 ignore an [EEXIST] error when extracting an archive.

102221 IEEE PASC Interpretation 1003.2 #180 is applied, clarifying how extracted files are created when  
102222 in **read** mode.

102223 IEEE PASC Interpretation 1003.2 #181 is applied, clarifying the description of the **-t** option.

102224 IEEE PASC Interpretation 1003.2 #195 is applied.

102225 IEEE PASC Interpretation 1003.2 #206 is applied, clarifying the handling of links for the **-H**, **-L**,  
102226 and **-l** options.

102227 IEEE Std 1003.1-2001/Cor 1-2002, item XCU/TC1/D6/35 is applied, adding the process ID of  
102228 the *pax* process into certain fields. This change provides a method for the implementation to  
102229 ensure that different instances of *pax* extracting a file named */a/b/foo* will not collide when  
102230 processing the extended header information associated with **foo**.

102231 IEEE Std 1003.1-2001/Cor 1-2002, item XCU/TC1/D6/36 is applied, changing **-x B** to **-x pax** in  
102232 the OPTIONS section.

102233 IEEE Std 1003.1-2001/Cor 2-2004, item XCU/TC2/D6/20 is applied, updating the SYNOPSIS to  
102234 be consistent with the normative text.

102235 IEEE Std 1003.1-2001/Cor 2-2004, item XCU/TC2/D6/21 is applied, updating the  
102236 DESCRIPTION to describe the behavior when files to be linked are symbolic links and the  
102237 system is not capable of making hard links to symbolic links.

- 102238 IEEE Std 1003.1-2001/Cor 2-2004, item XCU/TC2/D6/22 is applied, updating the OPTIONS  
102239 section to describe the behavior for how multiple **-odelete=pattern** options are to be handled.
- 102240 IEEE Std 1003.1-2001/Cor 2-2004, item XCU/TC2/D6/23 is applied, updating the **write** option  
102241 within the OPTIONS section.
- 102242 IEEE Std 1003.1-2001/Cor 2-2004, item XCU/TC2/D6/24 is applied, adding a paragraph into  
102243 the OPTIONS section that states that specifying more than one of the mutually-exclusive options  
102244 (**-H** and **-L**) is not considered an error and that the last option specified will determine the  
102245 behavior of the utility.
- 102246 IEEE Std 1003.1-2001/Cor 2-2004, item XCU/TC2/D6/25 is applied, removing the *ctime*  
102247 paragraph within the EXTENDED DESCRIPTION. There is a contradiction in the definition of  
102248 the *ctime* keyword for the *pax* extended header, in that the *st\_ctime* member of the **stat** structure  
102249 does not refer to a file creation time. No field in the standard **stat** structure from **<sys/stat.h>**  
102250 includes a file creation time.
- 102251 IEEE Std 1003.1-2001/Cor 2-2004, item XCU/TC2/D6/26 is applied, making it clear that *typeflag*  
102252 1 (**ustar** Interchange Format) applies not only to files that are hard-linked, but also to files that  
102253 are aliased via symbolic links.
- 102254 IEEE Std 1003.1-2001/Cor 2-2004, item XCU/TC2/D6/27 is applied, clarifying the *cpio c\_nlink*  
102255 field.
- 102256 **Issue 7**
- 102257 Austin Group Interpretations 1003.1-2001 #011, #036, #086, and #109 are applied.
- 102258 Austin Group Interpretation 1003.1-2001 #126 is applied, changing the description of the  
102259 *LC\_MESSAGES* environment variable.
- 102260 SD5-XCU-ERN-2 is applied, making **-c** and **-n** mutually-exclusive in the SYNOPSIS.
- 102261 SD5-XCU-ERN-3 is applied, revising the default behavior of **-H** and **-L**.
- 102262 SD5-XCU-ERN-5, SD5-XCU-ERN-6, SD5-XCU-ERN-7, SD5-XCU-ERN-60 are applied.
- 102263 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.
- 102264 The *pax* utility is no longer allowed to create separate identical symbolic links when extracting  
102265 linked symbolic links from an archive.
- 102266 POSIX.1-2008, Technical Corrigendum 1, XCU/TC1-2008/0128 [260], XCU/TC1-2008/0129 +  
102267 [261], XCU/TC1-2008/0130 [261], XCU/TC1-2008/0131 [313], and XCU/TC1-2008/0132 [233] +  
102268 are applied.

102269 **NAME**102270 `pr` — print files102271 **SYNOPSIS**

```
102272 pr [+page] [-column] [-adFmrt] [-e[char][gap]] [-h header] [-i[char][gap]]
102273 XSI [-l lines] [-n[char][width]] [-o offset] [-s[char]] [-w width] [-fp]
102274 [file...]
```

102275 **DESCRIPTION**

102276 The *pr* utility is a printing and pagination filter. If multiple input files are specified, each shall be  
 102277 read, formatted, and written to standard output. By default, the input shall be separated into  
 102278 66-line pages, each with:

- 102279 • A 5-line header that includes the page number, date, time, and the pathname of the file
- 102280 • A 5-line trailer consisting of blank lines

102281 If standard output is associated with a terminal, diagnostic messages shall be deferred until the  
 102282 *pr* utility has completed processing.

102283 When options specifying multi-column output are specified, output text columns shall be of  
 102284 equal width; input lines that do not fit into a text column shall be truncated. By default, text  
 102285 columns shall be separated with at least one <blank>.

102286 **OPTIONS**

102287 The *pr* utility shall conform to XBD [Section 12.2](#) (on page 215), except that: the *page* option has a  
 102288 '+' delimiter; *page* and *column* can be multi-digit numbers; some of the option-arguments are  
 102289 optional; and some of the option-arguments cannot be specified as separate arguments from the  
 102290 preceding option letter. In particular, the *-s* option does not allow the option letter to be  
 102291 separated from its argument, and the options *-e*, *-i*, and *-n* require that both arguments, if  
 102292 present, not be separated from the option letter.

102293 The following options shall be supported. In the following option descriptions, *column*, *lines*,  
 102294 *offset*, *page*, and *width* are positive decimal integers; *gap* is a non-negative decimal integer.

- |        |                      |                                                                                                          |
|--------|----------------------|----------------------------------------------------------------------------------------------------------|
| 102295 | <b>+page</b>         | Begin output at page number <i>page</i> of the formatted input.                                          |
| 102296 | <b>-column</b>       | Produce multi-column output that is arranged in <i>column</i> columns (the default shall                 |
| 102297 |                      | be 1) and is written down each column in the order in which the text is received                         |
| 102298 |                      | from the input file. This option should not be used with <i>-m</i> . The options <i>-e</i> and <i>-i</i> |
| 102299 |                      | shall be assumed for multiple text-column output. Whether or not text columns are                        |
| 102300 |                      | produced with identical vertical lengths is unspecified, but a text column shall                         |
| 102301 |                      | never exceed the length of the page (see the <i>-l</i> option). When used with <i>-t</i> , use the       |
| 102302 |                      | minimum number of lines to write the output.                                                             |
| 102303 | <b>-a</b>            | Modify the effect of the <i>-column</i> option so that the columns are filled across the                 |
| 102304 |                      | page in a round-robin order (for example, when <i>column</i> is 2, the first input line                  |
| 102305 |                      | heads column 1, the second heads column 2, the third is the second line in column                        |
| 102306 |                      | 1, and so on).                                                                                           |
| 102307 | <b>-d</b>            | Produce output that is double-spaced; append an extra <newline> following every                          |
| 102308 |                      | <newline> found in the input.                                                                            |
| 102309 | <b>-e[char][gap]</b> |                                                                                                          |
| 102310 |                      | Expand each input <tab> to the next greater column position specified by the                             |
| 102311 |                      | formula $n*gap+1$ , where <i>n</i> is an integer > 0. If <i>gap</i> is zero or is omitted, it shall      |
| 102312 |                      | default to 8. All <tab> characters in the input shall be expanded into the                               |
| 102313 |                      | appropriate number of <space> characters. If any non-digit character, <i>char</i> , is                   |
| 102314 |                      | specified, it shall be used as the input <tab>. If the first character of the <i>-e</i> option-          |

|        |     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|--------|-----|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 102315 |     | argument is a digit, the entire option-argument shall be assumed to be <i>gap</i> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| 102316 | XSI | <b>-f</b> Use a <form-feed> for new pages, instead of the default behavior that uses a sequence of <newline> characters. Pause before beginning the first page if the standard output is associated with a terminal.                                                                                                                                                                                                                                                                                                                                                |
| 102317 |     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 102318 |     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 102319 |     | <b>-F</b> Use a <form-feed> for new pages, instead of the default behavior that uses a sequence of <newline> characters.                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 102320 |     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 102321 |     | <b>-h header</b> Use the string <i>header</i> to replace the contents of the <i>file</i> operand in the page header.                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 102322 |     | <b>-i[char][gap]</b> In output, replace <space> characters with <tab> characters wherever one or more adjacent <space> characters reach column positions <i>gap</i> +1, 2* <i>gap</i> +1, 3* <i>gap</i> +1, and so on. If <i>gap</i> is zero or is omitted, default tab settings at every eighth column position shall be assumed. If any non-digit character, <i>char</i> , is specified, it shall be used as the output <tab>. If the first character of the <b>-i</b> option-argument is a digit, the entire option-argument shall be assumed to be <i>gap</i> . |
| 102323 |     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 102324 |     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 102325 |     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 102326 |     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 102327 |     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 102328 |     | <b>-l lines</b> Override the 66-line default and reset the page length to <i>lines</i> . If <i>lines</i> is not greater than the sum of both the header and trailer depths (in lines), the <i>pr</i> utility shall suppress both the header and trailer, as if the <b>-t</b> option were in effect.                                                                                                                                                                                                                                                                 |
| 102329 |     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 102330 |     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 102331 |     | <b>-m</b> Merge files. Standard output shall be formatted so the <i>pr</i> utility writes one line from each file specified by a <i>file</i> operand, side by side into text columns of equal fixed widths, in terms of the number of column positions. Implementations shall support merging of at least nine <i>file</i> operands.                                                                                                                                                                                                                                |
| 102332 |     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 102333 |     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 102334 |     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 102335 |     | <b>-n[char][width]</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 102336 |     | Provide <i>width</i> -digit line numbering (default for <i>width</i> shall be 5). The number shall occupy the first <i>width</i> column positions of each text column of default output or each line of <b>-m</b> output. If <i>char</i> (any non-digit character) is given, it shall be appended to the line number to separate it from whatever follows (default for <i>char</i> is a <tab>).                                                                                                                                                                     |
| 102337 |     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 102338 |     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 102339 |     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 102340 |     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 102341 |     | <b>-o offset</b> Each line of output shall be preceded by offset <space> characters. If the <b>-o</b> option is not specified, the default offset shall be zero. The space taken is in addition to the output line width (see the <b>-w</b> option below).                                                                                                                                                                                                                                                                                                          |
| 102342 |     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 102343 |     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 102344 |     | <b>-p</b> Pause before beginning each page if the standard output is directed to a terminal ( <i>pr</i> shall write an <alert> to standard error and wait for a <carriage-return> to be read on <i>/dev/tty</i> ).                                                                                                                                                                                                                                                                                                                                                  |
| 102345 |     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 102346 |     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 102347 |     | <b>-r</b> Write no diagnostic reports on failure to open files.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 102348 |     | <b>-s[char]</b> Separate text columns by the single character <i>char</i> instead of by the appropriate number of <space> characters (default for <i>char</i> shall be <tab>).                                                                                                                                                                                                                                                                                                                                                                                      |
| 102349 |     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 102350 |     | <b>-t</b> Write neither the five-line identifying header nor the five-line trailer usually supplied for each page. Quit writing after the last line of each file without spacing to the end of the page.                                                                                                                                                                                                                                                                                                                                                            |
| 102351 |     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 102352 |     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 102353 |     | <b>-w width</b> Set the width of the line to <i>width</i> column positions for multiple text-column output only. If the <b>-w</b> option is not specified and the <b>-s</b> option is not specified, the default width shall be 72. If the <b>-w</b> option is not specified and the <b>-s</b> option is specified, the default width shall be 512.                                                                                                                                                                                                                 |
| 102354 |     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 102355 |     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 102356 |     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 102357 |     | For single column output, input lines shall not be truncated.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |

102358 **OPERANDS**

102359 The following operand shall be supported:

102360 *file* A pathname of a file to be written. If no *file* operands are specified, or if a *file*  
 102361 operand is *'-'*, the standard input shall be used.

102362 **STDIN**

102363 The standard input shall be used only if no *file* operands are specified, or if a *file* operand is *'-'*.  
 102364 See the INPUT FILES section.

102365 **INPUT FILES**

102366 The input files shall be text files.

102367 The file */dev/tty* shall be used to read responses required by the *-p* option.102368 **ENVIRONMENT VARIABLES**102369 The following environment variables shall affect the execution of *pr*:

102370 *LANG* Provide a default value for the internationalization variables that are unset or null.  
 102371 (See XBD [Section 8.2](#) (on page 174) the precedence of internationalization variables  
 102372 used to determine the values of locale categories.)

102373 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
 102374 internationalization variables.

102375 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
 102376 characters (for example, single-byte as opposed to multi-byte characters in  
 102377 arguments and input files) and which characters are defined as printable (character  
 102378 class **print**). Non-printable characters are still written to standard output, but are  
 102379 not counted for the purpose for column-width and line-length calculations.

102380 *LC\_MESSAGES*

102381 Determine the locale that should be used to affect the format and contents of  
 102382 diagnostic messages written to standard error.

102383 *LC\_TIME* Determine the format of the date and time for use in writing header lines.102384 XSI *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

102385 *TZ* Determine the timezone used to calculate date and time strings written in header  
 102386 lines. If *TZ* is unset or null, an unspecified default timezone shall be used.

102387 **ASYNCHRONOUS EVENTS**

102388 If *pr* receives an interrupt while writing to a terminal, it shall flush all accumulated error  
 102389 messages to the screen before terminating.

102390 **STDOUT**

102391 The *pr* utility output shall be a paginated version of the original file (or files). This pagination  
 102392 shall be accomplished using either *<form-feed>* characters or a sequence of *<newline>*  
 102393 XSI characters, as controlled by the *-F* or *-f* option. Page headers shall be generated unless the *-t*  
 102394 option is specified. The page headers shall be of the form:

102395 *"\n\n%s %s Page %d\n\n\n", <output of date>, <file>, <page number>*

102396 In the POSIX locale, the *<output of date>* field, representing the date and time of last modification  
 102397 of the input file (or the current date and time if the input file is standard input), shall be  
 102398 equivalent to the output of the following command as it would appear if executed at the given  
 102399 time:

102400 *date "+%b %e %H:%M %Y"*

without the trailing <newline>, if the page being written is from standard input. If the page being written is not from standard input, in the POSIX locale, the same format shall be used, but the time used shall be the modification time of the file corresponding to *file* instead of the current time. When the *LC\_TIME* locale category is not set to the POSIX locale, a different format and order of presentation of this field may be used.

If the standard input is used instead of a *file* operand, the <*file*> field shall be replaced by a null string.

If the **-h** option is specified, the <*file*> field shall be replaced by the *header* argument.

#### 102409 **STDERR**

The standard error shall be used for diagnostic messages and for alerting the terminal when **-p** is specified.

#### 102412 **OUTPUT FILES**

None.

#### 102414 **EXTENDED DESCRIPTION**

None.

#### 102416 **EXIT STATUS**

The following exit values shall be returned:

0 Successful completion.

>0 An error occurred.

#### 102420 **CONSEQUENCES OF ERRORS**

Default.

#### 102422 **APPLICATION USAGE**

A conforming application must protect its first operand, if it starts with a <plus-sign>, by preceding it with the "--" argument that denotes the end of the options. For example, *pr+x* could be interpreted as an invalid page number or a *file* operand.

#### 102426 **EXAMPLES**

1. Print a numbered list of all files in the current directory:

```
ls -a | pr -n -h "Files in $(pwd)."

```

2. Print **file1** and **file2** as a double-spaced, three-column listing headed by "file list":

```
pr -3d -h "file list" file1 file2

```

3. Write **file1** on **file2**, expanding tabs to columns 10, 19, 28, ...:

```
pr -e9 -t <file1 >file2

```

#### 102433 **RATIONALE**

This utility is one of those that does not follow the Utility Syntax Guidelines because of its historical origins. The standard developers could have added new options that obeyed the guidelines (and marked the old options obsolescent) or devised an entirely new utility; there are examples of both actions in this volume of POSIX.1-2008. Because of its widespread use by historical applications, the standard developers decided to exempt this version of *pr* from many of the guidelines.

Implementations are required to accept option-arguments to the **-h**, **-l**, **-o**, and **-w** options whether presented as part of the same argument or as a separate argument to *pr*, as suggested by the Utility Syntax Guidelines. The **-n** and **-s** options, however, are specified as in historical

practice because they are frequently specified without their optional arguments. If a <blank> were allowed before the option-argument in these cases, a *file* operand could mistakenly be interpreted as an option-argument in historical applications.

The text about the minimum number of lines in multi-column output was included to ensure that a best effort is made in balancing the length of the columns. There are known historical implementations in which, for example, 60-line files are listed by *pr -2* as one column of 56 lines and a second of 4. Although this is not a problem when a full page with headers and trailers is produced, it would be relatively useless when used with *-t*.

Historical implementations of the *pr* utility have differed in the action taken for the *-f* option. BSD uses it as described here for the *-F* option; System V uses it to change trailing <newline> characters on each page to a <form-feed> and, if standard output is a TTY device, sends an <alert> to standard error and reads a line from */dev/tty* before the first page. There were strong arguments from both sides of this issue concerning historical practice and as a result the *-F* option was added. XSI-conformant systems support the System V historical actions for the *-f* option.

The <output of date> field in the *-l* format is specified only for the POSIX locale. As noted, the format can be different in other locales. No mechanism for defining this is present in this volume of POSIX.1-2008, as the appropriate vehicle is a message catalog; that is, the format should be specified as a “message”.

#### FUTURE DIRECTIONS

None.

#### SEE ALSO

*expand*, *lp*

XBD [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 215)

#### CHANGE HISTORY

First released in Issue 2.

#### Issue 6

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The *-p* option is added.

The normative text is reworded to avoid use of the term “must” for application requirements.

#### Issue 7

PASC Interpretation 1003.2-92 #151 (SD5-XCU-ERN-44) is applied.

Austin Group Interpretation 1003.1-2001 #093 is applied.

SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.



102478 **NAME**

102479       printf — write formatted output

102480 **SYNOPSIS**102481       printf *format* [*argument...*]102482 **DESCRIPTION**

102483       The *printf* utility shall write formatted operands to the standard output. The *argument* operands  
 102484       shall be formatted under control of the *format* operand.

102485 **OPTIONS**

102486       None.

102487 **OPERANDS**

102488       The following operands shall be supported:

102489       *format*       A string describing the format to use to write the remaining operands. See the  
 102490       EXTENDED DESCRIPTION section.

102491       *argument*   The strings to be written to standard output, under the control of *format*. See the  
 102492       EXTENDED DESCRIPTION section.

102493 **STDIN**

102494       Not used.

102495 **INPUT FILES**

102496       None.

102497 **ENVIRONMENT VARIABLES**102498       The following environment variables shall affect the execution of *printf*:

102499       *LANG*       Provide a default value for the internationalization variables that are unset or null.  
 102500       (See XBD [Section 8.2](#) (on page 174) the precedence of internationalization variables  
 102501       used to determine the values of locale categories.)

102502       *LC\_ALL*      If set to a non-empty string value, override the values of all the other  
 102503       internationalization variables.

102504       *LC\_CTYPE*   Determine the locale for the interpretation of sequences of bytes of text data as  
 102505       characters (for example, single-byte as opposed to multi-byte characters in  
 102506       arguments).

102507       *LC\_MESSAGES*

102508       Determine the locale that should be used to affect the format and contents of  
 102509       diagnostic messages written to standard error.

102510       *LC\_NUMERIC*

102511       Determine the locale for numeric formatting. It shall affect the format of numbers  
 102512       written using the e, E, f, g, and G conversion specifier characters (if supported).

102513 *XSI*       *NLSPATH*   Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

102514 **ASYNCHRONOUS EVENTS**

102515       Default.

102516 **STDOUT**

102517       See the EXTENDED DESCRIPTION section.

102518 **STDERR**

102519 The standard error shall be used only for diagnostic messages.

102520 **OUTPUT FILES**

102521 None.

102522 **EXTENDED DESCRIPTION**

102523 The *format* operand shall be used as the *format* string described in XBD [Chapter 5](#) (on page 121)  
 102524 with the following exceptions:

- 102525 1. A <space> in the format string, in any context other than a flag of a conversion  
 102526 specification, shall be treated as an ordinary character that is copied to the output.
- 102527 2. A 'Δ' character in the format string shall be treated as a 'Δ' character, not as a <space>.
- 102528 3. In addition to the escape sequences shown in XBD [Chapter 5](#) (on page 121) ('\\', '\a',  
 102529 '\b', '\f', '\n', '\r', '\t', '\v'), "\ddd", where *ddd* is a one, two, or three-digit  
 102530 octal number, shall be written as a byte with the numeric value specified by the octal  
 102531 number.
- 102532 4. The implementation shall not precede or follow output from the d or u conversion  
 102533 specifiers with <blank> characters not specified by the *format* operand.
- 102534 5. The implementation shall not precede output from the o conversion specifier with zeros  
 102535 not specified by the *format* operand.
- 102536 6. The a, A, e, E, f, F, g, and G conversion specifiers need not be supported.
- 102537 7. An additional conversion specifier character, b, shall be supported as follows. The  
 102538 argument shall be taken to be a string that may contain <backslash>-escape sequences.  
 102539 The following <backslash>-escape sequences shall be supported:
  - 102540 — The escape sequences listed in XBD [Chapter 5](#) (on page 121) ('\\', '\a', '\b',  
 102541 '\f', '\n', '\r', '\t', '\v'), which shall be converted to the characters they  
 102542 represent
  - 102543 — "\0ddd", where *ddd* is a zero, one, two, or three-digit octal number that shall be  
 102544 converted to a byte with the numeric value specified by the octal number
  - 102545 — '\c', which shall not be written and shall cause *printf* to ignore any remaining  
 102546 characters in the string operand containing it, any remaining string operands, and  
 102547 any additional characters in the *format* operand

102548 The interpretation of a <backslash> followed by any other sequence of characters is  
 102549 unspecified.

102550 Bytes from the converted string shall be written until the end of the string or the number  
 102551 of bytes indicated by the precision specification is reached. If the precision is omitted, it  
 102552 shall be taken to be infinite, so all bytes up to the end of the converted string shall be  
 102553 written.

- 102554 8. For each conversion specification that consumes an argument, the next argument operand  
 102555 shall be evaluated and converted to the appropriate type for the conversion as specified  
 102556 below.
- 102557 9. The *format* operand shall be reused as often as necessary to satisfy the argument  
 102558 operands. Any extra c or s conversion specifiers shall be evaluated as if a null string  
 102559 argument were supplied; other extra conversion specifications shall be evaluated as if a  
 102560 zero argument were supplied. If the *format* operand contains no conversion specifications  
 102561 and *argument* operands are present, the results are unspecified.

10. If a character sequence in the *format* operand begins with a '%' character, but does not form a valid conversion specification, the behavior is unspecified.
11. The argument to the *c* conversion specifier can be a string containing zero or more bytes. If it contains one or more bytes, the first byte shall be written and any additional bytes shall be ignored. If the argument is an empty string, it is unspecified whether nothing is written or a null byte is written.

The *argument* operands shall be treated as strings if the corresponding conversion specifier is *b*, *c*, or *s*, and shall be evaluated as if by the *strtod()* function if the corresponding conversion specifier is *a*, *A*, *e*, *E*, *f*, *F*, *g*, or *G*. Otherwise, they shall be evaluated as unsuffixed C integer constants, as described by the ISO C standard, with the following extensions:

- A leading <plus-sign> or minus-sign shall be allowed.
- If the leading character is a single-quote or double-quote, the value shall be the numeric value in the underlying codeset of the character following the single-quote or double-quote.
- Suffixed integer constants may be allowed.

If an argument operand cannot be completely converted into an internal value appropriate to the corresponding conversion specification, a diagnostic message shall be written to standard error and the utility shall not exit with a zero exit status, but shall continue processing any remaining operands and shall write the value accumulated at the time the error was detected to standard output.

It is not considered an error if an argument operand is not completely used for a *c* or *s* conversion.

## EXIT STATUS

The following exit values shall be returned:

- 0 Successful completion.
- >0 An error occurred.

## CONSEQUENCES OF ERRORS

Default.

## APPLICATION USAGE

The floating-point formatting conversion specifications of *printf()* are not required because all arithmetic in the shell is integer arithmetic. The *awk* utility performs floating-point calculations and provides its own **printf** function. The *bc* utility can perform arbitrary-precision floating-point arithmetic, but does not provide extensive formatting capabilities. (This *printf* utility cannot really be used to format *bc* output; it does not support arbitrary precision.) Implementations are encouraged to support the floating-point conversions as an extension.

Note that this *printf* utility, like the *printf()* function defined in the System Interfaces volume of POSIX.1-2008 on which it is based, makes no special provision for dealing with multi-byte characters when using the *%c* conversion specification or when a precision is specified in a *%b* or *%s* conversion specification. Applications should be extremely cautious using either of these features when there are multi-byte characters in the character set.

No provision is made in this volume of POSIX.1-2008 which allows field widths and precisions to be specified as '\*' since the '\*' can be replaced directly in the *format* operand using shell variable substitution. Implementations can also provide this feature as an extension if they so choose.

Hexadecimal character constants as defined in the ISO C standard are not recognized in the *format* operand because there is no consistent way to detect the end of the constant. Octal character constants are limited to, at most, three octal digits, but hexadecimal character constants are only terminated by a non-hex-digit character. In the ISO C standard, the "##" concatenation operator can be used to terminate a constant and follow it with a hexadecimal character to be written. In the shell, concatenation occurs before the *printf* utility has a chance to parse the end of the hexadecimal constant.

The %b conversion specification is not part of the ISO C standard; it has been added here as a portable way to process <backslash>-escapes expanded in string operands as provided by the *echo* utility. See also the APPLICATION USAGE section of *echo* (on page 2639) for ways to use *printf* as a replacement for all of the traditional versions of the *echo* utility.

If an argument cannot be parsed correctly for the corresponding conversion specification, the *printf* utility is required to report an error. Thus, overflow and extraneous characters at the end of an argument being used for a numeric conversion shall be reported as errors.

## EXAMPLES

To alert the user and then print and read a series of prompts:

```
printf "\aPlease fill in the following: \nName: "
read name
printf "Phone number: "
read phone
```

To read out a list of right and wrong answers from a file, calculate the percentage correctly, and print them out. The numbers are right-justified and separated by a single <tab>. The percentage is written to one decimal place of accuracy:

```
while read right wrong ; do
 percent=$(echo "scale=1; ($right*100)/($right+$wrong)" | bc)
 printf "%2d right\t%2d wrong\t(%s%%)\n" \
 $right $wrong $percent
done < database_file
```

The command:

```
printf "%5d%4d\n" 1 21 321 4321 54321
```

produces:

```
 1 21
 3214321
54321 0
```

Note that the *format* operand is used three times to print all of the given strings and that a '0' was supplied by *printf* to satisfy the last %4d conversion specification.

The *printf* utility is required to notify the user when conversion errors are detected while producing numeric output; thus, the following results would be expected on an implementation with 32-bit twos-complement integers when %d is specified as the *format* operand:

| Argument    | Standard Output | Diagnostic Output                                |
|-------------|-----------------|--------------------------------------------------|
| 5a          | 5               | <b>printf: "5a" not completely converted</b>     |
| 9999999999  | 2147483647      | <b>printf: "9999999999" arithmetic overflow</b>  |
| -9999999999 | -2147483648     | <b>printf: "-9999999999" arithmetic overflow</b> |
| ABC         | 0               | <b>printf: "ABC" expected numeric value</b>      |

The diagnostic message format is not specified, but these examples convey the type of information that should be reported. Note that the value shown on standard output is what would be expected as the return value from the *strtol()* function as defined in the System Interfaces volume of POSIX.1-2008. A similar correspondence exists between *%u* and *strtoul()* and *%e*, *%f*, and *%g* (if the implementation supports floating-point conversions) and *strtod()*.

In a locale using the ISO/IEC 646: 1991 standard as the underlying codeset, the command:

```
printf "%d\n" 3 +3 -3 \'3 \"+3 "'-3"
```

produces:

```
3 Numeric value of constant 3
```

```
3 Numeric value of constant 3
```

```
-3 Numeric value of constant -3
```

```
51 Numeric value of the character '3' in the ISO/IEC 646: 1991 standard codeset
```

```
43 Numeric value of the character '+' in the ISO/IEC 646: 1991 standard codeset
```

```
45 Numeric value of the character '-' in the ISO/IEC 646: 1991 standard codeset
```

Note that in a locale with multi-byte characters, the value of a character is intended to be the value of the equivalent of the *wchar\_t* representation of the character as described in the System Interfaces volume of POSIX.1-2008.

## RATIONALE

The *printf* utility was added to provide functionality that has historically been provided by *echo*. However, due to irreconcilable differences in the various versions of *echo* extant, the version has few special features, leaving those to this new *printf* utility, which is based on one in the Ninth Edition system.

The EXTENDED DESCRIPTION section almost exactly matches the *printf()* function in the ISO C standard, although it is described in terms of the file format notation in XBD [Chapter 5](#) (on page 121).

Earlier versions of this standard specified that arguments for all conversions other than *b*, *c*, and *s* were evaluated in the same way (as C constants, but with stated exceptions). For implementations supporting the floating-point conversions it was not clear whether integer conversions need only accept integer constants and floating-point conversions need only accept floating-point constants, or whether both types of conversions should accept both types of constants. Also by not distinguishing between them, the requirement relating to a leading single-quote or double-quote applied to floating-point conversions even though this provided no useful functionality to applications that was not already available through the integer conversions. The current standard clarifies the situation by specifying that the arguments for floating-point conversions are evaluated as if by *strtod()*, and the arguments for integer conversions are evaluated as C integer constants, with the special treatment of leading single-quote and double-quote applying only to integer conversions.

102688 **FUTURE DIRECTIONS**

102689       None.

102690 **SEE ALSO**102691       *awk, bc, echo*102692       XBD [Chapter 5](#) (on page 121), [Chapter 8](#) (on page 173)102693       XSH *fprintf()*, *strtod()*102694 **CHANGE HISTORY**

102695       First released in Issue 4.

102696 **Issue 7**

102697       Austin Group Interpretations 1003.1-2001 #175 and #177 are applied.

102698       SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

102699 **NAME**102700        **prs** — print an SCCS file (**DEVELOPMENT**)102701 **SYNOPSIS**

```

102702 XSI prs [-a] [-d dataspec] [-r[SID]] file...
102703 prs [-e|-l] -c cutoff [-d dataspec] file...
102704 prs [-e|-l] -r[SID] [-d dataspec] file...

```

102705 **DESCRIPTION**

102706        The *prs* utility shall write to standard output parts or all of an SCCS file in a user-supplied  
 102707        format.

102708 **OPTIONS**

102709        The *prs* utility shall conform to XBD [Section 12.2](#) (on page 215), except that the **-r** option has an  
 102710        optional option-argument. This optional option-argument cannot be presented as a separate  
 102711        argument. The following options shall be supported:

102712        **-d *dataspec***    Specify the output data specification. The *dataspec* shall be a string consisting of  
 102713        SCCS file *data keywords* (see [Data Keywords](#), on page 3082) interspersed with  
 102714        optional user-supplied text.

102715        **-r[*SID*]**        Specify the SCCS identification string (SID) of a delta for which information is  
 102716        desired. If no *SID* option-argument is specified, the SID of the most recently  
 102717        created delta shall be assumed.

102718        **-e**                Request information for all deltas created earlier than and including the delta  
 102719        designated via the **-r** option or the date-time given by the **-c** option.

102720        **-l**                Request information for all deltas created later than and including the delta  
 102721        designated via the **-r** option or the date-time given by the **-c** option.

102722        **-c *cutoff***        Indicate the *cutoff* date-time, in the form:

102723        *YY*[*MM*[*DD*[*HH*[*MM*[*SS*]]]]]

102724        For the *YY* component, values in the range [69,99] shall refer to years 1969 to 1999  
 102725        inclusive, and values in the range [00,68] shall refer to years 2000 to 2068 inclusive.

102726        **Note:**        It is expected that in a future version of this standard the default century inferred  
 102727        from a 2-digit year will change. (This would apply to all commands accepting a  
 102728        2-digit year as input.)

102729        No changes (deltas) to the SCCS file that were created after the specified *cutoff*  
 102730        date-time shall be included in the output. Units omitted from the date-time default  
 102731        to their maximum possible values; for example, **-c 7502** is equivalent to  
 102732        **-c 750228235959**.

102733        **-a**                Request writing of information for both removed—that is, *delta type=R* (see  
 102734        *rm~~del~~*)—and existing—that is, *delta type=D*,—deltas. If the **-a** option is not  
 102735        specified, information for existing deltas only shall be provided.

102736 **OPERANDS**

102737        The following operand shall be supported:

102738        *file*            A pathname of an existing SCCS file or a directory. If *file* is a directory, the *prs*  
 102739        utility shall behave as though each file in the directory were specified as a named  
 102740        file, except that non-SCCS files (last component of the pathname does not begin  
 102741        with **s**.) and unreadable files shall be silently ignored.

102742 If exactly one *file* operand appears, and it is '–', the standard input shall be read;  
 102743 each line of the standard input shall be taken to be the name of an SCCS file to be  
 102744 processed. Non-SCCS files and unreadable files shall be silently ignored.

#### 102745 STDIN

102746 The standard input shall be a text file used only when the *file* operand is specified as '–'. Each  
 102747 line of the text file shall be interpreted as an SCCS pathname.

#### 102748 INPUT FILES

102749 Any SCCS files displayed are files of an unspecified format.

#### 102750 ENVIRONMENT VARIABLES

102751 The following environment variables shall affect the execution of *prs*:

102752 *LANG* Provide a default value for the internationalization variables that are unset or null.  
 102753 (See XBD [Section 8.2](#) (on page 174) the precedence of internationalization variables  
 102754 used to determine the values of locale categories.)

102755 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
 102756 internationalization variables.

102757 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
 102758 characters (for example, single-byte as opposed to multi-byte characters in  
 102759 arguments and input files).

102760 *LC\_MESSAGES*

102761 Determine the locale that should be used to affect the format and contents of  
 102762 diagnostic messages written to standard error.

102763 *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

#### 102764 ASYNCHRONOUS EVENTS

102765 Default.

#### 102766 STDOUT

102767 The standard output shall be a text file whose format is dependent on the data keywords  
 102768 specified with the *–d* option.

#### 102769 Data Keywords

102770 Data keywords specify which parts of an SCCS file shall be retrieved and output. All parts of an  
 102771 SCCS file have an associated data keyword. A data keyword may appear in a *dataspec* multiple  
 102772 times.

102773 The information written by *prs* shall consist of:

- 102774 1. The user-supplied text
- 102775 2. Appropriate values (extracted from the SCCS file) substituted for the recognized data  
 102776 keywords in the order of appearance in the *dataspec*

102777 The format of a data keyword value shall either be simple ('S'), in which keyword substitution  
 102778 is direct, or multi-line ('M').

102779 User-supplied text shall be any text other than recognized data keywords. A <tab> shall be  
 102780 specified by '\t' and <newline> by '\n'. When the *–r* option is not specified, the default  
 102781 *dataspec* shall be:

102782 :PN::\n\n

102783 and the following *dataspec* shall be used for each selected delta:



102784 :Dt:\t:DL:\nMRs:\n:MR:COMMENTS:\n:C:

102785

102786

102787

102788

102789

102790

102791

102792

102793

102794

102795

102796

102797

102798

102799

102800

102801

102802

102803

102804

102805

102806

102807

102808

102809

102810

102811

102812

102813

102814

102815

102816

102817

102818

102819

102820

102821

102822

102823

102824

102825

102826

102827

102828

102829

102830

102831

102832

| SCCS File Data Keywords |                                                          |              |                |        |
|-------------------------|----------------------------------------------------------|--------------|----------------|--------|
| Keyword                 | Data Item                                                | File Section | Value          | Format |
| :Dt:                    | Delta information                                        | Delta Table  | See below*     | S      |
| :DL:                    | Delta line statistics                                    | "            | :Li:/:Ld:/:Lu: | S      |
| :Li:                    | Lines inserted by Delta                                  | "            | nnnnn***       | S      |
| :Ld:                    | Lines deleted by Delta                                   | "            | nnnnn***       | S      |
| :Lu:                    | Lines unchanged by Delta                                 | "            | nnnnn***       | S      |
| :DT:                    | Delta type                                               | "            | D or R         | S      |
| :I:                     | SCCS ID string (SID)                                     | "            | See below**    | S      |
| :R:                     | Release number                                           | "            | nnnn           | S      |
| :L:                     | Level number                                             | "            | nnnn           | S      |
| :B:                     | Branch number                                            | "            | nnnn           | S      |
| :S:                     | Sequence number                                          | "            | nnnn           | S      |
| :D:                     | Date delta created                                       | "            | :Dy:/:Dm:/:Dd: | S      |
| :Dy:                    | Year delta created                                       | "            | nn             | S      |
| :Dm:                    | Month delta created                                      | "            | nn             | S      |
| :Dd:                    | Day delta created                                        | "            | nn             | S      |
| :T:                     | Time delta created                                       | "            | :Th:::Tm:::Ts: | S      |
| :Th:                    | Hour delta created                                       | "            | nn             | S      |
| :Tm:                    | Minutes delta created                                    | "            | nn             | S      |
| :Ts:                    | Seconds delta created                                    | "            | nn             | S      |
| :P:                     | Programmer who created Delta                             | "            | logname        | S      |
| :DS:                    | Delta sequence number                                    | "            | nnnn           | S      |
| :DP:                    | Predecessor Delta sequence number                        | "            | nnnn           | S      |
| :DI:                    | Sequence number of deltas included, excluded, or ignored | "            | :Dn:/:Dx:/:Dg: | S      |
| :Dn:                    | Deltas included (sequence #)                             | "            | :DS: :DS: ...  | S      |
| :Dx:                    | Deltas excluded (sequence #)                             | "            | :DS: :DS: ...  | S      |
| :Dg:                    | Deltas ignored (sequence #)                              | "            | :DS: :DS: ...  | S      |
| :MR:                    | MR numbers for delta                                     | "            | text           | M      |
| :C:                     | Comments for delta                                       | "            | text           | M      |
| :UN:                    | User names                                               | User Names   | text           | M      |
| :FL:                    | Flag list                                                | Flags        | text           | M      |
| :Y:                     | Module type flag                                         | "            | text           | S      |
| :MF:                    | MR validation flag                                       | "            | yes or no      | S      |
| :MP:                    | MR validation program name                               | "            | text           | S      |
| :KF:                    | Keyword error, warning flag                              | "            | yes or no      | S      |
| :KV:                    | Keyword validation string                                | "            | text           | S      |
| :BF:                    | Branch flag                                              | "            | yes or no      | S      |
| :J:                     | Joint edit flag                                          | "            | yes or no      | S      |
| :LK:                    | Locked releases                                          | "            | :R: ...        | S      |
| :Q:                     | User-defined keyword                                     | "            | text           | S      |
| :M:                     | Module name                                              | "            | text           | S      |
| :FB:                    | Floor boundary                                           | "            | :R:            | S      |
| :CB:                    | Ceiling boundary                                         | "            | :R:            | S      |
| :Ds:                    | Default SID                                              | "            | :I:            | S      |
| :ND:                    | Null delta flag                                          | "            | yes or no      | S      |

102833  
102834  
102835  
102836  
102837  
102838  
102839  
102840  
102841  
102842

| SCCS File Data Keywords |                              |              |                          |        |
|-------------------------|------------------------------|--------------|--------------------------|--------|
| Keyword                 | Data Item                    | File Section | Value                    | Format |
| <b>:FD:</b>             | File descriptive text        | Comments     | <i>text</i>              | M      |
| <b>:BD:</b>             | Body                         | Body         | <i>text</i>              | M      |
| <b>:GB:</b>             | Gotten body                  | "            | <i>text</i>              | M      |
| <b>:W:</b>              | A form of <i>what</i> string | N/A          | <b>:Z::M:\t:I:</b>       | S      |
| <b>:A:</b>              | A form of <i>what</i> string | N/A          | <b>:Z::Y: :M: :I::Z:</b> | S      |
| <b>:Z:</b>              | <i>what</i> string delimiter | N/A          | @ (#)                    | S      |
| <b>:F:</b>              | SCCS filename                | N/A          | <i>text</i>              | S      |
| <b>:PN:</b>             | SCCS file pathname           | N/A          | <i>text</i>              | S      |

102843  
102844  
102845  
102846  
102847

- \* **:Dt:=:DT: :I: :D: :T: :P: :DS: :DP:**
- \*\* **:R::L::B::S:** if the delta is a branch delta (**:BF:= =yes**)  
**:R::L:** if the delta is not a branch delta (**:BF:= =no**)
- \*\*\* The line statistics are capped at 99 999. For example, if 100 000 lines were unchanged in a certain revision, **:Lu:** shall produce the value 99 999.

## 102848 STDERR

102849 The standard error shall be used only for diagnostic messages.

## 102850 OUTPUT FILES

102851 None.

## 102852 EXTENDED DESCRIPTION

102853 None.

## 102854 EXIT STATUS

102855 The following exit values shall be returned:

- 102856 0 Successful completion.
- 102857 >0 An error occurred.

## 102858 CONSEQUENCES OF ERRORS

102859 Default.

## 102860 APPLICATION USAGE

102861 None.

## 102862 EXAMPLES

- 102863 1. The following example:

102864 prs -d "User Names for :F: are:\n:UN:" s.file  
 102865 might write to standard output:  
 102866 User Names for s.file are:  
 102867 xyz  
 102868 131  
 102869 abc

- 102870 2. The following example:

102871 prs -d "Delta for pgm :M:: :I: - :D: By :P:" -r s.file  
 102872 might write to standard output:  
 102873 Delta for pgm main.c: 3.7 - 77/12/01 By cas

102874           3. As a special case:

102875            prs s.file

102876           might write to standard output:

102877            s.file:

102878            <blank line>

102879            D 1.1 77/12/01 00:00:00 cas 1 000000/00000/00000

102880            MRs:

102881            b178-12345

102882            b179-54321

102883            COMMENTS:

102884            this is the comment line for s.file initial delta

102885            <blank line>

102886            for each delta table entry of the **D** type. The only option allowed to be used with this

102887            special case is the **-a** option.

102888 **RATIONALE**

102889            None.

102890 **FUTURE DIRECTIONS**

102891            None.

102892 **SEE ALSO**

102893            [admin](#), [delta](#), [get](#), [what](#)

102894            XBD [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 215)

102895 **CHANGE HISTORY**

102896            First released in Issue 2.

102897 **Issue 5**

102898            The phrase “in which keyword substitution is followed by a <newline>” is deleted from the end

102899            of the second paragraph of [Data Keywords](#) (on page 3082).

102900            The interpretation of the YY component of the **-c cutoff** argument is noted.

102901 **Issue 6**

102902            The normative text is reworded to emphasize the term “shall” for implementation requirements.

102903            The Open Group Base Resolution bwg2001-007 is applied, updating the table in STDOUT with a

102904            note that line statistics are capped at 99 999 for the **:Li:**, **:Ld:**, **:Lu:**, and **:DL:** keywords.

102905            The Open Group Interpretation PIN4C.00009 is applied.

102906 **Issue 7**

102907            SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

102908 **NAME**

102909       ps — report process status

102910 **SYNOPSIS**

```
102911 XSI ps [-aA] [-defl] [-g grouplist] [-G grouplist]
102912 [-n namelist] [-o format]... [-p proclist] [-t termlist]
102913 [-u userlist] [-U userlist]
```

102914 **DESCRIPTION**

102915       The *ps* utility shall write information about processes, subject to having appropriate privileges to  
 102916       obtain information about those processes.

102917       By default, *ps* shall select all processes with the same effective user ID as the current user and the  
 102918       same controlling terminal as the invoker.

102919 **OPTIONS**

102920       The *ps* utility shall conform to XBD [Section 12.2](#) (on page 215).

102921       The following options shall be supported:

102922       **-a**       Write information for all processes associated with terminals. Implementations  
 102923       may omit session leaders from this list.

102924       **-A**       Write information for all processes.

102925 XSI       **-d**       Write information for all processes, except session leaders.

102926 XSI       **-e**       Write information for all processes. (Equivalent to **-A**.)

102927 XSI       **-f**       Generate a **full** listing. (See the STDOUT section for the contents of a **full** listing.)

102928 XSI       **-g grouplist** Write information for processes whose session leaders are given in *grouplist*. The  
 102929       application shall ensure that the *grouplist* is a single argument in the form of a  
 102930       <blank> or <comma>-separated list.

102931       **-G grouplist** Write information for processes whose real group ID numbers are given in  
 102932       *grouplist*. The application shall ensure that the *grouplist* is a single argument in the  
 102933       form of a <blank> or <comma>-separated list.

102934 XSI       **-l**       Generate a **long** listing. (See STDOUT for the contents of a **long** listing.)

102935 XSI       **-n namelist** Specify the name of an alternative system *namelist* file in place of the default. The  
 102936       name of the default file and the format of a *namelist* file are unspecified.

102937       **-o format**   Write information according to the format specification given in *format*. This is  
 102938       fully described in the STDOUT section. Multiple **-o** options can be specified; the  
 102939       format specification shall be interpreted as the <space>-separated concatenation of  
 102940       all the *format* option-arguments.

102941       **-p proclist** Write information for processes whose process ID numbers are given in *proclist*.  
 102942       The application shall ensure that the *proclist* is a single argument in the form of a  
 102943       <blank> or <comma>-separated list.

102944       **-t termlist** Write information for processes associated with terminals given in *termlist*. The  
 102945       application shall ensure that the *termlist* is a single argument in the form of a  
 102946       <blank> or <comma>-separated list. Terminal identifiers shall be given in an  
 102947 XSI       implementation-defined format. On XSI-conformant systems, they shall be given  
 102948       in one of two forms: the device's filename (for example, **tty04**) or, if the device's  
 102949       filename starts with **tty**, just the identifier following the characters **tty** (for example,  
 102950       "04").

|        |     |                              |                                                                                                                                                                                                                                                                                                                                                                               |
|--------|-----|------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 102951 | XSI | <b>-u <i>userlist</i></b>    | Write information for processes whose user ID numbers or login names are given in <i>userlist</i> . The application shall ensure that the <i>userlist</i> is a single argument in the form of a <blank> or <comma>-separated list. In the listing, the numerical user ID shall be written unless the <b>-f</b> option is used, in which case the login name shall be written. |
| 102952 |     |                              |                                                                                                                                                                                                                                                                                                                                                                               |
| 102953 |     |                              |                                                                                                                                                                                                                                                                                                                                                                               |
| 102954 |     |                              |                                                                                                                                                                                                                                                                                                                                                                               |
| 102955 |     |                              |                                                                                                                                                                                                                                                                                                                                                                               |
| 102956 |     | <b>-U <i>userlist</i></b>    | Write information for processes whose real user ID numbers or login names are given in <i>userlist</i> . The application shall ensure that the <i>userlist</i> is a single argument in the form of a <blank> or <comma>-separated list.                                                                                                                                       |
| 102957 |     |                              |                                                                                                                                                                                                                                                                                                                                                                               |
| 102958 |     |                              |                                                                                                                                                                                                                                                                                                                                                                               |
| 102959 | XSI |                              | With the exception of <b>-f</b> , <b>-l</b> , <b>-n <i>namelist</i></b> , and <b>-o <i>format</i></b> , all of the options shown are used to select processes. If any are specified, the default list shall be ignored and <i>ps</i> shall select the processes represented by the inclusive OR of all the selection-criteria options.                                        |
| 102960 |     |                              |                                                                                                                                                                                                                                                                                                                                                                               |
| 102961 |     |                              |                                                                                                                                                                                                                                                                                                                                                                               |
| 102962 |     | <b>OPERANDS</b>              |                                                                                                                                                                                                                                                                                                                                                                               |
| 102963 |     |                              | None.                                                                                                                                                                                                                                                                                                                                                                         |
| 102964 |     | <b>STDIN</b>                 |                                                                                                                                                                                                                                                                                                                                                                               |
| 102965 |     |                              | Not used.                                                                                                                                                                                                                                                                                                                                                                     |
| 102966 |     | <b>INPUT FILES</b>           |                                                                                                                                                                                                                                                                                                                                                                               |
| 102967 |     |                              | None.                                                                                                                                                                                                                                                                                                                                                                         |
| 102968 |     | <b>ENVIRONMENT VARIABLES</b> |                                                                                                                                                                                                                                                                                                                                                                               |
| 102969 |     |                              | The following environment variables shall affect the execution of <i>ps</i> :                                                                                                                                                                                                                                                                                                 |
| 102970 |     | <b>COLUMNS</b>               | Override the system-selected horizontal display line size, used to determine the number of text columns to display. See XBD <a href="#">Chapter 8</a> (on page 173) for valid values and results when it is unset or null.                                                                                                                                                    |
| 102971 |     |                              |                                                                                                                                                                                                                                                                                                                                                                               |
| 102972 |     |                              |                                                                                                                                                                                                                                                                                                                                                                               |
| 102973 |     | <b>LANG</b>                  | Provide a default value for the internationalization variables that are unset or null. (See XBD <a href="#">Section 8.2</a> (on page 174) the precedence of internationalization variables used to determine the values of locale categories.)                                                                                                                                |
| 102974 |     |                              |                                                                                                                                                                                                                                                                                                                                                                               |
| 102975 |     |                              |                                                                                                                                                                                                                                                                                                                                                                               |
| 102976 |     | <b>LC_ALL</b>                | If set to a non-empty string value, override the values of all the other internationalization variables.                                                                                                                                                                                                                                                                      |
| 102977 |     |                              |                                                                                                                                                                                                                                                                                                                                                                               |
| 102978 |     | <b>LC_CTYPE</b>              | Determine the locale for the interpretation of sequences of bytes of text data as characters (for example, single-byte as opposed to multi-byte characters in arguments).                                                                                                                                                                                                     |
| 102979 |     |                              |                                                                                                                                                                                                                                                                                                                                                                               |
| 102980 |     |                              |                                                                                                                                                                                                                                                                                                                                                                               |
| 102981 |     | <b>LC_MESSAGES</b>           |                                                                                                                                                                                                                                                                                                                                                                               |
| 102982 |     |                              | Determine the locale that should be used to affect the format and contents of diagnostic messages written to standard error and informative messages written to standard output.                                                                                                                                                                                              |
| 102983 |     |                              |                                                                                                                                                                                                                                                                                                                                                                               |
| 102984 |     |                              |                                                                                                                                                                                                                                                                                                                                                                               |
| 102985 |     | <b>LC_TIME</b>               | Determine the format and contents of the date and time strings displayed.                                                                                                                                                                                                                                                                                                     |
| 102986 | XSI | <b>NLSPATH</b>               | Determine the location of message catalogs for the processing of <b>LC_MESSAGES</b> .                                                                                                                                                                                                                                                                                         |
| 102987 |     | <b>TZ</b>                    | Determine the timezone used to calculate date and time strings displayed. If <b>TZ</b> is unset or null, an unspecified default timezone shall be used.                                                                                                                                                                                                                       |
| 102988 |     |                              |                                                                                                                                                                                                                                                                                                                                                                               |
| 102989 |     | <b>ASYNCHRONOUS EVENTS</b>   |                                                                                                                                                                                                                                                                                                                                                                               |
| 102990 |     |                              | Default.                                                                                                                                                                                                                                                                                                                                                                      |

102991 **STDOUT**

102992 When the **-o** option is not specified, the standard output format is unspecified.

102993 XSI On XSI-conformant systems, the output format shall be as follows. The column headings and  
 102994 descriptions of the columns in a *ps* listing are given below. The precise meanings of these fields  
 102995 are implementation-defined. The letters 'f' and 'l' (below) indicate the option (**full** or **long**)  
 102996 that shall cause the corresponding heading to appear; **all** means that the heading always  
 102997 appears. Note that these two options determine only what information is provided for a process;  
 102998 they do not determine which processes are listed.

|        |              |       |                                                                         |
|--------|--------------|-------|-------------------------------------------------------------------------|
| 102999 | <b>F</b>     | (l)   | Flags (octal and additive) associated with the process.                 |
| 103000 | <b>S</b>     | (l)   | The state of the process.                                               |
| 103001 | <b>UID</b>   | (f,l) | The user ID number of the process owner; the login name is printed      |
| 103002 |              |       | under the <b>-f</b> option.                                             |
| 103003 | <b>PID</b>   | (all) | The process ID of the process; it is possible to kill a process if this |
| 103004 |              |       | datum is known.                                                         |
| 103005 | <b>PPID</b>  | (f,l) | The process ID of the parent process.                                   |
| 103006 | <b>C</b>     | (f,l) | Processor utilization for scheduling.                                   |
| 103007 | <b>PRI</b>   | (l)   | The priority of the process; higher numbers mean lower priority.        |
| 103008 | <b>NI</b>    | (l)   | Nice value; used in priority computation.                               |
| 103009 | <b>ADDR</b>  | (l)   | The address of the process.                                             |
| 103010 | <b>SZ</b>    | (l)   | The size in blocks of the core image of the process.                    |
| 103011 | <b>WCHAN</b> | (l)   | The event for which the process is waiting or sleeping; if blank, the   |
| 103012 |              |       | process is running.                                                     |
| 103013 | <b>STIME</b> | (f)   | Starting time of the process.                                           |
| 103014 | <b>TTY</b>   | (all) | The controlling terminal for the process.                               |
| 103015 | <b>TIME</b>  | (all) | The cumulative execution time for the process.                          |
| 103016 | <b>CMD</b>   | (all) | The command name; the full command name and its arguments are           |
| 103017 |              |       | written under the <b>-f</b> option.                                     |

103018 A process that has exited and has a parent, but has not yet been waited for by the parent, shall be  
 103019 marked **defunct**.

103020 Under the option **-f**, *ps* tries to determine the command name and arguments given when the  
 103021 process was created by examining memory or the swap area. Failing this, the command name, as  
 103022 it would appear without the option **-f**, is written in square brackets.

103023 The **-o** option allows the output format to be specified under user control.

103024 The application shall ensure that the format specification is a list of names presented as a single  
 103025 argument, <blank> or <comma>-separated. Each variable has a default header. The default  
 103026 header can be overridden by appending an <equals-sign> and the new text of the header. The  
 103027 rest of the characters in the argument shall be used as the header text. The fields specified shall  
 103028 be written in the order specified on the command line, and should be arranged in columns in the  
 103029 output. The field widths shall be selected by the system to be at least as wide as the header text  
 103030 (default or overridden value). If the header text is null, such as **-o user=**, the field width shall be  
 103031 at least as wide as the default header text. If all header text fields are null, no header line shall  
 103032 be written.

103033 The following names are recognized in the POSIX locale:

|        |              |                                                                                           |
|--------|--------------|-------------------------------------------------------------------------------------------|
| 103034 | <b>ruser</b> | The real user ID of the process. This shall be the textual user ID, if it can be obtained |
| 103035 |              | and the field width permits, or a decimal representation otherwise.                       |

|        |               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|--------|---------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 103036 | <b>user</b>   | The effective user ID of the process. This shall be the textual user ID, if it can be obtained and the field width permits, or a decimal representation otherwise.                                                                                                                                                                                                                                                                                                                                                                                       |
| 103037 |               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 103038 | <b>rgroup</b> | The real group ID of the process. This shall be the textual group ID, if it can be obtained and the field width permits, or a decimal representation otherwise.                                                                                                                                                                                                                                                                                                                                                                                          |
| 103039 |               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 103040 | <b>group</b>  | The effective group ID of the process. This shall be the textual group ID, if it can be obtained and the field width permits, or a decimal representation otherwise.                                                                                                                                                                                                                                                                                                                                                                                     |
| 103041 |               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 103042 | <b>pid</b>    | The decimal value of the process ID.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 103043 | <b>ppid</b>   | The decimal value of the parent process ID.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 103044 | <b>pgid</b>   | The decimal value of the process group ID.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| 103045 | <b>pcpu</b>   | The ratio of CPU time used recently to CPU time available in the same period, expressed as a percentage. The meaning of “recently” in this context is unspecified. The CPU time available is determined in an unspecified manner.                                                                                                                                                                                                                                                                                                                        |
| 103046 |               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 103047 |               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 103048 | <b>vsz</b>    | The size of the process in (virtual) memory in 1 024 byte units as a decimal integer.                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 103049 | <b>nice</b>   | The decimal value of the nice value of the process; see <i>nice</i> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 103050 | <b>etime</b>  | In the POSIX locale, the elapsed time since the process was started, in the form:                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 103051 |               | <code>[ [<i>dd</i>-] <i>hh</i>: ] <i>mm</i>: <i>ss</i></code>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 103052 |               | where <i>dd</i> shall represent the number of days, <i>hh</i> the number of hours, <i>mm</i> the number of minutes, and <i>ss</i> the number of seconds. The <i>dd</i> field shall be a decimal integer. The <i>hh</i> , <i>mm</i> , and <i>ss</i> fields shall be two-digit decimal integers padded on the left with zeros.                                                                                                                                                                                                                             |
| 103053 |               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 103054 |               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 103055 | <b>time</b>   | In the POSIX locale, the cumulative CPU time of the process in the form:                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| 103056 |               | <code>[ <i>dd</i>- ] <i>hh</i>:<i>mm</i>: <i>ss</i></code>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| 103057 |               | The <i>dd</i> , <i>hh</i> , <i>mm</i> , and <i>ss</i> fields shall be as described in the <b>etime</b> specifier.                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 103058 | <b>tty</b>    | The name of the controlling terminal of the process (if any) in the same format used by the <i>who</i> utility.                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 103059 |               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 103060 | <b>comm</b>   | The name of the command being executed ( <i>argv</i> [0] value) as a string.                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| 103061 | <b>args</b>   | The command with all its arguments as a string. The implementation may truncate this value to the field width; it is implementation-defined whether any further truncation occurs. It is unspecified whether the string represented is a version of the argument list as it was passed to the command when it started, or is a version of the arguments as they may have been modified by the application. Applications cannot depend on being able to modify their argument list and having that modification be reflected in the output of <i>ps</i> . |
| 103062 |               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 103063 |               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 103064 |               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 103065 |               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 103066 |               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 103067 |               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 103068 |               | Any field need not be meaningful in all implementations. In such a case a <hyphen> (‘-’) should be output in place of the field value.                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 103069 |               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 103070 |               | Only <b>comm</b> and <b>args</b> shall be allowed to contain <blank> characters; all others shall not. Any implementation-defined variables shall be specified in the system documentation along with the default header and indicating whether the field may contain <blank> characters.                                                                                                                                                                                                                                                                |
| 103071 |               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 103072 |               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 103073 |               | The following table specifies the default header to be used in the POSIX locale corresponding to each format specifier.                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 103074 |               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |

Table 4-18 Variable Names and Default Headers in *ps*

| Format Specifier | Default Header | Format Specifier | Default Header |
|------------------|----------------|------------------|----------------|
| <b>args</b>      | <b>COMMAND</b> | <b>ppid</b>      | <b>PPID</b>    |
| <b>comm</b>      | <b>COMMAND</b> | <b>rgroup</b>    | <b>RGROUP</b>  |
| <b>etime</b>     | <b>ELAPSED</b> | <b>ruser</b>     | <b>RUSER</b>   |
| <b>group</b>     | <b>GROUP</b>   | <b>time</b>      | <b>TIME</b>    |
| <b>nice</b>      | <b>NI</b>      | <b>tty</b>       | <b>TT</b>      |
| <b>pcpu</b>      | <b>%CPU</b>    | <b>user</b>      | <b>USER</b>    |
| <b>pgid</b>      | <b>PGID</b>    | <b>vsz</b>       | <b>VSZ</b>     |
| <b>pid</b>       | <b>PID</b>     |                  |                |

**STDERR**

The standard error shall be used only for diagnostic messages.

**OUTPUT FILES**

None.

**EXTENDED DESCRIPTION**

None.

**EXIT STATUS**

The following exit values shall be returned:

0 Successful completion.

>0 An error occurred.

**CONSEQUENCES OF ERRORS**

Default.

**APPLICATION USAGE**

Things can change while *ps* is running; the snapshot it gives is only true for an instant, and might not be accurate by the time it is displayed.

The **args** format specifier is allowed to produce a truncated version of the command arguments. In some implementations, this information is no longer available when the *ps* utility is executed.

If the field width is too narrow to display a textual ID, the system may use a numeric version. Normally, the system would be expected to choose large enough field widths, but if a large number of fields were selected to write, it might squeeze fields to their minimum sizes to fit on one line. One way to ensure adequate width for the textual IDs is to override the default header for a field to make it larger than most or all user or group names.

There is no special quoting mechanism for header text. The header text is the rest of the argument. If multiple header changes are needed, multiple **-o** options can be used, such as:

```
ps -o "user=User Name" -o pid=Process\ ID
```

On some implementations, especially multi-level secure systems, *ps* may be severely restricted and produce information only about child processes owned by the user.

**EXAMPLES**

The command:

```
ps -o user,pid,ppid=MOM -o args
```

writes at least the following in the POSIX locale:

```
USER PID MOM COMMAND
```



103117 helene 34 12 ps -o uid,pid,ppid=MOM -o args

103118 The contents of the **COMMAND** field need not be the same in all implementations, due to  
103119 possible truncation.

#### 103120 RATIONALE

103121 There is very little commonality between BSD and System V implementations of *ps*. Many  
103122 options conflict or have subtly different usages. The standard developers attempted to select a  
103123 set of options for the base standard that were useful on a wide range of systems and selected  
103124 options that either can be implemented on both BSD and System V-based systems without  
103125 breaking the current implementations or where the options are sufficiently similar that any  
103126 changes would not be unduly problematic for users or implementors.

103127 It is recognized that on some implementations, especially multi-level secure systems, *ps* may be  
103128 nearly useless. The default output has therefore been chosen such that it does not break  
103129 historical implementations and also is likely to provide at least some useful information on most  
103130 systems.

103131 The major change is the addition of the format specification capability. The motivation for this  
103132 invention is to provide a mechanism for users to access a wider range of system information, if  
103133 the system permits it, in a portable manner. The fields chosen to appear in this volume of  
103134 POSIX.1-2008 were arrived at after considering what concepts were likely to be both reasonably  
103135 useful to the “average” user and had a reasonable chance of being implemented on a wide range  
103136 of systems. Again it is recognized that not all systems are able to provide all the information  
103137 and, conversely, some may wish to provide more. It is hoped that the approach adopted will be  
103138 sufficiently flexible and extensible to accommodate most systems. Implementations may be  
103139 expected to introduce new format specifiers.

103140 The default output should consist of a short listing containing the process ID, terminal name,  
103141 cumulative execution time, and command name of each process.

103142 The preference of the standard developers would have been to make the format specification an  
103143 operand of the *ps* command. Unfortunately, BSD usage precluded this.

103144 At one time a format was included to display the environment array of the process. This was  
103145 deleted because there is no portable way to display it.

103146 The **-A** option is equivalent to the BSD **-g** and the SVID **-e**. Because the two systems differed, a  
103147 mnemonic compromise was selected.

103148 The **-a** option is described with some optional behavior because the SVID omits session leaders,  
103149 but BSD does not.

103150 In an early proposal, format specifiers appeared for priority and start time. The former was not  
103151 defined adequately in this volume of POSIX.1-2008 and was removed in deference to the defined  
103152 nice value; the latter because elapsed time was considered to be more useful.

103153 In a new BSD version of *ps*, a **-O** option can be used to write all of the default information,  
103154 followed by additional format specifiers. This was not adopted because the default output is  
103155 implementation-defined. Nevertheless, this is a useful option that should be reserved for that  
103156 purpose. In the **-o** option for the POSIX Shell and Utilities *ps*, the format is the concatenation of  
103157 each **-o**. Therefore, the user can have an alias or function that defines the beginning of their  
103158 desired format and add more fields to the end of the output in certain cases where that would be  
103159 useful.

103160 The format of the terminal name is unspecified, but the descriptions of *ps*, *talk*, *who*, and *write*  
103161 require that they all use the same format.

103162 The **pcpu** field indicates that the CPU time available is determined in an unspecified manner.  
103163 This is because it is difficult to express an algorithm that is useful across all possible machine  
103164 architectures. Historical counterparts to this value have attempted to show percentage of use in  
103165 the recent past, such as the preceding minute. Frequently, these values for all processes did not  
103166 add up to 100%. Implementations are encouraged to provide data in this field to users that will  
103167 help them identify processes currently affecting the performance of the system.

103168 **FUTURE DIRECTIONS**

103169 None.

103170 **SEE ALSO**

103171 *kill, nice, renice*

103172 XBD [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 215)

103173 **CHANGE HISTORY**

103174 First released in Issue 2.

103175 **Issue 6**

103176 This utility is marked as part of the User Portability Utilities option.

103177 The normative text is reworded to avoid use of the term “must” for application requirements.

103178 The *TZ* entry is added to the ENVIRONMENT VARIABLES section.

103179 **Issue 7**

103180 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

103181 SD5-XCU-ERN-148 is applied, updating the OPTIONS section.

103182 **NAME**103183        `pwd` — return working directory name103184 **SYNOPSIS**103185        `pwd [-L|-P]`103186 **DESCRIPTION**

103187        The *pwd* utility shall write to standard output an absolute pathname of the current working  
 103188        directory, which does not contain the filenames dot or dot-dot.

103189 **OPTIONS**103190        The *pwd* utility shall conform to XBD [Section 12.2](#) (on page 215).

103191        The following options shall be supported by the implementation:

103192        **-L**        If the *PWD* environment variable contains an absolute pathname of the current  
 103193        directory that does not contain the filenames dot or dot-dot, *pwd* shall write this  
 103194        pathname to standard output. Otherwise, if the *PWD* environment variable  
 103195        contains a pathname of the current directory that is longer than {PATH\_MAX}  
 103196        bytes including the terminating null, and the pathname does not contain any  
 103197        components that are dot or dot-dot, it is unspecified whether *pwd* writes this  
 103198        pathname to standard output or behaves as if the **-P** option had been specified.  
 103199        Otherwise, the **-L** option shall behave as the **-P** option.

103200        **-P**        The pathname written to standard output shall not contain any components that  
 103201        refer to files of type symbolic link. If there are multiple pathnames that the *pwd*  
 103202        utility could write to standard output, one beginning with a single <slash>  
 103203        character and one or more beginning with two <slash> characters, then it shall  
 103204        write the pathname beginning with a single <slash> character. The pathname shall  
 103205        not contain any unnecessary <slash> characters after the leading one or two  
 103206        <slash> characters.

103207        If both **-L** and **-P** are specified, the last one shall apply. If neither **-L** nor **-P** is specified, the *pwd*  
 103208        utility shall behave as if **-L** had been specified.

103209 **OPERANDS**

103210        None.

103211 **STDIN**

103212        Not used.

103213 **INPUT FILES**

103214        None.

103215 **ENVIRONMENT VARIABLES**103216        The following environment variables shall affect the execution of *pwd*:

103217        *LANG*        Provide a default value for the internationalization variables that are unset or null.  
 103218        (See XBD [Section 8.2](#) (on page 174) the precedence of internationalization variables  
 103219        used to determine the values of locale categories.)

103220        *LC\_ALL*        If set to a non-empty string value, override the values of all the other  
 103221        internationalization variables.

103222        *LC\_MESSAGES*

103223        Determine the locale that should be used to affect the format and contents of  
 103224        diagnostic messages written to standard error.

|        |     |                               |                                                                                                                                                              |
|--------|-----|-------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 103225 | XSI | <b>NLSPATH</b>                | Determine the location of message catalogs for the processing of <i>LC_MESSAGES</i> .                                                                        |
| 103226 |     | <b>PWD</b>                    | An absolute pathname of the current working directory. If an application sets or unsets the value of <i>PWD</i> , the behavior of <i>pwd</i> is unspecified. |
| 103227 |     |                               |                                                                                                                                                              |
| 103228 |     | <b>ASYNCHRONOUS EVENTS</b>    |                                                                                                                                                              |
| 103229 |     |                               | Default.                                                                                                                                                     |
| 103230 |     | <b>STDOUT</b>                 |                                                                                                                                                              |
| 103231 |     |                               | The <i>pwd</i> utility output is an absolute pathname of the current working directory:                                                                      |
| 103232 |     |                               | "%s\n", <directory pathname>                                                                                                                                 |
| 103233 |     | <b>STDERR</b>                 |                                                                                                                                                              |
| 103234 |     |                               | The standard error shall be used only for diagnostic messages.                                                                                               |
| 103235 |     | <b>OUTPUT FILES</b>           |                                                                                                                                                              |
| 103236 |     |                               | None.                                                                                                                                                        |
| 103237 |     | <b>EXTENDED DESCRIPTION</b>   |                                                                                                                                                              |
| 103238 |     |                               | None.                                                                                                                                                        |
| 103239 |     | <b>EXIT STATUS</b>            |                                                                                                                                                              |
| 103240 |     |                               | The following exit values shall be returned:                                                                                                                 |
| 103241 |     |                               | 0 Successful completion.                                                                                                                                     |
| 103242 |     |                               | >0 An error occurred.                                                                                                                                        |
| 103243 |     | <b>CONSEQUENCES OF ERRORS</b> |                                                                                                                                                              |
| 103244 |     |                               | If an error is detected, output shall not be written to standard output, a diagnostic message shall                                                          |
| 103245 |     |                               | be written to standard error, and the exit status is not zero.                                                                                               |
| 103246 |     | <b>APPLICATION USAGE</b>      |                                                                                                                                                              |
| 103247 |     |                               | If the pathname obtained from <i>pwd</i> is longer than {PATH_MAX} bytes, it could produce an error                                                          |
| 103248 |     |                               | if passed to <i>cd</i> . Therefore, in order to return to that directory it may be necessary to break the                                                    |
| 103249 |     |                               | pathname into sections shorter than {PATH_MAX} and call <i>cd</i> on each section in turn (the first                                                         |
| 103250 |     |                               | section being an absolute pathname and subsequent sections being relative pathnames).                                                                        |
| 103251 |     | <b>EXAMPLES</b>               |                                                                                                                                                              |
| 103252 |     |                               | None.                                                                                                                                                        |
| 103253 |     | <b>RATIONALE</b>              |                                                                                                                                                              |
| 103254 |     |                               | Some implementations have historically provided <i>pwd</i> as a shell special built-in command.                                                              |
| 103255 |     |                               | In most utilities, if an error occurs, partial output may be written to standard output. This does                                                           |
| 103256 |     |                               | not happen in historical implementations of <i>pwd</i> . Because <i>pwd</i> is frequently used in historical                                                 |
| 103257 |     |                               | shell scripts without checking the exit status, it is important that the historical behavior is                                                              |
| 103258 |     |                               | required here; therefore, the CONSEQUENCES OF ERRORS section specifically disallows any                                                                      |
| 103259 |     |                               | partial output being written to standard output.                                                                                                             |
| 103260 |     |                               | An earlier version of this standard stated that the <i>PWD</i> environment variable was affected when                                                        |
| 103261 |     |                               | the <b>-P</b> option was in effect. This was incorrect; conforming implementations do not do this.                                                           |
| 103262 |     | <b>FUTURE DIRECTIONS</b>      |                                                                                                                                                              |
| 103263 |     |                               | None.                                                                                                                                                        |

103264 **SEE ALSO**103265 *cd*103266 XBD [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 215)103267 XSH *getcwd()*103268 **CHANGE HISTORY**

103269 First released in Issue 2.

103270 **Issue 6**103271 The **-P** and **-L** options are added to describe actions relating to symbolic links as specified in the  
103272 IEEE P1003.2b draft standard.103273 **Issue 7**

103274 Austin Group Interpretation 1003.1-2001 #097 is applied.

103275 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

103276 Changes to the *pwd* utility and *PWD* environment variable have been made to match the  
103277 changes to the *getcwd()* function made for Austin Group Interpretation 1003.1-2001 #140.

103278 **NAME**

103279 qalter — alter batch job

103280 **SYNOPSIS**

```
103281 OB BE qalter [-a date_time] [-A account_string] [-c interval] [-e path_name]
103282 [-h hold_list] [-j join_list] [-k keep_list] [-l resource_list]
103283 [-m mail_options] [-M mail_list] [-N name] [-o path_name]
103284 [-p priority] [-r y|n] [-S path_name_list] [-u user_list]
103285 job_identifier...
```

103286 **DESCRIPTION**

103287 The attributes of a batch job are altered by a request to the batch server that manages the batch  
 103288 job. The *qalter* utility is a user-accessible batch client that requests the alteration of the attributes  
 103289 of one or more batch jobs.

103290 The *qalter* utility shall alter the attributes of those batch jobs, and only those batch jobs, for which  
 103291 a batch *job\_identifier* is presented to the utility.

103292 The *qalter* utility shall alter the attributes of batch jobs in the order in which the batch  
 103293 *job\_identifiers* are presented to the utility.

103294 If the *qalter* utility fails to process a batch *job\_identifier* successfully, the utility shall proceed to  
 103295 process the remaining batch *job\_identifiers*, if any.

103296 For each batch *job\_identifier* for which the *qalter* utility succeeds, each attribute of the identified  
 103297 batch job shall be altered as indicated by all the options presented to the utility.

103298 For each identified batch job for which the *qalter* utility fails, the utility shall not alter any  
 103299 attribute of the batch job.

103300 For each batch job that the *qalter* utility processes, the utility shall not modify any attribute other  
 103301 than those required by the options and option-arguments presented to the utility.

103302 The *qalter* utility shall alter batch jobs by sending a *Modify Job Request* to the batch server that  
 103303 manages each batch job. At the time the *qalter* utility exits, it shall have modified the batch job  
 103304 corresponding to each successfully processed batch *job\_identifier*. An attempt to alter the  
 103305 attributes of a batch job in the RUNNING state is implementation-defined.

103306 **OPTIONS**

103307 The *qalter* utility shall conform to XBD [Section 12.2](#) (on page 215).

103308 The following options shall be supported by the implementation:

103309 **-a date\_time** Redefine the time at which the batch job becomes eligible for execution.

103310 The *date\_time* argument shall be in the same form and represent the same time as  
 103311 for the *touch* utility. The time so represented shall be set into the *Execution\_Time*  
 103312 attribute of the batch job. If the time specified is earlier than the current time, the  
 103313 **-a** option shall have no effect.

103314 **-A account\_string**

103315 Redefine the account to which the resource consumption of the batch job should be  
 103316 charged.

103317 The syntax of the *account\_string* option-argument is unspecified.

103318 The *qalter* utility shall set the *Account\_Name* attribute of the batch job to the value  
 103319 of the *account\_string* option-argument.

103320       **-c interval**     Redefine whether the batch job should be checkpointed, and if so, how often.

103321                     The *qalter* utility shall accept a value for the interval option-argument that is one of

103322                     the following:

103323               n             No checkpointing is to be performed on the batch job

103324                     (NO\_CHECKPOINT).

103325               s             Checkpointing is to be performed only when the batch server is shut

103326                     down (CHECKPOINT\_AT\_SHUTDOWN).

103327               c             Automatic periodic checkpointing is to be performed at the

103328                     *Minimum\_Cpu\_Interval* attribute of the batch queue, in units of CPU

103329                     minutes (CHECKPOINT\_AT\_MIN\_CPU\_INTERVAL).

103330               c=*minutes*   Automatic periodic checkpointing is to be performed every *minutes*

103331                     of CPU time, or every *Minimum\_Cpu\_Interval* minutes, whichever is

103332                     greater. The *minutes* argument shall conform to the syntax for

103333                     unsigned integers and shall be greater than zero.

103334                     An implementation may define other checkpoint intervals. The conformance

103335                     document for an implementation shall describe any alternative checkpoint

103336                     intervals, how they are specified, their internal behavior, and how they affect the

103337                     behavior of the utility.

103338                     The *qalter* utility shall set the *Checkpoint* attribute of the batch job to the value of the

103339                     *interval* option-argument.

103340       **-e path\_name**     Redefine the path to be used for the standard error stream of the batch job.

103341                     The *qalter* utility shall accept a *path\_name* option-argument that conforms to the

103342                     syntax of the *path\_name* element defined in the System Interfaces volume of

103343                     POSIX.1-2008, which can be preceded by a host name element of the form

103344                     *hostname*::.

103345                     If the *path\_name* option-argument constitutes an absolute pathname, the *qalter*

103346                     utility shall set the *Error\_Path* attribute of the batch job to the value of the

103347                     *path\_name* option-argument, including the host name element, if present.

103348                     If the *path\_name* option-argument constitutes a relative pathname and no host

103349                     name element is specified, the *qalter* utility shall set the *Error\_Path* attribute of the

103350                     batch job to the value of the absolute pathname derived by expanding the

103351                     *path\_name* option-argument relative to the current directory of the process that

103352                     executes the *qalter* utility.

103353                     If the *path\_name* option-argument constitutes a relative pathname and a host name

103354                     element is specified, the *qalter* utility shall set the *Error\_Path* attribute of the batch

103355                     job to the value of the option-argument without expansion.

103356                     If the *path\_name* option-argument does not include a host name element, the *qalter*

103357                     utility shall prefix the pathname in the *Error\_Path* attribute with *hostname*::, where

103358                     *hostname* is the name of the host upon which the *qalter* utility is being executed.

103359                     103360       **-h hold\_list**     Redefine the types of holds, if any, on the batch job. The *qalter* **-h** option shall

103361                     accept a value for the *hold\_list* option-argument that is a string of alphanumeric

103362                     characters in the portable character set.

103363                     The *qalter* utility shall accept a value for the *hold\_list* option-argument that is a

103364 string of one or more of the characters 'u', 's', or 'o', or the single character  
 103365 'n'. For each unique character in the *hold\_list* option-argument, the *qalter* utility  
 103366 shall add a value to the *Hold\_Types* attribute of the batch job as follows, each  
 103367 representing a different hold type:

- 103368 u USER
- 103369 s SYSTEM
- 103370 o OPERATOR

103371 If any of these characters are duplicated in the *hold\_list* option-argument, the  
 103372 duplicates shall be ignored. An existing *Hold\_Types* attribute can be cleared by the  
 103373 hold type:

- 103374 n NO\_HOLD

103375 The *qalter* utility shall consider it an error if any hold type other than 'n' is  
 103376 combined with hold type 'n'. Strictly conforming applications shall not repeat  
 103377 any of the characters 'u', 's', 'o', or 'n' within the *hold\_list* option-argument.  
 103378 The *qalter* utility shall permit the repetition of characters, but shall not assign  
 103379 additional meaning to the repeated characters. An implementation may define  
 103380 other hold types. The conformance document for an implementation shall describe  
 103381 any additional hold types, how they are specified, their internal behavior, and how  
 103382 they affect the behavior of the utility.

103383 *-j join\_list* Redefine which streams of the batch job are to be merged. The *qalter* *-j* option shall  
 103384 accept a value for the *join\_list* option-argument that is a string of alphanumeric  
 103385 characters in the portable character set.

103386 The *qalter* utility shall accept a *join\_list* option-argument that consists of one or  
 103387 more of the characters 'e' and 'o', or the single character 'n'.

103388 All of the other batch job output streams specified shall be merged into the output  
 103389 stream represented by the character listed first in the *join\_list* option-argument.

103390 For each unique character in the *join\_list* option-argument, the *qalter* utility shall  
 103391 add a value to the *Join\_Path* attribute of the batch job as follows, each representing  
 103392 a different batch job stream to join:

- 103393 e The standard error of the batch job (JOIN\_STD\_ERROR).
- 103394 o The standard output of the batch job (JOIN\_STD\_OUTPUT).

103395 An existing *Join\_Path* attribute can be cleared by the join type:

- 103396 n NO\_JOIN

103397 If 'n' is specified, then no files are joined. The *qalter* utility shall consider it an  
 103398 error if any join type other than 'n' is combined with join type 'n'.

103399 Strictly conforming applications shall not repeat any of the characters 'e', 'o', or  
 103400 'n' within the *join\_list* option-argument. The *qalter* utility shall permit the  
 103401 repetition of characters, but shall not assign additional meaning to the repeated  
 103402 characters.

103403 An implementation may define other join types. The conformance document for an  
 103404 implementation shall describe any additional batch job streams, how they are  
 103405 specified, their internal behavior, and how they affect the behavior of the utility.



103406       **-k keep\_list**   Redefine which output of the batch job to retain on the execution host.

103407                   The *qalter* **-k** option shall accept a value for the *keep\_list* option-argument that is a

103408                   string of alphanumeric characters in the portable character set.

103409                   The *qalter* utility shall accept a *keep\_list* option-argument that consists of one or

103410                   more of the characters 'e' and 'o', or the single character 'n'.

103411                   For each unique character in the *keep\_list* option-argument, the *qalter* utility shall

103412                   add a value to the *Keep\_Files* attribute of the batch job as follows, each representing

103413                   a different batch job stream to keep:

103414                   e    The standard error of the batch job (KEEP\_STD\_ERROR).

103415                   o    The standard output of the batch job (KEEP\_STD\_OUTPUT).

103416                   If both 'e' and 'o' are specified, then both files are retained. An existing

103417                   *Keep\_Files* attribute can be cleared by the keep type:

103418                   n    NO\_KEEP

103419                   If 'n' is specified, then no files are retained. The *qalter* utility shall consider it an

103420                   error if any keep type other than 'n' is combined with keep type 'n'.

103421                   Strictly conforming applications shall not repeat any of the characters 'e', 'o', or

103422                   'n' within the *keep\_list* option-argument. The *qalter* utility shall permit the

103423                   repetition of characters, but shall not assign additional meaning to the repeated

103424                   characters. An implementation may define other keep types. The conformance

103425                   document for an implementation shall describe any additional keep types, how

103426                   they are specified, their internal behavior, and how they affect the behavior of the

103427                   utility.

103428       **-l resource\_list**   Redefine the resources that are allowed or required by the batch job.

103429                   The *qalter* utility shall accept a *resource\_list* option-argument that conforms to the

103430                   following syntax:

103431                   

103432                   resource=value[, , resource=value, , ...]

103433                   The *qalter* utility shall set one entry in the value of the *Resource\_List* attribute of the

103434                   batch job for each resource listed in the *resource\_list* option-argument.

103435                   Because the list of supported resource names might vary by batch server, the *qalter*

103436                   utility shall rely on the batch server to validate the resource names and associated

103437                   values. See [Section 3.3.3](#) (on page 2421) for a means of removing *keyword=value*

103438                   (and *value@keyword*) pairs and other general rules for list-oriented batch job

103439                   attributes.

103440       **-m mail\_options**   Redefine the points in the execution of the batch job at which the batch server is to

103441                   send mail about a change in the state of the batch job.

103442                   

103443                   The *qalter* **-m** option shall accept a value for the *mail\_options* option-argument that

103444                   is a string of alphanumeric characters in the portable character set.

103445                   The *qalter* utility shall accept a value for the *mail\_options* option-argument that is a

103446                   string of one or more of the characters 'e', 'b', and 'a', or the single character

103447                   'n'. For each unique character in the *mail\_options* option-argument, the *qalter*

103448                   utility shall add a value to the *Mail\_Users* attribute of the batch job as follows, each

103449 representing a different time during the life of a batch job at which to send mail:

103450 e MAIL\_AT\_EXIT

103451 b MAIL\_AT\_BEGINNING

103452 a MAIL\_AT\_ABORT

103453 If any of these characters are duplicated in the *mail\_options* option-argument, the

103454 duplicates shall be ignored.

103455 An existing *Mail\_Points* attribute can be cleared by the mail type:

103456 n NO\_MAIL

103457 If 'n' is specified, then mail is not sent. The *qalter* utility shall consider it an error

103458 if any mail type other than 'n' is combined with mail type 'n'. Strictly

103459 conforming applications shall not repeat any of the characters 'e', 'b', 'a', or

103460 'n' within the *mail\_options* option-argument. The *qalter* utility shall permit the

103461 repetition of characters but shall not assign additional meaning to the repeated

103462 characters.

103463 An implementation may define other mail types. The conformance document for

103464 an implementation shall describe any additional mail types, how they are

103465 specified, their internal behavior, and how they affect the behavior of the utility.

103466 **-M mail\_list** Redefine the list of users to which the batch server that executes the batch job is to

103467 send mail, if the batch server sends mail about the batch job.

103468 The syntax of the *mail\_list* option-argument is unspecified. If the implementation

103469 of the *qalter* utility uses a name service to locate users, the utility shall accept the

103470 syntax used by the name service.

103471 If the implementation of the *qalter* utility does not use a name service to locate

103472 users, the implementation shall accept the following syntax for user names:

103473 mail\_address[, mail\_address, ...]

103474 The interpretation of *mail\_address* is implementation-defined.

103475 The *qalter* utility shall set the *Mail\_Users* attribute of the batch job to the value of

103476 the *mail\_list* option-argument.

103477 **-N name** Redefine the name of the batch job.

103478 The *qalter* -N option shall accept a value for the *name* option-argument that is a

103479 string of up to 15 alphanumeric characters in the portable character set where the

103480 first character is alphabetic.

103481 The syntax of the *name* option-argument is unspecified.

103482 The *qalter* utility shall set the *Job\_Name* attribute of the batch job to the value of the

103483 *name* option-argument.

103484 **-o path\_name**

103485 Redefine the path for the standard output of the batch job.

103486 The *qalter* utility shall accept a *path\_name* option-argument that conforms to the

103487 syntax of the *path\_name* element defined in the System Interfaces volume of

103488 POSIX.1-2008, which can be preceded by a host name element of the form

103489 *hostname::*.

|        |                          |                                                                                                                                                                                                                                                                                                                                                                                   |
|--------|--------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 103490 |                          | If the <i>path_name</i> option-argument constitutes an absolute pathname, the <i>qalter</i> utility shall set the <i>Output_Path</i> attribute of the batch job to the value of the <i>path_name</i> option-argument.                                                                                                                                                             |
| 103491 |                          |                                                                                                                                                                                                                                                                                                                                                                                   |
| 103492 |                          |                                                                                                                                                                                                                                                                                                                                                                                   |
| 103493 |                          | If the <i>path_name</i> option-argument constitutes a relative pathname and no host name element is specified, the <i>qalter</i> utility shall set the <i>Output_Path</i> attribute of the batch job to the absolute pathname derived by expanding the <i>path_name</i> option-argument relative to the current directory of the process that executes the <i>qalter</i> utility. |
| 103494 |                          |                                                                                                                                                                                                                                                                                                                                                                                   |
| 103495 |                          |                                                                                                                                                                                                                                                                                                                                                                                   |
| 103496 |                          |                                                                                                                                                                                                                                                                                                                                                                                   |
| 103497 |                          |                                                                                                                                                                                                                                                                                                                                                                                   |
| 103498 |                          | If the <i>path_name</i> option-argument constitutes a relative pathname and a host name element is specified, the <i>qalter</i> utility shall set the <i>Output_Path</i> attribute of the batch job to the value of the <i>path_name</i> option-argument without any expansion of the pathname.                                                                                   |
| 103499 |                          |                                                                                                                                                                                                                                                                                                                                                                                   |
| 103500 |                          |                                                                                                                                                                                                                                                                                                                                                                                   |
| 103501 |                          |                                                                                                                                                                                                                                                                                                                                                                                   |
| 103502 |                          | If the <i>path_name</i> option-argument does not include a host name element, the <i>qalter</i> utility shall prefix the pathname in the <i>Output_Path</i> attribute with <i>hostname:</i> , where <i>hostname</i> is the name of the host upon which the <i>qalter</i> utility is being executed.                                                                               |
| 103503 |                          |                                                                                                                                                                                                                                                                                                                                                                                   |
| 103504 |                          |                                                                                                                                                                                                                                                                                                                                                                                   |
| 103505 | <b>-p priority</b>       | Redefine the priority of the batch job.                                                                                                                                                                                                                                                                                                                                           |
| 103506 |                          | The <i>qalter</i> utility shall accept a value for the priority option-argument that conforms to the syntax for signed decimal integers, and which is not less than -1 024 and not greater than 1 023.                                                                                                                                                                            |
| 103507 |                          |                                                                                                                                                                                                                                                                                                                                                                                   |
| 103508 |                          |                                                                                                                                                                                                                                                                                                                                                                                   |
| 103509 |                          | The <i>qalter</i> utility shall set the <i>Priority</i> attribute of the batch job to the value of the <i>priority</i> option-argument.                                                                                                                                                                                                                                           |
| 103510 |                          |                                                                                                                                                                                                                                                                                                                                                                                   |
| 103511 | <b>-r y   n</b>          | Redefine whether the batch job is rerunnable.                                                                                                                                                                                                                                                                                                                                     |
| 103512 |                          | If the value of the option-argument is 'y', the <i>qalter</i> utility shall set the <i>Rerunable</i> attribute of the batch job to TRUE.                                                                                                                                                                                                                                          |
| 103513 |                          |                                                                                                                                                                                                                                                                                                                                                                                   |
| 103514 |                          | If the value of the option-argument is 'n', the <i>qalter</i> utility shall set the <i>Rerunable</i> attribute of the batch job to FALSE.                                                                                                                                                                                                                                         |
| 103515 |                          |                                                                                                                                                                                                                                                                                                                                                                                   |
| 103516 |                          | The <i>qalter</i> utility shall consider it an error if any character other than 'y' or 'n' is specified in the option-argument.                                                                                                                                                                                                                                                  |
| 103517 |                          |                                                                                                                                                                                                                                                                                                                                                                                   |
| 103518 | <b>-S path_name_list</b> |                                                                                                                                                                                                                                                                                                                                                                                   |
| 103519 |                          | Redefine the shell that interprets the script at the destination system.                                                                                                                                                                                                                                                                                                          |
| 103520 |                          | The <i>qalter</i> utility shall accept a <i>path_name_list</i> option-argument that conforms to the following syntax:                                                                                                                                                                                                                                                             |
| 103521 |                          |                                                                                                                                                                                                                                                                                                                                                                                   |
| 103522 |                          | pathname[@host] [, pathname[@host] , ...]                                                                                                                                                                                                                                                                                                                                         |
| 103523 |                          | The <i>qalter</i> utility shall accept only one pathname that is missing a corresponding host name. The <i>qalter</i> utility shall allow only one pathname per named host.                                                                                                                                                                                                       |
| 103524 |                          |                                                                                                                                                                                                                                                                                                                                                                                   |
| 103525 |                          | The <i>qalter</i> utility shall add a value to the <i>Shell_Path_List</i> attribute of the batch job for each entry in the <i>path_name_list</i> option-argument. See <a href="#">Section 3.3.3</a> (on page 2421) for a means of removing <i>keyword=value</i> (and <i>value@keyword</i> ) pairs and other general rules for list-oriented batch job attributes.                 |
| 103526 |                          |                                                                                                                                                                                                                                                                                                                                                                                   |
| 103527 |                          |                                                                                                                                                                                                                                                                                                                                                                                   |
| 103528 |                          |                                                                                                                                                                                                                                                                                                                                                                                   |
| 103529 | <b>-u user_list</b>      | Redefine the user name under which the batch job is to run at the destination system.                                                                                                                                                                                                                                                                                             |
| 103530 |                          |                                                                                                                                                                                                                                                                                                                                                                                   |
| 103531 |                          | The <i>qalter</i> utility shall accept a <i>user_list</i> option-argument that conforms to the following syntax:                                                                                                                                                                                                                                                                  |
| 103532 |                          |                                                                                                                                                                                                                                                                                                                                                                                   |

103533                   username[@host] [, , username[@host] , , ...]

103534                   The *qalter* utility shall accept only one user name that is missing a corresponding  
103535                   host name. The *qalter* utility shall accept only one user name per named host.

103536                   The *qalter* utility shall add a value to the *User\_List* attribute of the batch job for each  
103537                   entry in the *user\_list* option-argument. See [Section 3.3.3](#) (on page 2421) for a means  
103538                   of removing *keyword=value* (and *value@keyword*) pairs and other general rules for  
103539                   list-oriented batch job attributes.

#### 103540 OPERANDS

103541                   The *qalter* utility shall accept one or more operands that conform to the syntax for a batch  
103542                   *job\_identifier* (see [Section 3.3.1](#), on page 2419).

#### 103543 STDIN

103544                   Not used.

#### 103545 INPUT FILES

103546                   None.

#### 103547 ENVIRONMENT VARIABLES

103548                   The following environment variables shall affect the execution of *qalter*:

103549                   *LANG*           Provide a default value for the internationalization variables that are unset or null.  
103550                   (See XBD [Section 8.2](#) (on page 174) the precedence of internationalization variables  
103551                   used to determine the values of locale categories.)

103552                   *LC\_ALL*          If set to a non-empty string value, override the values of all the other  
103553                   internationalization variables.

103554                   *LC\_CTYPE*       Determine the locale for the interpretation of sequences of bytes of text data as  
103555                   characters (for example, single-byte as opposed to multi-byte characters in  
103556                   arguments).

103557                   *LC\_MESSAGES*

103558                   Determine the locale that should be used to affect the format and contents of  
103559                   diagnostic messages written to standard error.

103560                   *LOGNAME*       Determine the login name of the user.

103561                   *TZ*             Determine the timezone used to interpret the *date-time* option-argument. If *TZ* is  
103562                   unset or null, an unspecified default timezone shall be used.

#### 103563 ASYNCHRONOUS EVENTS

103564                   Default.

#### 103565 STDOUT

103566                   None.

#### 103567 STDERR

103568                   The standard error shall be used only for diagnostic messages.

#### 103569 OUTPUT FILES

103570                   None.

#### 103571 EXTENDED DESCRIPTION

103572                   None.

103573 **EXIT STATUS**

103574 The following exit values shall be returned:

103575 0 Successful completion.

103576 >0 An error occurred.

103577 **CONSEQUENCES OF ERRORS**

103578 In addition to the default behavior, the *qalter* utility shall not be required to write a diagnostic  
 103579 message to standard error when the error reply received from a batch server indicates that the  
 103580 batch *job\_identifier* does not exist on the server. Whether or not the *qalter* utility attempts to locate  
 103581 the batch job on other batch servers is implementation-defined.

103582 **APPLICATION USAGE**

103583 None.

103584 **EXAMPLES**

103585 None.

103586 **RATIONALE**

103587 The *qalter* utility allows users to change the attributes of a batch job.

103588 As a means of altering a queued job, the *qalter* utility is superior to deleting and requeuing the  
 103589 batch job insofar as an altered job retains its place in the queue with some traditional selection  
 103590 algorithms. In addition, the *qalter* utility is both shorter and simpler than a sequence of *qdel* and  
 103591 *qsub* utilities.

103592 The result of an attempt on the part of a user to alter a batch job in a RUNNING state is  
 103593 implementation-defined because a batch job in the RUNNING state will already have opened its  
 103594 output files and otherwise performed any actions indicated by the options in effect at the time  
 103595 the batch job began execution.

103596 The options processed by the *qalter* utility are identical to those of the *qsub* utility, with a few  
 103597 exceptions: **-V**, **-v**, and **-q**. The **-V** and **-v** are inappropriate for the *qalter* utility, since they  
 103598 capture potentially transient environment information from the submitting process. The **-q**  
 103599 option would specify a new queue, which would largely negate the previously stated advantage  
 103600 of using *qalter*; furthermore, the *qmove* utility provides a superior means of moving jobs.

103601 Each of the following paragraphs provides the rationale for a *qalter* option.

103602 Additional rationale concerning these options can be found in the rationale for the *qsub* utility.

103603 The **-a** option allows users to alter the date and time at which a batch job becomes eligible to  
 103604 run.

103605 The **-A** option allows users to change the account that will be charged for the resources  
 103606 consumed by the batch job. Support for the **-A** option is mandatory for conforming  
 103607 implementations of *qalter*, even though support of accounting is optional for servers. Whether or  
 103608 not to support accounting is left to the implementor of the server, but mandatory support of the  
 103609 **-A** option assures users of a consistent interface and allows them to control accounting on  
 103610 servers that support accounting.

103611 The **-c** option allows users to alter the checkpointing interval of a batch job. A checkpointing  
 103612 system, which is not defined by POSIX.1-2008, allows recovery of a batch job at the most recent  
 103613 checkpoint in the event of a crash. Checkpointing is typically used for jobs that consume  
 103614 expensive computing time or must meet a critical schedule. Users should be allowed to make  
 103615 the tradeoff between the overhead of checkpointing and the risk to the timely completion of the  
 103616 batch job; therefore, this volume of POSIX.1-2008 provides the checkpointing interval option.  
 103617 Support for checkpointing is optional for servers.

The **-e** option allows users to alter the name and location of the standard error stream written by a batch job. However, the path of the standard error stream is meaningless if the value of the *Join\_Path* attribute of the batch job is TRUE.

The **-h** option allows users to set the hold type in the *Hold\_Types* attribute of a batch job. The *qhold* and *qrls* utilities add or remove hold types to the *Hold\_Types* attribute, respectively. The **-h** option has been modified to allow for implementation-defined hold types.

The **-j** option allows users to alter the decision to join (merge) the standard error stream of the batch job with the standard output stream of the batch job.

The **-l** option allows users to change the resource limits imposed on a batch job.

The **-m** option allows users to modify the list of points in the life of a batch job at which the designated users will receive mail notification.

The **-M** option allows users to alter the list of users who will receive notification about events in the life of a batch job.

The **-N** option allows users to change the name of a batch job.

The **-o** option allows users to alter the name and path to which the standard output stream of the batch job will be written.

The **-P** option allows users to modify the priority of a batch job. Support for priority is optional for batch servers.

The **-r** option allows users to alter the rerunability status of a batch job.

The **-S** option allows users to change the name and location of the shell image that will be invoked to interpret the script of the batch job. This option has been modified to allow a list of shell name and locations associated with different hosts.

The **-u** option allows users to change the user identifier under which the batch job will execute.

The *job\_identifier* operand syntax is provided so that the user can differentiate between the originating and destination (or executing) batch server. These may or may not be the same. The *.server\_name* portion identifies the originating batch server, while the *@server* portion identifies the destination batch server.

Historically, the *qalter* utility has been a component of the Network Queuing System (NQS), the existing practice from which this utility has been derived.

#### FUTURE DIRECTIONS

The *qalter* utility may be removed in a future version.

#### SEE ALSO

[Chapter 3](#) (on page 2397), *qdel*, *qhold*, *qmove*, *qrls*, *qsub*, *touch*

XBD [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 215)

#### CHANGE HISTORY

Derived from IEEE Std 1003.2d-1994.

#### Issue 6

The *TZ* entry is added to the ENVIRONMENT VARIABLES section.

IEEE PASC Interpretation 1003.2 #182 is applied, clarifying the description of the **-a** option.

103657 **Issue 7**

103658 The *qalter* utility is marked obsolescent.

103659 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

103660 **NAME**

103661 qdel — delete batch jobs

103662 **SYNOPSIS**103663 OB BE `qdel job_identifier...`103664 **DESCRIPTION**

103665 A batch job is deleted by sending a request to the batch server that manages the batch job. A  
 103666 batch job that has been deleted is no longer subject to management by batch services.

103667 The *qdel* utility is a user-accessible client of batch services that requests the deletion of one or  
 103668 more batch jobs.

103669 The *qdel* utility shall request a batch server to delete those batch jobs for which a batch  
 103670 *job\_identifier* is presented to the utility.

103671 The *qdel* utility shall delete batch jobs in the order in which their batch *job\_identifiers* are  
 103672 presented to the utility.

103673 If the *qdel* utility fails to process any batch *job\_identifier* successfully, the utility shall proceed to  
 103674 process the remaining batch *job\_identifiers*, if any.

103675 The *qdel* utility shall delete each batch job by sending a *Delete Job Request* to the batch server that  
 103676 manages the batch job.

103677 The *qdel* utility shall not exit until the batch job corresponding to each successfully processed  
 103678 batch *job\_identifier* has been deleted.

103679 **OPTIONS**

103680 None.

103681 **OPERANDS**

103682 The *qdel* utility shall accept one or more operands that conform to the syntax for a batch  
 103683 *job\_identifier* (see [Section 3.3.1](#), on page 2419).

103684 **STDIN**

103685 Not used.

103686 **INPUT FILES**

103687 None.

103688 **ENVIRONMENT VARIABLES**103689 The following environment variables shall affect the execution of *qdel*:

103690 *LANG* Provide a default value for the internationalization variables that are unset or null.  
 103691 (See XBD [Section 8.2](#) (on page 174) the precedence of internationalization variables  
 103692 used to determine the values of locale categories.)

103693 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
 103694 internationalization variables.

103695 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
 103696 characters (for example, single-byte as opposed to multi-byte characters in  
 103697 arguments).

103698 *LC\_MESSAGES*

103699 Determine the locale that should be used to affect the format and contents of  
 103700 diagnostic messages written to standard error.



- 103701 *LOGNAME* Determine the login name of the user.
- 103702 **ASYNCHRONOUS EVENTS**
- 103703 Default.
- 103704 **STDOUT**
- 103705 An implementation of the *qdel* utility may write informative messages to standard output.
- 103706 **STDERR**
- 103707 The standard error shall be used only for diagnostic messages.
- 103708 **OUTPUT FILES**
- 103709 None.
- 103710 **EXTENDED DESCRIPTION**
- 103711 None.
- 103712 **EXIT STATUS**
- 103713 The following exit values shall be returned:
- 103714 0 Successful completion.
- 103715 >0 An error occurred.
- 103716 **CONSEQUENCES OF ERRORS**
- 103717 In addition to the default behavior, the *qdel* utility shall not be required to write a diagnostic message to standard error when the error reply received from a batch server indicates that the batch *job\_identifier* does not exist on the server. Whether or not the *qdel* utility waits to output the diagnostic message while attempting to locate the job on other servers is implementation-defined.
- 103722 **APPLICATION USAGE**
- 103723 None.
- 103724 **EXAMPLES**
- 103725 None.
- 103726 **RATIONALE**
- 103727 The *qdel* utility allows users and administrators to delete jobs.
- 103728 The *qdel* utility provides functionality that is not otherwise available. For example, the *kill* utility of the operating system does not suffice. First, to use the *kill* utility, the user might have to log in on a remote node, because the *kill* utility does not operate across the network. Second, unlike *qdel*, *kill* cannot remove jobs from queues. Lastly, the arguments of the *qdel* utility are job identifiers rather than process identifiers, and so this utility can be passed the output of the *qselect* utility, thus providing users with a means of deleting a list of jobs.
- 103734 Because a set of jobs can be selected using the *qselect* utility, the *qdel* utility has not been complicated with options that provide for selection of jobs. Instead, the batch jobs to be deleted are identified individually by their job identifiers.
- 103737 Historically, the *qdel* utility has been a component of NQS, the existing practice on which it is based. However, the *qdel* utility defined in this volume of POSIX.1-2008 does not provide an option for specifying a signal number to send to the batch job prior to the killing of the process; that capability has been subsumed by the *qsig* utility.
- 103741 A discussion was held about the delays of networking and the possibility that the batch server may never respond, due to a down router, down batch server, or other network mishap. The DESCRIPTION records this under the words “fails to process any job identifier”. In the broad sense, the network problem is also an error, which causes the failure to process the batch job

103745 identifier.

103746 **FUTURE DIRECTIONS**

103747 The *qdel* utility may be removed in a future version.

103748 **SEE ALSO**

103749 [Chapter 3](#) (on page 2397), *kill*, *qselect*, *qsig*

103750 XBD [Chapter 8](#) (on page 173)

103751 **CHANGE HISTORY**

103752 Derived from IEEE Std 1003.2d-1994.

103753 **Issue 6**

103754 The *LC\_TIME* and *TZ* entries are removed from the ENVIRONMENT VARIABLES section.

103755 **Issue 7**

103756 The *qdel* utility is marked obsolescent.

103757 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

103758 **NAME**

103759 qhold — hold batch jobs

103760 **SYNOPSIS**103761 OB BE `qhold [-h hold_list] job_identifier...`103762 **DESCRIPTION**

103763 A hold is placed on a batch job by a request to the batch server that manages the batch job. A  
 103764 batch job that has one or more holds is not eligible for execution. The *qhold* utility is a user-  
 103765 accessible client of batch services that requests one or more types of hold to be placed on one or  
 103766 more batch jobs.

103767 The *qhold* utility shall place holds on those batch jobs for which a batch *job\_identifier* is presented  
 103768 to the utility.

103769 The *qhold* utility shall place holds on batch jobs in the order in which their batch *job\_identifiers*  
 103770 are presented to the utility. If the *qhold* utility fails to process any batch *job\_identifier* successfully,  
 103771 the utility shall proceed to process the remaining batch *job\_identifiers*, if any.

103772 The *qhold* utility shall place holds on each batch job by sending a *Hold Job Request* to the batch  
 103773 server that manages the batch job.

103774 The *qhold* utility shall not exit until holds have been placed on the batch job corresponding to  
 103775 each successfully processed batch *job\_identifier*.

103776 **OPTIONS**

103777 The *qhold* utility shall conform to XBD [Section 12.2](#) (on page 215).

103778 The following option shall be supported by the implementation:

103779 **-h *hold\_list*** Define the types of holds to be placed on the batch job.

103780 The *qhold* **-h** option shall accept a value for the *hold\_list* option-argument that is a  
 103781 string of alphanumeric characters in the portable character set (see XBD [Section](#)  
 103782 [6.1](#), on page 125).

103783 The *qhold* utility shall accept a value for the *hold\_list* option-argument that is a  
 103784 string of one or more of the characters 'u', 's', or 'o', or the single character  
 103785 'n'.

103786 For each unique character in the *hold\_list* option-argument, the *qhold* utility shall  
 103787 add a value to the *Hold\_Types* attribute of the batch job as follows, each  
 103788 representing a different hold type:

103789 u USER

103790 s SYSTEM

103791 o OPERATOR

103792 If any of these characters are duplicated in the *hold\_list* option-argument, the  
 103793 duplicates shall be ignored.

103794 An existing *Hold\_Types* attribute can be cleared by the following hold type:

103795 n NO\_HOLD

103796 The *qhold* utility shall consider it an error if any hold type other than 'n' is  
 103797 combined with hold type 'n'.

103798 Strictly conforming applications shall not repeat any of the characters 'u', 's',

103799 'o', or 'n' within the *hold\_list* option-argument. The *qhold* utility shall permit the  
 103800 repetition of characters, but shall not assign additional meaning to the repeated  
 103801 characters.

103802 An implementation may define other hold types. The conformance document for  
 103803 an implementation shall describe any additional hold types, how they are  
 103804 specified, their internal behavior, and how they affect the behavior of the utility.

103805 If the **-h** option is not presented to the *qhold* utility, the implementation shall set  
 103806 the *Hold\_Types* attribute to USER.

#### 103807 OPERANDS

103808 The *qhold* utility shall accept one or more operands that conform to the syntax for a batch  
 103809 *job\_identifier* (see [Section 3.3.1](#), on page 2419).

#### 103810 STDIN

103811 Not used.

#### 103812 INPUT FILES

103813 None.

#### 103814 ENVIRONMENT VARIABLES

103815 The following environment variables shall affect the execution of *qhold*:

103816 *LANG* Provide a default value for the internationalization variables that are unset or null.  
 103817 (See XBD [Section 8.2](#) (on page 174) the precedence of internationalization variables  
 103818 used to determine the values of locale categories.)

103819 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
 103820 internationalization variables.

103821 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
 103822 characters (for example, single-byte as opposed to multi-byte characters in  
 103823 arguments).

103824 *LC\_MESSAGES*

103825 Determine the locale that should be used to affect the format and contents of  
 103826 diagnostic messages written to standard error.

103827 *LOGNAME* Determine the login name of the user.

#### 103828 ASYNCHRONOUS EVENTS

103829 Default.

#### 103830 STDOUT

103831 None.

#### 103832 STDERR

103833 The standard error shall be used only for diagnostic messages.

#### 103834 OUTPUT FILES

103835 None.

#### 103836 EXTENDED DESCRIPTION

103837 None.

#### 103838 EXIT STATUS

103839 The following exit values shall be returned:

103840 0 Successful completion.

103841 >0 An error occurred.

#### 103842 CONSEQUENCES OF ERRORS

103843 In addition to the default behavior, the *qhold* utility shall not be required to write a diagnostic  
103844 message to standard error when the error reply received from a batch server indicates that the  
103845 batch *job\_identifier* does not exist on the server. Whether or not the *qhold* utility waits to output  
103846 the diagnostic message while attempting to locate the job on other servers is implementation-  
103847 defined.

#### 103848 APPLICATION USAGE

103849 None.

#### 103850 EXAMPLES

103851 None.

#### 103852 RATIONALE

103853 The *qhold* utility allows users to place a hold on one or more jobs. A hold makes a batch job  
103854 ineligible for execution.

103855 The *qhold* utility has options that allow the user to specify the type of hold. Should the user wish  
103856 to place a hold on a set of jobs that meet a selection criteria, such a list of jobs can be acquired  
103857 using the *qselect* utility.

103858 The *-h* option allows the user to specify the type of hold that is to be placed on the job. This  
103859 option allows for USER, SYSTEM, OPERATOR, and implementation-defined hold types. The  
103860 USER and OPERATOR holds are distinct. The batch server that manages the batch job will verify  
103861 that the user is authorized to set the specified hold for the batch job.

103862 Mail is not required on hold because the administrator has the tools and libraries to build this  
103863 option if he or she wishes.

103864 Historically, the *qhold* utility has been a part of some existing batch systems, although it has not  
103865 traditionally been a part of the NQS.

#### 103866 FUTURE DIRECTIONS

103867 The *qhold* utility may be removed in a future version.

#### 103868 SEE ALSO

103869 [Chapter 3](#) (on page 2397), *qselect*

103870 [XBD Section 6.1](#) (on page 125), [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 215)

#### 103871 CHANGE HISTORY

103872 Derived from IEEE Std 1003.2d-1994.

#### 103873 Issue 6

103874 The *LC\_TIME* and *TZ* entries are removed from the ENVIRONMENT VARIABLES section.

#### 103875 Issue 7

103876 The *qhold* utility is marked obsolescent.

103877 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

103878 **NAME**

103879 qmove — move batch jobs

103880 **SYNOPSIS**103881 OB BE `qmove destination job_identifier...`103882 **DESCRIPTION**

103883 To move a batch job is to remove the batch job from the batch queue in which it resides and  
 103884 instantiate the batch job in another batch queue. A batch job is moved by a request to the batch  
 103885 server that manages the batch job. The *qmove* utility is a user-accessible batch client that requests  
 103886 the movement of one or more batch jobs.

103887 The *qmove* utility shall move those batch jobs, and only those batch jobs, for which a batch  
 103888 *job\_identifier* is presented to the utility.

103889 The *qmove* utility shall move batch jobs in the order in which the corresponding batch  
 103890 *job\_identifiers* are presented to the utility.

103891 If the *qmove* utility fails to process a batch *job\_identifier* successfully, the utility shall proceed to  
 103892 process the remaining batch *job\_identifiers*, if any.

103893 The *qmove* utility shall move batch jobs by sending a *Move Job Request* to the batch server that  
 103894 manages each batch job. The *qmove* utility shall not exit before the batch jobs corresponding to all  
 103895 successfully processed batch *job\_identifiers* have been moved.

103896 **OPTIONS**

103897 None.

103898 **OPERANDS**

103899 The *qmove* utility shall accept one operand that conforms to the syntax for a destination (see  
 103900 [Section 3.3.2](#), on page 2420).

103901 The *qmove* utility shall accept one or more operands that conform to the syntax for a batch  
 103902 *job\_identifier* (see [Section 3.3.1](#), on page 2419).

103903 **STDIN**

103904 Not used.

103905 **INPUT FILES**

103906 None.

103907 **ENVIRONMENT VARIABLES**103908 The following environment variables shall affect the execution of *qmove*:

103909 *LANG* Provide a default value for the internationalization variables that are unset or null.  
 103910 (See XBD [Section 8.2](#) (on page 174) the precedence of internationalization variables  
 103911 used to determine the values of locale categories.)

103912 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
 103913 internationalization variables.

103914 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
 103915 characters (for example, single-byte as opposed to multi-byte characters in  
 103916 arguments).

103917 *LC\_MESSAGES*

103918 Determine the locale that should be used to affect the format and contents of  
 103919 diagnostic messages written to standard error.

- 103920 *LOGNAME* Determine the login name of the user.
- 103921 **ASYNCHRONOUS EVENTS**
- 103922 Default.
- 103923 **STDOUT**
- 103924 None.
- 103925 **STDERR**
- 103926 The standard error shall be used only for diagnostic messages.
- 103927 **OUTPUT FILES**
- 103928 None.
- 103929 **EXTENDED DESCRIPTION**
- 103930 None.
- 103931 **EXIT STATUS**
- 103932 The following exit values shall be returned:
- 103933 0 Successful completion.
- 103934 >0 An error occurred.
- 103935 **CONSEQUENCES OF ERRORS**
- 103936 In addition to the default behavior, the *qmove* utility shall not be required to write a diagnostic message to standard error when the error reply received from a batch server indicates that the batch *job\_identifier* does not exist on the server. Whether or not the *qmove* utility waits to output the diagnostic message while attempting to locate the job on other servers is implementation-defined.
- 103941 **APPLICATION USAGE**
- 103942 None.
- 103943 **EXAMPLES**
- 103944 None.
- 103945 **RATIONALE**
- 103946 The *qmove* utility allows users to move jobs between queues.
- 103947 The alternative to using the *qmove* utility—deleting the batch job and requeuing it—entails considerably more typing.
- 103949 Since the means of selecting jobs based on attributes has been encapsulated in the *qselect* utility, the only option of the *qmove* utility concerns authorization. The **-u** option provides the user with the convenience of changing the user identifier under which the batch job will execute.
- 103952 Minimalism and consistency have taken precedence over convenience; the **-u** option has been deleted because the equivalent capability exists with the **-u** option of the *qalter* utility.
- 103954 **FUTURE DIRECTIONS**
- 103955 The *qmove* utility may be removed in a future version.
- 103956 **SEE ALSO**
- 103957 [Chapter 3](#) (on page 2397), *qalter*, *qselect*
- 103958 [XBD Chapter 8](#) (on page 173)

103959 **CHANGE HISTORY**

103960 Derived from IEEE Std 1003.2d-1994.

103961 **Issue 6**

103962 The *LC\_TIME* and *TZ* entries are removed from the ENVIRONMENT VARIABLES section.

103963 **Issue 7**

103964 The *qmove* utility is marked obsolescent.

103965 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.



103966 **NAME**

103967       qmsg — send message to batch jobs

103968 **SYNOPSIS**103969 OB BE   qmsg [-EO] *message\_string job\_identifier...*103970 **DESCRIPTION**

103971 To send a message to a batch job is to request that a server write a message string into one or  
 103972 more output files of the batch job. A message is sent to a batch job by a request to the batch  
 103973 server that manages the batch job. The *qmsg* utility is a user-accessible batch client that requests  
 103974 the sending of messages to one or more batch jobs.

103975 The *qmsg* utility shall write messages into the files of batch jobs by sending a *Job Message Request*  
 103976 to the batch server that manages the batch job. The *qmsg* utility shall not directly write the  
 103977 message into the files of the batch job.

103978 The *qmsg* utility shall send a *Job Message Request* for those batch jobs, and only those batch jobs,  
 103979 for which a batch *job\_identifier* is presented to the utility.

103980 The *qmsg* utility shall send *Job Message Requests* for batch jobs in the order in which their batch  
 103981 *job\_identifiers* are presented to the utility.

103982 If the *qmsg* utility fails to process any batch *job\_identifier* successfully, the utility shall proceed to  
 103983 process the remaining batch *job\_identifiers*, if any.

103984 The *qmsg* utility shall not exit before a *Job Message Request* has been sent to the server that  
 103985 manages the batch job that corresponds to each successfully processed batch *job\_identifier*.

103986 **OPTIONS**

103987 The *qmsg* utility shall conform to XBD [Section 12.2](#) (on page 215).

103988 The following options shall be supported by the implementation:

103989    -E       Specify that the message is written to the standard error of each batch job.

103990           The *qmsg* utility shall write the message into the standard error of the batch job.

103991    -O       Specify that the message is written to the standard output of each batch job.

103992           The *qmsg* utility shall write the message into the standard output of the batch job.

103993 If neither the -O nor the -E option is presented to the *qmsg* utility, the utility shall write the  
 103994 message into an implementation-defined file. The conformance document for the  
 103995 implementation shall describe the name and location of the implementation-defined file. If both  
 103996 the -O and the -E options are presented to the *qmsg* utility, then the utility shall write the  
 103997 messages to both standard output and standard error.

103998 **OPERANDS**

103999 The *qmsg* utility shall accept a minimum of two operands, *message\_string* and one or more batch  
 104000 *job\_identifiers*.

104001 The *message\_string* operand shall be the string to be written to one or more output files of the  
 104002 batch job followed by a <newline>. If the string contains <blank> characters, then the  
 104003 application shall ensure that the string is quoted. The *message\_string* shall be encoded in the  
 104004 portable character set (see XBD [Section 6.1](#), on page 125).

104005 All remaining operands are batch *job\_identifiers* that conform to the syntax for a batch  
 104006 *job\_identifier* (see [Section 3.3.1](#), on page 2419).

104007 **STDIN**

104008 Not used.

104009 **INPUT FILES**

104010 None.

104011 **ENVIRONMENT VARIABLES**104012 The following environment variables shall affect the execution of *qmsg*:

104013 *LANG* Provide a default value for the internationalization variables that are unset or null.  
 104014 (See XBD [Section 8.2](#) (on page 174) the precedence of internationalization variables  
 104015 used to determine the values of locale categories.)

104016 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
 104017 internationalization variables.

104018 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
 104019 characters (for example, single-byte as opposed to multi-byte characters in  
 104020 arguments).

104021 *LC\_MESSAGES*

104022 Determine the locale that should be used to affect the format and contents of  
 104023 diagnostic messages written to standard error.

104024 *LOGNAME* Determine the login name of the user.

104025 **ASYNCHRONOUS EVENTS**

104026 Default.

104027 **STDOUT**

104028 None.

104029 **STDERR**

104030 The standard error shall be used only for diagnostic messages.

104031 **OUTPUT FILES**

104032 None.

104033 **EXTENDED DESCRIPTION**

104034 None.

104035 **EXIT STATUS**

104036 The following exit values shall be returned:

104037 0 Successful completion.

104038 &gt;0 An error occurred.

104039 **CONSEQUENCES OF ERRORS**

104040 In addition to the default behavior, the *qmsg* utility shall not be required to write a diagnostic  
 104041 message to standard error when the error reply received from a batch server indicates that the  
 104042 batch *job\_identifier* does not exist on the server. Whether or not the *qmsg* utility waits to output  
 104043 the diagnostic message while attempting to locate the job on other servers is implementation-  
 104044 defined.

**104045 APPLICATION USAGE**

104046 None.

**104047 EXAMPLES**

104048 None.

**104049 RATIONALE**

104050 The *qmsg* utility allows users to write messages into the output files of running jobs. Users,  
104051 including operators and administrators, have a number of occasions when they want to place  
104052 messages in the output files of a batch job. For example, if a disk that is being used by a batch job  
104053 is showing errors, the operator might note this in the standard error stream of the batch job.

104054 The options of the *qmsg* utility provide users with the means of placing the message in the  
104055 output stream of their choice. The default output stream for the message—if the user does not  
104056 designate an output stream—is implementation-defined, since many implementations will  
104057 provide, as an extension to this volume of POSIX.1-2008, a log file that shows the history of  
104058 utility execution.

104059 If users wish to send a message to a set of jobs that meet a selection criteria, the *qselect* utility can  
104060 be used to acquire the appropriate list of job identifiers.

104061 The **-E** option allows users to place the message in the standard error stream of the batch job.

104062 The **-O** option allows users to place the message in the standard output stream of the batch job.

104063 Historically, the *qmsg* utility is an existing practice in the offerings of one or more implementors  
104064 of an NQS-derived batch system. The utility has been found to be useful enough that it deserves  
104065 to be included in this volume of POSIX.1-2008.

**104066 FUTURE DIRECTIONS**

104067 The *qmsg* utility may be removed in a future version.

**104068 SEE ALSO**

104069 [Chapter 3](#) (on page 2397), *qselect*

104070 XBD [Section 6.1](#) (on page 125), [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 215)

**104071 CHANGE HISTORY**

104072 Derived from IEEE Std 1003.2d-1994.

**104073 Issue 6**

104074 The *LC\_TIME* and *TZ* entries are removed from the ENVIRONMENT VARIABLES section.

**104075 Issue 7**

104076 The *qmsg* utility is marked obsolescent.

104077 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

104078 **NAME**

104079        qrerun — rerun batch jobs

104080 **SYNOPSIS**104081 OB BE    `qrerun job_identifier...`104082 **DESCRIPTION**

104083        To rerun a batch job is to terminate the session leader of the batch job, delete any associated  
 104084        checkpoint files, and return the batch job to the batch queued state. A batch job is rerun by a  
 104085        request to the batch server that manages the batch job. The *qrerun* utility is a user-accessible  
 104086        batch client that requests the rerunning of one or more batch jobs.

104087        The *qrerun* utility shall rerun those batch jobs for which a batch *job\_identifier* is presented to the  
 104088        utility.

104089        The *qrerun* utility shall rerun batch jobs in the order in which their batch *job\_identifiers* are  
 104090        presented to the utility.

104091        If the *qrerun* utility fails to process any batch *job\_identifier* successfully, the utility shall proceed to  
 104092        process the remaining batch *job\_identifiers*, if any.

104093        The *qrerun* utility shall rerun batch jobs by sending a *Rerun Job Request* to the batch server that  
 104094        manages each batch job.

104095        For each successfully processed batch *job\_identifier*, the *qrerun* utility shall have rerun the  
 104096        corresponding batch job at the time the utility exits.

104097 **OPTIONS**

104098        None.

104099 **OPERANDS**

104100        The *qrerun* utility shall accept one or more operands that conform to the syntax for a batch  
 104101        *job\_identifier* (see [Section 3.3.1](#), on page 2419).

104102 **STDIN**

104103        Not used.

104104 **INPUT FILES**

104105        None.

104106 **ENVIRONMENT VARIABLES**104107        The following environment variables shall affect the execution of *qrerun*:

104108        *LANG*        Provide a default value for the internationalization variables that are unset or null.  
 104109        (See [XBD Section 8.2](#) (on page 174) the precedence of internationalization variables  
 104110        used to determine the values of locale categories.)

104111        *LC\_ALL*       If set to a non-empty string value, override the values of all the other  
 104112        internationalization variables.

104113        *LC\_CTYPE*    Determine the locale for the interpretation of sequences of bytes of text data as  
 104114        characters (for example, single-byte as opposed to multi-byte characters in  
 104115        arguments).

104116        *LC\_MESSAGES*

104117        Determine the locale that should be used to affect the format and contents of  
 104118        diagnostic messages written to standard error.

- 104119 *LOGNAME* Determine the login name of the user.
- 104120 **ASYNCHRONOUS EVENTS**
- 104121 Default.
- 104122 **STDOUT**
- 104123 None.
- 104124 **STDERR**
- 104125 The standard error shall be used only for diagnostic messages.
- 104126 **OUTPUT FILES**
- 104127 None.
- 104128 **EXTENDED DESCRIPTION**
- 104129 None.
- 104130 **EXIT STATUS**
- 104131 The following exit values shall be returned:
- 104132 0 Successful completion.
- 104133 >0 An error occurred.
- 104134 **CONSEQUENCES OF ERRORS**
- 104135 In addition to the default behavior, the *qrerun* utility shall not be required to write a diagnostic message to standard error when the error reply received from a batch server indicates that the batch *job\_identifier* does not exist on the server. Whether or not the *qrerun* utility waits to output the diagnostic message while attempting to locate the job on other servers is implementation-defined.
- 104140 **APPLICATION USAGE**
- 104141 None.
- 104142 **EXAMPLES**
- 104143 None.
- 104144 **RATIONALE**
- 104145 The *qrerun* utility allows users to cause jobs in the running state to exit and rerun.
- 104146 The *qrerun* utility is a new utility, *vis-a-vis* existing practice, that has been defined in this volume of POSIX.1-2008 to correct user-perceived deficiencies in the existing practice.
- 104147
- 104148 **FUTURE DIRECTIONS**
- 104149 The *qrerun* utility may be removed in a future version.
- 104150 **SEE ALSO**
- 104151 [Chapter 3](#) (on page 2397)
- 104152 XBD [Chapter 8](#) (on page 173)
- 104153 **CHANGE HISTORY**
- 104154 Derived from IEEE Std 1003.2d-1994.
- 104155 **Issue 6**
- 104156 The *LC\_TIME* and *TZ* entries are removed from the ENVIRONMENT VARIABLES section.

104157 **Issue 7**

104158 The *qrerun* utility is marked obsolescent.

104159 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

104160 **NAME**

104161 qrln — release batch jobs

104162 **SYNOPSIS**104163 OB BE `qrln [-h hold_list] job_identifier...`104164 **DESCRIPTION**

104165 A batch job might have one or more holds, which prevent the batch job from executing. A batch  
 104166 job from which all the holds have been removed becomes eligible for execution and is said to  
 104167 have been released. A batch job hold is removed by sending a request to the batch server that  
 104168 manages the batch job. The *qrln* utility is a user-accessible client of batch services that requests  
 104169 holds be removed from one or more batch jobs.

104170 The *qrln* utility shall remove one or more holds from those batch jobs for which a batch  
 104171 *job\_identifier* is presented to the utility.

104172 The *qrln* utility shall remove holds from batch jobs in the order in which their batch *job\_identifiers*  
 104173 are presented to the utility.

104174 If the *qrln* utility fails to process a batch *job\_identifier* successfully, the utility shall proceed to  
 104175 process the remaining batch *job\_identifiers*, if any.

104176 The *qrln* utility shall remove holds on each batch job by sending a *Release Job Request* to the batch  
 104177 server that manages the batch job.

104178 The *qrln* utility shall not exit until the holds have been removed from the batch job  
 104179 corresponding to each successfully processed batch *job\_identifier*.

104180 **OPTIONS**

104181 The *qrln* utility shall conform to XBD [Section 12.2](#) (on page 215).

104182 The following option shall be supported by the implementation:

104183 **-h *hold\_list*** Define the types of holds to be removed from the batch job.

104184 The *qrln* **-h** option shall accept a value for the *hold\_list* option-argument that is a  
 104185 string of alphanumeric characters in the portable character set (see XBD [Section](#)  
 104186 [6.1](#), on page 125).

104187 The *qrln* utility shall accept a value for the *hold\_list* option-argument that is a string  
 104188 of one or more of the characters 'u', 's', or 'o', or the single character 'n'.

104189 For each unique character in the *hold\_list* option-argument, the *qrln* utility shall add  
 104190 a value to the *Hold\_Types* attribute of the batch job as follows, each representing a  
 104191 different hold type:

104192 u USER

104193 s SYSTEM

104194 o OPERATOR

104195 If any of these characters are duplicated in the *hold\_list* option-argument, the  
 104196 duplicates shall be ignored.

104197 An existing *Hold\_Types* attribute can be cleared by the following hold type:

104198 n NO\_HOLD

104199 The *qrln* utility shall consider it an error if any hold type other than 'n' is  
 104200 combined with hold type 'n'.

104201 Strictly conforming applications shall not repeat any of the characters 'u', 's',  
 104202 'o', or 'n' within the *hold\_list* option-argument. The *qrls* utility shall permit the  
 104203 repetition of characters, but shall not assign additional meaning to the repeated  
 104204 characters.

104205 An implementation may define other hold types. The conformance document for  
 104206 an implementation shall describe any additional hold types, how they are  
 104207 specified, their internal behavior, and how they affect the behavior of the utility.

104208 If the **-h** option is not presented to the *qrls* utility, the implementation shall remove  
 104209 the USER hold in the *Hold\_Types* attribute.

#### 104210 OPERANDS

104211 The *qrls* utility shall accept one or more operands that conform to the syntax for a batch  
 104212 *job\_identifier* (see [Section 3.3.1](#), on page 2419).

#### 104213 STDIN

104214 Not used.

#### 104215 INPUT FILES

104216 None.

#### 104217 ENVIRONMENT VARIABLES

104218 The following environment variables shall affect the execution of *qrls*:

104219 *LANG* Provide a default value for the internationalization variables that are unset or null.  
 104220 (See XBD [Section 8.2](#) (on page 174) the precedence of internationalization variables  
 104221 used to determine the values of locale categories.)

104222 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
 104223 internationalization variables.

104224 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
 104225 characters (for example, single-byte as opposed to multi-byte characters in  
 104226 arguments).

104227 *LC\_MESSAGES*

104228 Determine the locale that should be used to affect the format and contents of  
 104229 diagnostic messages written to standard error.

104230 *LOGNAME* Determine the login name of the user.

#### 104231 ASYNCHRONOUS EVENTS

104232 Default.

#### 104233 STDOUT

104234 None.

#### 104235 STDERR

104236 The standard error shall be used only for diagnostic messages.

#### 104237 OUTPUT FILES

104238 None.

#### 104239 EXTENDED DESCRIPTION

104240 None.



**104241 EXIT STATUS**

104242 The following exit values shall be returned:

104243 0 Successful completion.

104244 >0 An error occurred.

**104245 CONSEQUENCES OF ERRORS**

104246 In addition to the default behavior, the *qrsl* utility shall not be required to write a diagnostic  
104247 message to standard error when the error reply received from a batch server indicates that the  
104248 batch *job\_identifier* does not exist on the server. Whether or not the *qrsl* utility waits to output the  
104249 diagnostic message while attempting to locate the job on other servers is implementation-  
104250 defined.

**104251 APPLICATION USAGE**

104252 None.

**104253 EXAMPLES**

104254 None.

**104255 RATIONALE**

104256 The *qrsl* utility allows users, operators, and administrators to remove holds from jobs.

104257 The *qrsl* utility does not support any job selection options or wildcard arguments. Users may  
104258 acquire a list of jobs selected by attributes using the *qselect* utility. For example, a user could  
104259 select all of their held jobs.

104260 The **-h** option allows the user to specify the type of hold that is to be removed. This option  
104261 allows for USER, SYSTEM, OPERATOR, and implementation-defined hold types. The batch  
104262 server that manages the batch job will verify whether the user is authorized to remove the  
104263 specified hold for the batch job. If more than one type of hold has been placed on the batch job, a  
104264 user may wish to remove only some of them.

104265 Mail is not required on release because the administrator has the tools and libraries to build this  
104266 option if required.

104267 The *qrsl* utility is a new utility *vis-a-vis* existing practice; it has been defined in this volume of  
104268 POSIX.1-2008 as the natural complement to the *qhold* utility.

**104269 FUTURE DIRECTIONS**

104270 The *qrsl* utility may be removed in a future version.

**104271 SEE ALSO**

104272 [Chapter 3](#) (on page 2397), *qhold*, *qselect*

104273 [XBD Section 6.1](#) (on page 125), [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 215)

**104274 CHANGE HISTORY**

104275 Derived from IEEE Std 1003.2d-1994.

**104276 Issue 6**

104277 The *LC\_TIME* and *TZ* entries are removed from the ENVIRONMENT VARIABLES section.

**104278 Issue 7**

104279 The *qrsl* utility is marked obsolescent.

104280 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

104281 **NAME**

104282 qselect — select batch jobs

104283 **SYNOPSIS**

```
104284 OB BE qselect [-a [op]date_time] [-A account_string] [-c [op]interval]
104285 [-h hold_list] [-l resource_list] [-N name] [-p [op]priority]
104286 [-q destination] [-r y|n] [-s states] [-u user_list]
```

104287 **DESCRIPTION**

104288 To select a set of batch jobs is to return the batch *job\_identifiers* for each batch job that meets a list  
 104289 of selection criteria. A set of batch jobs is selected by a request to a batch server. The *qselect* utility  
 104290 is a user-accessible batch client that requests the selection of batch jobs.

104291 Upon successful completion, the *qselect* utility shall have returned a list of zero or more batch  
 104292 *job\_identifiers* that meet the criteria specified by the options and option-arguments presented to  
 104293 the utility.

104294 The *qselect* utility shall select batch jobs by sending a *Select Jobs Request* to a batch server. The  
 104295 *qselect* utility shall not exit until the server replies to each request generated.

104296 For each option presented to the *qselect* utility, the utility shall restrict the set of selected batch  
 104297 jobs as described in the OPTIONS section.

104298 The *qselect* utility shall not restrict selection of batch jobs except by authorization and as required  
 104299 by the options presented to the utility.

104300 When an option is specified with a mandatory or optional *op* component to the option-  
 104301 argument, then *op* shall specify a relation between the value of a certain batch job attribute and  
 104302 the *value* component of the option-argument. If an *op* is allowable on an option, then the  
 104303 description of the option letter indicates the *op* as either mandatory or optional. Acceptable  
 104304 strings for the *op* component, and the relation the string indicates, are shown in the following  
 104305 list:

- 104306 .eq. The value represented by the attribute of the batch job is equal to the value represented
- 104307 by the option-argument.
- 104308 .ge. The value represented by the attribute of the batch job is greater than or equal to the
- 104309 value represented by the option-argument.
- 104310 .gt. The value represented by the attribute of the batch job is greater than the value
- 104311 represented by the option-argument.
- 104312 .lt. The value represented by the attribute of the batch job is less than the value represented
- 104313 by the option-argument.
- 104314 .le. The value represented by the attribute of the batch job is less than or equal to the value
- 104315 represented by the option-argument.
- 104316 .ne. The value represented by the attribute of the batch job is not equal to the value
- 104317 represented by the option-argument.

104318 **OPTIONS**

104319 The *qselect* utility shall conform to XBD [Section 12.2](#) (on page 215).

104320 The following options shall be supported by the implementation:

104321 **-a [op]date\_time**

104322 Restrict selection to a specific time, or a range of times.

104323 The *qselect* utility shall select only batch jobs for which the value of the

104324 *Execution\_Time* attribute is related to the Epoch equivalent of the local time  
 104325 expressed by the value of the *date\_time* component of the option-argument in the  
 104326 manner indicated by the value of the *op* component of the option-argument.

104327 The *qselect* utility shall accept a *date\_time* component of the option-argument that  
 104328 conforms to the syntax of the *time* operand of the *touch* utility.

104329 If the *op* component of the option-argument is not presented to the *qselect* utility,  
 104330 the utility shall select batch jobs for which the *Execution\_Time* attribute is equal to  
 104331 the *date\_time* component of the option-argument.

104332 When comparing times, the *qselect* utility shall use the following definitions for the  
 104333 *op* component of the option-argument:

104334 .eq. The time represented by value of the *Execution\_Time* attribute of the batch  
 104335 job is equal to the time represented by the *date\_time* component of the  
 104336 option-argument.

104337 .ge. The time represented by value of the *Execution\_Time* attribute of the batch  
 104338 job is after or equal to the time represented by the *date\_time* component of  
 104339 the option-argument.

104340 .gt. The time represented by value of the *Execution\_Time* attribute of the batch  
 104341 job is after the time represented by the *date\_time* component of the option-  
 104342 argument.

104343 .lt. The time represented by value of the *Execution\_Time* attribute of the batch  
 104344 job is before the time represented by the *date\_time* component of the  
 104345 option-argument.

104346 .le. The time represented by value of the *Execution\_Time* attribute of the batch  
 104347 job is before or equal to the time represented by the *date\_time* component  
 104348 of the option-argument.

104349 .ne. The time represented by value of the *Execution\_Time* attribute of the batch  
 104350 job is not equal to the time represented by the *date\_time* component of the  
 104351 option-argument.

104352 The *qselect* utility shall accept the defined character strings for the *op* component of  
 104353 the option-argument.

104354 **-A** *account\_string*  
 104355 Restrict selection to the batch jobs charging a specified account.

104356 The *qselect* utility shall select only batch jobs for which the value of the  
 104357 *Account\_Name* attribute of the batch job matches the value of the *account\_string*  
 104358 option-argument.

104359 The syntax of the *account\_string* option-argument is unspecified.

104360 **-c** [*op*]*interval*  
 104361 Restrict selection to batch jobs within a range of checkpoint intervals.

104362 The *qselect* utility shall select only batch jobs for which the value of the *Checkpoint*  
 104363 attribute relates to the value of the *interval* component of the option-argument in  
 104364 the manner indicated by the value of the *op* component of the option-argument.

104365 If the *op* component of the option-argument is omitted, the *qselect* utility shall select  
 104366 batch jobs for which the value of the *Checkpoint* attribute is equal to the value of the  
 104367 *interval* component of the option-argument.

104368 When comparing checkpoint intervals, the *qselect* utility shall use the following  
 104369 definitions for the *op* component of the option-argument:

104370 .eq. The value of the *Checkpoint* attribute of the batch job equals the value of  
 104371 the *interval* component of the option-argument.

104372 .ge. The value of the *Checkpoint* attribute of the batch job is greater than or  
 104373 equal to the value of the *interval* component option-argument.

104374 .gt. The value of the *Checkpoint* attribute of the batch job is greater than the  
 104375 value of the *interval* component option-argument.

104376 .lt. The value of the *Checkpoint* attribute of the batch job is less than the value  
 104377 of the *interval* component option-argument.

104378 .le. The value of the *Checkpoint* attribute of the batch job is less than or equal  
 104379 to the value of the *interval* component option-argument.

104380 .ne. The value of the *Checkpoint* attribute of the batch job does not equal the  
 104381 value of the *interval* component option-argument.

104382 The *qselect* utility shall accept the defined character strings for the *op* component of  
 104383 the option-argument.

104384 The ordering relationship for the values of the interval option-argument is defined  
 104385 to be:

104386 'n' .gt. 's' .gt. 'c=minutes' .ge. 'c'

104387 When comparing *Checkpoint* attributes with an interval having the value of the  
 104388 single character 'u', only equality or inequality are valid comparisons.

104389 **-h hold\_list** Restrict selection to batch jobs that have a specific type of hold.

104390 The *qselect* utility shall select only batch jobs for which the value of the *Hold\_Types*  
 104391 attribute matches the value of the *hold\_list* option-argument.

104392 The *qselect* **-h** option shall accept a value for the *hold\_list* option-argument that is a  
 104393 string of alphanumeric characters in the portable character set (see XBD [Section 6.1](#), on page 125).  
 104394

104395 The *qselect* utility shall accept a value for the *hold\_list* option-argument that is a  
 104396 string of one or more of the characters 'u', 's', or 'o', or the single character  
 104397 'n'.

104398 Each unique character in the *hold\_list* option-argument of the *qselect* utility is  
 104399 defined as follows, each representing a different hold type:

104400 u USER  
 104401 s SYSTEM  
 104402 o OPERATOR

104403 If any of these characters are duplicated in the *hold\_list* option-argument, the  
 104404 duplicates shall be ignored.

104405 The *qselect* utility shall consider it an error if any hold type other than 'n' is  
 104406 combined with hold type 'n'.

104407 Strictly conforming applications shall not repeat any of the characters 'u', 's',  
 104408 'o', or 'n' within the *hold\_list* option-argument. The *qselect* utility shall permit

104409 the repetition of characters, but shall not assign additional meaning to the repeated  
 104410 characters.

104411 An implementation may define other hold types. The conformance document for  
 104412 an implementation shall describe any additional hold types, how they are  
 104413 specified, their internal behavior, and how they affect the behavior of the utility.

104414 **-I *resource\_list***  
 104415 Restrict selection to batch jobs with specified resource limits and attributes.

104416 The *qselect* utility shall accept a *resource\_list* option-argument with the following  
 104417 syntax:

104418 *resource\_name op value [, , resource\_name op value, , ...]*

104419 When comparing resource values, the *qselect* utility shall use the following  
 104420 definitions for the *op* component of the option-argument:

104421 *.eq.* The value of the resource of the same name in the *Resource\_List* attribute  
 104422 of the batch job equals the value of the value component of the option-  
 104423 argument.

104424 *.ge.* The value of the resource of the same name in the *Resource\_List* attribute  
 104425 of the batch job is greater than or equal to the value of the *value*  
 104426 component of the option-argument.

104427 *.gt.* The value of the resource of the same name in the *Resource\_List* attribute  
 104428 of the batch job is greater than the value of the value component of the  
 104429 option-argument.

104430 *.lt.* The value of the resource of the same name in the *Resource\_List* attribute  
 104431 of the batch job is less than the value of the value component of the  
 104432 option-argument.

104433 *.ne.* The value of the resource of the same name in the *Resource\_List* attribute  
 104434 of the batch job does not equal the value of the value component of the  
 104435 option-argument.

104436 *.le.* The value of the resource of the same name in the *Resource\_List* attribute  
 104437 of the batch job is less than or equal to the value of the *value* component of  
 104438 the option-argument.

104439 When comparing the limit of a *Resource\_List* attribute with the *value* component of  
 104440 the option-argument, if the limit, the value, or both are non-numeric, only equality  
 104441 or inequality are valid comparisons.

104442 The *qselect* utility shall select only batch jobs for which the values of the  
 104443 *resource\_names* listed in the *resource\_list* option-argument match the corresponding  
 104444 limits of the *Resource\_List* attribute of the batch job.

104445 Limits of *resource\_names* present in the *Resource\_List* attribute of the batch job that  
 104446 have no corresponding values in the *resource\_list* option-argument shall not be  
 104447 considered when selecting batch jobs.

104448 **-N *name*** Restrict selection to batch jobs with a specified name.

104449 The *qselect* utility shall select only batch jobs for which the value of the *Job\_Name*  
 104450 attribute matches the value of the *name* option-argument. The string specified in  
 104451 the *name* option-argument shall be passed, uninterpreted, to the server. This allows  
 104452 an implementation to match “wildcard” patterns against batch job names.

|        |                                         |                                                                                                                                                                                                                                                                                                    |
|--------|-----------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 104453 |                                         | An implementation shall describe in the conformance document the format it supports for matching against the <i>Job_Name</i> attribute.                                                                                                                                                            |
| 104454 |                                         |                                                                                                                                                                                                                                                                                                    |
| 104455 | <b>-p</b> [ <i>op</i> ] <i>priority</i> |                                                                                                                                                                                                                                                                                                    |
| 104456 |                                         | Restrict selection to batch jobs of the specified priority or range of priorities.                                                                                                                                                                                                                 |
| 104457 |                                         | The <i>qselect</i> utility shall select only batch jobs for which the value of the <i>Priority</i> attribute of the batch job relates to the value of the <i>priority</i> component of the option-argument in the manner indicated by the value of the <i>op</i> component of the option-argument. |
| 104458 |                                         |                                                                                                                                                                                                                                                                                                    |
| 104459 |                                         |                                                                                                                                                                                                                                                                                                    |
| 104460 |                                         |                                                                                                                                                                                                                                                                                                    |
| 104461 |                                         | If the <i>op</i> component of the option-argument is omitted, the <i>qselect</i> utility shall select batch jobs for which the value of the <i>Priority</i> attribute of the batch job is equal to the value of the <i>priority</i> component of the option-argument.                              |
| 104462 |                                         |                                                                                                                                                                                                                                                                                                    |
| 104463 |                                         |                                                                                                                                                                                                                                                                                                    |
| 104464 |                                         | When comparing priority values, the <i>qselect</i> utility shall use the following definitions for the <i>op</i> component of the option-argument:                                                                                                                                                 |
| 104465 |                                         |                                                                                                                                                                                                                                                                                                    |
| 104466 | .eq.                                    | The value of the <i>Priority</i> attribute of the batch job equals the value of the <i>priority</i> component of the option-argument.                                                                                                                                                              |
| 104467 |                                         |                                                                                                                                                                                                                                                                                                    |
| 104468 | .ge.                                    | The value of the <i>Priority</i> attribute of the batch job is greater than or equal to the value of the <i>priority</i> component option-argument.                                                                                                                                                |
| 104469 |                                         |                                                                                                                                                                                                                                                                                                    |
| 104470 | .gt.                                    | The value of the <i>Priority</i> attribute of the batch job is greater than the value of the <i>priority</i> component option-argument.                                                                                                                                                            |
| 104471 |                                         |                                                                                                                                                                                                                                                                                                    |
| 104472 | .lt.                                    | The value of the <i>Priority</i> attribute of the batch job is less than the value of the <i>priority</i> component option-argument.                                                                                                                                                               |
| 104473 |                                         |                                                                                                                                                                                                                                                                                                    |
| 104474 | .lte.                                   | The value of the <i>Priority</i> attribute of the batch job is less than or equal to the value of the <i>priority</i> component option-argument.                                                                                                                                                   |
| 104475 |                                         |                                                                                                                                                                                                                                                                                                    |
| 104476 | .ne.                                    | The value of the <i>Priority</i> attribute of the batch job does not equal the value of the <i>priority</i> component option-argument.                                                                                                                                                             |
| 104477 |                                         |                                                                                                                                                                                                                                                                                                    |
| 104478 | <b>-q</b> <i>destination</i>            |                                                                                                                                                                                                                                                                                                    |
| 104479 |                                         | Restrict selection to the specified batch queue or server, or both.                                                                                                                                                                                                                                |
| 104480 |                                         | The <i>qselect</i> utility shall select only batch jobs that are located at the destination indicated by the value of the <i>destination</i> option-argument.                                                                                                                                      |
| 104481 |                                         |                                                                                                                                                                                                                                                                                                    |
| 104482 |                                         | The destination defines a batch queue, a server, or a batch queue at a server.                                                                                                                                                                                                                     |
| 104483 |                                         | The <i>qselect</i> utility shall accept an option-argument for the <b>-q</b> option that conforms to the syntax for a destination. If the <b>-q</b> option is not presented to the <i>qselect</i> utility, the utility shall select batch jobs from all batch queues at the default batch server.  |
| 104484 |                                         |                                                                                                                                                                                                                                                                                                    |
| 104485 |                                         |                                                                                                                                                                                                                                                                                                    |
| 104486 |                                         | If the option-argument describes only a batch queue, the <i>qselect</i> utility shall select only batch jobs from the batch queue of the specified name at the default batch server. The means by which <i>qselect</i> determines the default server is implementation-defined.                    |
| 104487 |                                         |                                                                                                                                                                                                                                                                                                    |
| 104488 |                                         |                                                                                                                                                                                                                                                                                                    |
| 104489 |                                         |                                                                                                                                                                                                                                                                                                    |
| 104490 |                                         | If the option-argument describes only a batch server, the <i>qselect</i> utility shall select batch jobs from all the batch queues at that batch server.                                                                                                                                           |
| 104491 |                                         |                                                                                                                                                                                                                                                                                                    |
| 104492 |                                         | If the option-argument describes both a batch queue and a batch server, the <i>qselect</i> utility shall select only batch jobs from the specified batch queue at the specified server.                                                                                                            |
| 104493 |                                         |                                                                                                                                                                                                                                                                                                    |
| 104494 |                                         |                                                                                                                                                                                                                                                                                                    |

|        |                              |                                                                                                                                                                                                                                                      |
|--------|------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 104495 | <b>-r y   n</b>              | Restrict selection to batch jobs with the specified rerunability status.                                                                                                                                                                             |
| 104496 |                              | The <i>qselect</i> utility shall select only batch jobs for which the value of the <i>Rerunable</i> attribute of the batch job matches the value of the option-argument.                                                                             |
| 104497 |                              |                                                                                                                                                                                                                                                      |
| 104498 |                              | The <i>qselect</i> utility shall accept a value for the option-argument that consists of either the single character 'y' or the single character 'n'. The character 'y' represents the value TRUE, and the character 'n' represents the value FALSE. |
| 104499 |                              |                                                                                                                                                                                                                                                      |
| 104500 |                              |                                                                                                                                                                                                                                                      |
| 104501 | <b>-s states</b>             | Restrict selection to batch jobs in the specified states.                                                                                                                                                                                            |
| 104502 |                              | The <i>qselect</i> utility shall accept an option-argument that consists of any combination of the characters 'e', 'q', 'r', 'w', 'h', and 't'.                                                                                                      |
| 104503 |                              |                                                                                                                                                                                                                                                      |
| 104504 |                              | Conforming applications shall not repeat any character in the option-argument.                                                                                                                                                                       |
| 104505 |                              | The <i>qselect</i> utility shall permit the repetition of characters in the option-argument, but shall not assign additional meaning to repeated characters.                                                                                         |
| 104506 |                              |                                                                                                                                                                                                                                                      |
| 104507 |                              | The <i>qselect</i> utility shall interpret the characters in the <i>states</i> option-argument as follows:                                                                                                                                           |
| 104508 |                              |                                                                                                                                                                                                                                                      |
| 104509 |                              | e Represents the EXITING state.                                                                                                                                                                                                                      |
| 104510 |                              | q Represents the QUEUED state.                                                                                                                                                                                                                       |
| 104511 |                              | r Represents the RUNNING state.                                                                                                                                                                                                                      |
| 104512 |                              | t Represents the TRANSITING state.                                                                                                                                                                                                                   |
| 104513 |                              | h Represents the HELD state.                                                                                                                                                                                                                         |
| 104514 |                              | w Represents the WAITING state.                                                                                                                                                                                                                      |
| 104515 |                              | For each character in the <i>states</i> option-argument, the <i>qselect</i> utility shall select batch jobs in the corresponding state.                                                                                                              |
| 104516 |                              |                                                                                                                                                                                                                                                      |
| 104517 | <b>-u user_list</b>          | Restrict selection to batch jobs owned by the specified user names.                                                                                                                                                                                  |
| 104518 |                              | The <i>qselect</i> utility shall select only the batch jobs of those users specified in the <i>user_list</i> option-argument.                                                                                                                        |
| 104519 |                              |                                                                                                                                                                                                                                                      |
| 104520 |                              | The <i>qselect</i> utility shall accept a <i>user_list</i> option-argument that conforms to the following syntax:                                                                                                                                    |
| 104521 |                              |                                                                                                                                                                                                                                                      |
| 104522 |                              | <i>username</i> [@ <i>host</i> ] [, , <i>username</i> [@ <i>host</i> ] , , ...]                                                                                                                                                                      |
| 104523 |                              | The <i>qselect</i> utility shall accept only one user name that is missing a corresponding host name. The <i>qselect</i> utility shall accept only one user name per named host.                                                                     |
| 104524 |                              |                                                                                                                                                                                                                                                      |
| 104525 | <b>OPERANDS</b>              |                                                                                                                                                                                                                                                      |
| 104526 |                              | None.                                                                                                                                                                                                                                                |
| 104527 | <b>STDIN</b>                 |                                                                                                                                                                                                                                                      |
| 104528 |                              | Not used.                                                                                                                                                                                                                                            |
| 104529 | <b>INPUT FILES</b>           |                                                                                                                                                                                                                                                      |
| 104530 |                              | None.                                                                                                                                                                                                                                                |
| 104531 | <b>ENVIRONMENT VARIABLES</b> |                                                                                                                                                                                                                                                      |
| 104532 |                              | The following environment variables shall affect the execution of <i>qselect</i> :                                                                                                                                                                   |

|        |                               |                                                                                                                                                                                                                                                |
|--------|-------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 104533 | <i>LANG</i>                   | Provide a default value for the internationalization variables that are unset or null. (See XBD <a href="#">Section 8.2</a> (on page 174) the precedence of internationalization variables used to determine the values of locale categories.) |
| 104534 |                               |                                                                                                                                                                                                                                                |
| 104535 |                               |                                                                                                                                                                                                                                                |
| 104536 | <i>LC_ALL</i>                 | If set to a non-empty string value, override the values of all the other internationalization variables.                                                                                                                                       |
| 104537 |                               |                                                                                                                                                                                                                                                |
| 104538 | <i>LC_CTYPE</i>               | Determine the locale for the interpretation of sequences of bytes of text data as characters (for example, single-byte as opposed to multi-byte characters in arguments).                                                                      |
| 104539 |                               |                                                                                                                                                                                                                                                |
| 104540 |                               |                                                                                                                                                                                                                                                |
| 104541 | <i>LC_MESSAGES</i>            |                                                                                                                                                                                                                                                |
| 104542 |                               | Determine the locale that should be used to affect the format and contents of diagnostic messages written to standard error.                                                                                                                   |
| 104543 |                               |                                                                                                                                                                                                                                                |
| 104544 | <i>LOGNAME</i>                | Determine the login name of the user.                                                                                                                                                                                                          |
| 104545 | <i>TZ</i>                     | Determine the timezone used to interpret the <i>date-time</i> option-argument. If <i>TZ</i> is unset or null, an unspecified default timezone shall be used.                                                                                   |
| 104546 |                               |                                                                                                                                                                                                                                                |
| 104547 | <b>ASYNCHRONOUS EVENTS</b>    |                                                                                                                                                                                                                                                |
| 104548 |                               | Default.                                                                                                                                                                                                                                       |
| 104549 | <b>STDOUT</b>                 |                                                                                                                                                                                                                                                |
| 104550 |                               | The <i>qselect</i> utility shall write zero or more batch <i>job_identifiers</i> to standard output.                                                                                                                                           |
| 104551 |                               | The <i>qselect</i> utility shall separate the batch <i>job_identifiers</i> written to standard output by white space.                                                                                                                          |
| 104552 |                               |                                                                                                                                                                                                                                                |
| 104553 |                               | The <i>qselect</i> utility shall write batch <i>job_identifiers</i> in the following format:                                                                                                                                                   |
| 104554 |                               | <i>sequence_number.server_name@server</i>                                                                                                                                                                                                      |
| 104555 | <b>STDERR</b>                 |                                                                                                                                                                                                                                                |
| 104556 |                               | The standard error shall be used only for diagnostic messages.                                                                                                                                                                                 |
| 104557 | <b>OUTPUT FILES</b>           |                                                                                                                                                                                                                                                |
| 104558 |                               | None.                                                                                                                                                                                                                                          |
| 104559 | <b>EXTENDED DESCRIPTION</b>   |                                                                                                                                                                                                                                                |
| 104560 |                               | None.                                                                                                                                                                                                                                          |
| 104561 | <b>EXIT STATUS</b>            |                                                                                                                                                                                                                                                |
| 104562 |                               | The following exit values shall be returned:                                                                                                                                                                                                   |
| 104563 | 0                             | Successful completion.                                                                                                                                                                                                                         |
| 104564 | >0                            | An error occurred.                                                                                                                                                                                                                             |
| 104565 | <b>CONSEQUENCES OF ERRORS</b> |                                                                                                                                                                                                                                                |
| 104566 |                               | Default.                                                                                                                                                                                                                                       |



**104567 APPLICATION USAGE**

104568 None.

**104569 EXAMPLES**

104570 The following example shows how a user might use the *qselect* utility in conjunction with the  
104571 *qdel* utility to delete all of his or her jobs in the queued state without affecting any jobs that are  
104572 already running:

104573 `qdel $(qselect -s q)`

104574 or:

104575 `qselect -s q || xargs qdel`

**104576 RATIONALE**

104577 The *qselect* utility allows users to acquire a list of job identifiers that match user-specified  
104578 selection criteria. The list of identifiers returned by the *qselect* utility conforms to the syntax of  
104579 the batch job identifier list processed by a utility such as *qmove*, *qdel*, and *qrls*. The *qselect* utility  
104580 is thus a powerful tool for causing another batch system utility to act upon a set of jobs that  
104581 match a list of selection criteria.

104582 The options of the *qselect* utility let the user apply a number of useful filters for selecting jobs.  
104583 Each option further restricts the selection of jobs. Many of the selection options allow the  
104584 specification of a relational operator. The FORTRAN-like syntax of the operator—that is,  
104585 ".lt."—was chosen rather than the C-like "<=" meta-characters.

104586 The **-a** option allows users to restrict the selected jobs to those that have been submitted (or  
104587 altered) to wait until a particular time. The time period is determined by the argument of this  
104588 option, which includes both a time and an operator—it is thus possible to select jobs waiting  
104589 until a specific time, jobs waiting until after a certain time, or those waiting for a time before the  
104590 specified time.

104591 The **-A** option allows users to restrict the selected jobs to those that have been submitted (or  
104592 altered) to charge a particular account.

104593 The **-c** option allows users to restrict the selected jobs to those whose checkpointing interval  
104594 falls within the specified range.

104595 The **-l** option allows users to select those jobs whose resource limits fall within the range  
104596 indicated by the value of the option. For example, a user could select those jobs for which the  
104597 CPU time limit is greater than two hours.

104598 The **-N** option allows users to select jobs by job name. For instance, all the parts of a task that  
104599 have been divided in parallel jobs might be given the same name, and thus manipulated as a  
104600 group by means of this option.

104601 The **-q** option allows users to select jobs in a specified queue.

104602 The **-r** option allows users to select only those jobs with a specified rerun criteria. For instance, a  
104603 user might select only those jobs that can be rerun for use with the *qrerun* utility.

104604 The **-s** option allows users to select only those jobs that are in a certain state.

104605 The **-u** option allows users to select jobs that have been submitted to execute under a particular  
104606 account.

104607 The selection criteria provided by the options of the *qselect* utility allow users to select jobs based  
104608 on all the appropriate attributes that can be assigned to jobs by the *qsub* utility.

104609 Historically, the *qselect* utility has not been a part of existing practice; it is an improvement that

104610 has been introduced in this volume of POSIX.1-2008.

104611 **FUTURE DIRECTIONS**

104612 The *qselect* utility may be removed in a future version.

104613 **SEE ALSO**

104614 Chapter 3 (on page 2397), *qdel*, *qrerun*, *qrls*, *qselect*, *qsub*, *touch*

104615 XBD Section 6.1 (on page 125), Chapter 8 (on page 173), Section 12.2 (on page 215)

104616 **CHANGE HISTORY**

104617 Derived from IEEE Std 1003.2d-1994.

104618 **Issue 7**

104619 The *qselect* utility is marked obsolescent.

104620 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

104621 **NAME**

104622       qsig — signal batch jobs

104623 **SYNOPSIS**104624 OB BE   qsig [-s *signal*] *job\_identifier...*104625 **DESCRIPTION**

104626       To signal a batch job is to send a signal to the session leader of the batch job. A batch job is  
 104627       signaled by sending a request to the batch server that manages the batch job. The *qsig* utility is a  
 104628       user-accessible batch client that requests the signaling of a batch job.

104629       The *qsig* utility shall signal those batch jobs for which a batch *job\_identifier* is presented to the  
 104630       utility. The *qsig* utility shall not signal any batch jobs whose batch *job\_identifiers* are not  
 104631       presented to the utility.

104632       The *qsig* utility shall signal batch jobs in the order in which the corresponding batch  
 104633       *job\_identifiers* are presented to the utility. If the *qsig* utility fails to process a batch *job\_identifier*  
 104634       successfully, the utility shall proceed to process the remaining batch *job\_identifiers*, if any.

104635       The *qsig* utility shall signal batch jobs by sending a *Signal Job Request* to the batch server that  
 104636       manages the batch job.

104637       For each successfully processed batch *job\_identifier*, the *qsig* utility shall have received a  
 104638       completion reply to each *Signal Job Request* sent to a batch server at the time the utility exits.

104639 **OPTIONS**104640       The *qsig* utility shall conform to XBD [Section 12.2](#) (on page 215).

104641       The following option shall be supported by the implementation:

104642       -s *signal*     Define the signal to be sent to the batch job.

104643       The *qsig* utility shall accept a *signal* option-argument that is either a symbolic signal  
 104644       name or an unsigned integer signal number (see the POSIX.1-1990 standard,  
 104645       Section 3.3.1.1). The *qsig* utility shall accept signal names for which the SIG prefix  
 104646       has been omitted.

104647       If the *signal* option-argument is a signal name, the *qsig* utility shall send that name.

104648       If the *signal* option-argument is a number, the *qsig* utility shall send the signal  
 104649       value represented by the number.

104650       If the -s option is not presented to the *qsig* utility, the utility shall send the signal  
 104651       SIGTERM to each signaled batch job.

104652 **OPERANDS**

104653       The *qsig* utility shall accept one or more operands that conform to the syntax for a batch  
 104654       *job\_identifier* (see [Section 3.3.1](#), on page 2419).

104655 **STDIN**

104656       Not used.

104657 **INPUT FILES**

104658       None.

104659 **ENVIRONMENT VARIABLES**104660       The following environment variables shall affect the execution of *qsig*:

|        |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|--------|-------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 104661 | <i>LANG</i>                   | Provide a default value for the internationalization variables that are unset or null. (See XBD <a href="#">Section 8.2</a> (on page 174) the precedence of internationalization variables used to determine the values of locale categories.)                                                                                                                                                                                       |
| 104662 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 104663 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 104664 | <i>LC_ALL</i>                 | If set to a non-empty string value, override the values of all the other internationalization variables.                                                                                                                                                                                                                                                                                                                             |
| 104665 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 104666 | <i>LC_CTYPE</i>               | Determine the locale for the interpretation of sequences of bytes of text data as characters (for example, single-byte as opposed to multi-byte characters in arguments).                                                                                                                                                                                                                                                            |
| 104667 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 104668 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 104669 | <i>LC_MESSAGES</i>            |                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 104670 |                               | Determine the locale that should be used to affect the format and contents of diagnostic messages written to standard error.                                                                                                                                                                                                                                                                                                         |
| 104671 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 104672 | <i>LOGNAME</i>                | Determine the login name of the user.                                                                                                                                                                                                                                                                                                                                                                                                |
| 104673 | <b>ASYNCHRONOUS EVENTS</b>    |                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 104674 |                               | Default.                                                                                                                                                                                                                                                                                                                                                                                                                             |
| 104675 | <b>STDOUT</b>                 |                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 104676 |                               | An implementation of the <i>qsig</i> utility may write informative messages to standard output.                                                                                                                                                                                                                                                                                                                                      |
| 104677 | <b>STDERR</b>                 |                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 104678 |                               | The standard error shall be used only for diagnostic messages.                                                                                                                                                                                                                                                                                                                                                                       |
| 104679 | <b>OUTPUT FILES</b>           |                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 104680 |                               | None.                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 104681 | <b>EXTENDED DESCRIPTION</b>   |                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 104682 |                               | None.                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 104683 | <b>EXIT STATUS</b>            |                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 104684 |                               | The following exit values shall be returned:                                                                                                                                                                                                                                                                                                                                                                                         |
| 104685 | 0                             | Successful completion.                                                                                                                                                                                                                                                                                                                                                                                                               |
| 104686 | >0                            | An error occurred.                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 104687 | <b>CONSEQUENCES OF ERRORS</b> |                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 104688 |                               | In addition to the default behavior, the <i>qsig</i> utility shall not be required to write a diagnostic message to standard error when the error reply received from a batch server indicates that the batch <i>job_identifier</i> does not exist on the server. Whether or not the <i>qsig</i> utility waits to output the diagnostic message while attempting to locate the batch job on other servers is implementation-defined. |
| 104689 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 104690 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 104691 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 104692 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 104693 | <b>APPLICATION USAGE</b>      |                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 104694 |                               | None.                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 104695 | <b>EXAMPLES</b>               |                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 104696 |                               | None.                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 104697 | <b>RATIONALE</b>              |                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 104698 |                               | The <i>qsig</i> utility allows users to signal batch jobs.                                                                                                                                                                                                                                                                                                                                                                           |
| 104699 |                               | A user may be unable to signal a batch job with the <i>kill</i> utility of the operating system for a number of reasons. First, the process ID of the batch job may be unknown to the user. Second, the processes of the batch job may be on a remote node. However, by virtue of communication between batch nodes, the <i>qsig</i> utility can arrange for the signaling of a process.                                             |
| 104700 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 104701 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 104702 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 104703 |                               | Because a batch job that is not running cannot be signaled, and because the signal may not                                                                                                                                                                                                                                                                                                                                           |

- 104704 terminate the batch job, the *qsig* utility is not a substitute for the *qdel* utility.
- 104705 The options of the *qsig* utility allow the user to specify the signal that is to be sent to the batch  
104706 job.
- 104707 The **-s** option allows users to specify a signal by name or by number, and thus override the  
104708 default signal. The POSIX.1-1990 standard defines signals by both name and number.
- 104709 The *qsig* utility is a new utility, *vis-a-vis* existing practice; it has been defined in this volume of  
104710 POSIX.1-2008 in response to user-perceived shortcomings in existing practice.
- 104711 **FUTURE DIRECTIONS**
- 104712 The *qsig* utility may be removed in a future version.
- 104713 **SEE ALSO**
- 104714 [Chapter 3](#) (on page 2397), *kill*, *qdel*
- 104715 XBD [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 215)
- 104716 **CHANGE HISTORY**
- 104717 Derived from IEEE Std 1003.2d-1994.
- 104718 **Issue 6**
- 104719 The *LC\_TIME* and *TZ* entries are removed from the ENVIRONMENT VARIABLES section.
- 104720 **Issue 7**
- 104721 The *qsig* utility is marked obsolescent.
- 104722 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

104723 **NAME**

104724       qstat — show status of batch jobs

104725 **SYNOPSIS**104726 OB BE   qstat [-f] *job\_identifier...*104727       qstat -Q [-f] *destination...*104728       qstat -B [-f] *server\_name...*104729 **DESCRIPTION**

104730       The status of a batch job, batch queue, or batch server is obtained by a request to the server. The  
 104731       *qstat* utility is a user-accessible batch client that requests the status of one or more batch jobs,  
 104732       batch queues, or servers, and writes the status information to standard output.

104733       For each successfully processed batch *job\_identifier*, the *qstat* utility shall display information  
 104734       about the corresponding batch job.

104735       For each successfully processed destination, the *qstat* utility shall display information about the  
 104736       corresponding batch queue.

104737       For each successfully processed server name, the *qstat* utility shall display information about the  
 104738       corresponding server.

104739       The *qstat* utility shall acquire batch job status information by sending a *Job Status Request* to a  
 104740       batch server. The *qstat* utility shall acquire batch queue status information by sending a *Queue*  
 104741       *Status Request* to a batch server. The *qstat* utility shall acquire server status information by  
 104742       sending a *Server Status Request* to a batch server.

104743 **OPTIONS**104744       The *qstat* utility shall conform to XBD [Section 12.2](#) (on page 215).

104745       The following options shall be supported by the implementation:

104746       **-f**       Specify that a full display is produced.

104747       The minimum contents of a full display are specified in the STDOUT section.

104748       Additional contents and format of a full display are implementation-defined.

104749       **-Q**       Specify that the operand is a destination.

104750       The *qstat* utility shall display information about each batch queue at each  
 104751       destination identified as an operand.

104752       **-B**       Specify that the operand is a server name.

104753       The *qstat* utility shall display information about each server identified as an  
 104754       operand.

104755 **OPERANDS**

104756       If the **-Q** option is presented to the *qstat* utility, the utility shall accept one or more operands that  
 104757       conform to the syntax for a destination (see [Section 3.3.2](#), on page 2420).

104758       If the **-B** option is presented to the *qstat* utility, the utility shall accept one or more *server\_name*  
 104759       operands.

104760       If neither the **-B** nor the **-Q** option is presented to the *qstat* utility, the utility shall accept one or  
 104761       more operands that conform to the syntax for a batch *job\_identifier* (see [Section 3.3.1](#), on page  
 104762       2419).

104763 **STDIN**

104764 Not used.

104765 **INPUT FILES**

104766 None.

104767 **ENVIRONMENT VARIABLES**104768 The following environment variables shall affect the execution of *qstat*:104769 *HOME* Determine the pathname of the user's home directory.

104770 *LANG* Provide a default value for the internationalization variables that are unset or null.  
 104771 (See XBD [Section 8.2](#) (on page 174) the precedence of internationalization variables  
 104772 used to determine the values of locale categories.)

104773 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
 104774 internationalization variables.

104775 *LC\_COLLATE*

104776 Determine the locale for the behavior of ranges, equivalence classes, and multi-  
 104777 character collating elements within regular expressions.

104778 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
 104779 characters (for example, single-byte as opposed to multi-byte characters in  
 104780 arguments).

104781 *LC\_MESSAGES*

104782 Determine the locale that should be used to affect the format and contents of  
 104783 diagnostic messages written to standard error.

104784 *LC\_NUMERIC*

104785 Determine the locale for selecting the radix character used when writing floating-  
 104786 point formatted output.

104787 **ASYNCHRONOUS EVENTS**

104788 Default.

104789 **STDOUT**

104790 If an operand presented to the *qstat* utility is a batch *job\_identifier* and the *-f* option is not  
 104791 specified, the *qstat* utility shall display the following items on a single line, in the stated order,  
 104792 with white space between each item, for each successfully processed operand:

- 104793 • The batch *job\_identifier*
- 104794 • The batch job name
- 104795 • The *Job\_Owner* attribute
- 104796 • The CPU time used by the batch job
- 104797 • The batch job state
- 104798 • The batch job location

104799 If an operand presented to the *qstat* utility is a batch *job\_identifier* and the *-f* option is specified,  
 104800 the *qstat* utility shall display the following items for each success fully processed operand:

- 104801 • The batch *job\_identifier*
- 104802 • The batch job name

- 104803 • The *Job\_Owner* attribute
  - 104804 • The execution user ID
  - 104805 • The CPU time used by the batch job
  - 104806 • The batch job state
  - 104807 • The batch job location
  - 104808 • Additional implementation-defined information, if any, about the batch job or batch queue
- 104809 If an operand presented to the *qstat* utility is a destination, the **-Q** option is specified, and the **-f**  
 104810 option is not specified, the *qstat* utility shall display the following items on a single line, in the  
 104811 stated order, with white space between each item, for each successfully processed operand:
- 104812 • The batch queue name
  - 104813 • The maximum number of batch jobs that shall be run in the batch queue concurrently
  - 104814 • The total number of batch jobs in the batch queue
  - 104815 • The status of the batch queue
  - 104816 • For each state, the number of batch jobs in that state in the batch queue and the name of  
 104817 the state
  - 104818 • The type of batch queue (execution or routing)
- 104819 If the operands presented to the *qstat* utility are destinations, the **-Q** option is specified, and the  
 104820 **-f** option is specified, the *qstat* utility shall display the following items for each successfully  
 104821 processed operand:
- 104822 • The batch queue name
  - 104823 • The maximum number of batch jobs that shall be run in the batch queue concurrently
  - 104824 • The total number of batch jobs in the batch queue
  - 104825 • The status of the batch queue
  - 104826 • For each state, the number of batch jobs in that state in the batch queue and the name of  
 104827 the state
  - 104828 • The type of batch queue (execution or routing)
  - 104829 • Additional implementation-defined information, if any, about the batch queue
- 104830 If the operands presented to the *qstat* utility are batch server names, the **-B** option is specified,  
 104831 and the **-f** option is not specified, the *qstat* utility shall display the following items on a single  
 104832 line, in the stated order, with white space between each item, for each successfully processed  
 104833 operand:
- 104834 • The batch server name
  - 104835 • The maximum number of batch jobs that shall be run in the batch queue concurrently
  - 104836 • The total number of batch jobs managed by the batch server
  - 104837 • The status of the batch server
  - 104838 • For each state, the number of batch jobs in that state and the name of the state
- 104839 If the operands presented to the *qstat* utility are server names, the **-B** option is specified, and the  
 104840 **-f** option is specified, the *qstat* utility shall display the following items for each successfully



104841 processed operand:

- 104842 • The server name
- 104843 • The maximum number of batch jobs that shall be run in the batch queue concurrently
- 104844 • The total number of batch jobs managed by the server
- 104845 • The status of the server
- 104846 • For each state, the number of batch jobs in that state and the name of the state
- 104847 • Additional implementation-defined information, if any, about the server

#### 104848 **STDERR**

104849 The standard error shall be used only for diagnostic messages.

#### 104850 **OUTPUT FILES**

104851 None.

#### 104852 **EXTENDED DESCRIPTION**

104853 None.

#### 104854 **EXIT STATUS**

104855 The following exit values shall be returned:

- 104856 0 Successful completion.
- 104857 >0 An error occurred.

#### 104858 **CONSEQUENCES OF ERRORS**

104859 In addition to the default behavior, the *qstat* utility shall not be required to write a diagnostic  
 104860 message to standard error when the error reply received from a batch server indicates that the  
 104861 batch *job\_identifier* does not exist on the server. Whether or not the *qstat* utility waits to output  
 104862 the diagnostic message while attempting to locate the batch job on other servers is  
 104863 implementation-defined.

#### 104864 **APPLICATION USAGE**

104865 None.

#### 104866 **EXAMPLES**

104867 None.

#### 104868 **RATIONALE**

104869 The *qstat* utility allows users to display the status of jobs and list the batch jobs in queues.

104870 The operands of the *qstat* utility may be either job identifiers, queues (specified as destination  
 104871 identifiers), or batch server names. The **-Q** and **-B** options, or absence thereof, indicate the  
 104872 nature of the operands.

104873 The other options of the *qstat* utility allow the user to control the amount of information  
 104874 displayed and the format in which it is displayed. Should a user wish to display the status of a  
 104875 set of jobs that match a selection criteria, the *qselect* utility may be used to acquire such a list.

104876 The **-f** option allows users to request a “full” display in an implementation-defined format.

104877 Historically, the *qstat* utility has been a part of the NQS and its derivatives, the existing practice  
 104878 on which it is based.

**104879 FUTURE DIRECTIONS**

104880 The *qstat* utility may be removed in a future version.

**104881 SEE ALSO**

104882 [Chapter 3](#) (on page 2397), [qselect](#)

104883 XBD [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 215)

**104884 CHANGE HISTORY**

104885 Derived from IEEE Std 1003.2d-1994.

**104886 Issue 6**

104887 IEEE PASC Interpretation 1003.2 #191 is applied, removing the following ENVIRONMENT  
104888 VARIABLES listed as affecting *qstat*: *COLUMNS*, *LINES*, *LOGNAME*, *TERM*, and *TZ*.

104889 The *LC\_TIME* entry is also removed from the ENVIRONMENT VARIABLES section.

**104890 Issue 7**

104891 The *qstat* utility is marked obsolescent.

104892 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

104893 **NAME**

104894       qsub — submit a script

104895 **SYNOPSIS**

```

104896 OB BE qsub [-a date_time] [-A account_string] [-c interval]
104897 [-C directive_prefix] [-e path_name] [-h] [-j join_list]
104898 [-k keep_list] [-m mail_options] [-M mail_list] [-N name]
104899 [-o path_name] [-p priority] [-q destination] [-r y|n]
104900 [-S path_name_list] [-u user_list] [-v variable_list] [-V]
104901 [-z] [script]

```

104902 **DESCRIPTION**

104903       To submit a script is to create a batch job that executes the script. A script is submitted by a  
 104904       request to a batch server. The *qsub* utility is a user-accessible batch client that submits a script.

104905       Upon successful completion, the *qsub* utility shall have created a batch job that will execute the  
 104906       submitted script.

104907       The *qsub* utility shall submit a script by sending a *Queue Job Request* to a batch server.

104908       The *qsub* utility shall place the value of the following environment variables in the *Variable\_List*  
 104909       attribute of the batch job: *HOME*, *LANG*, *LOGNAME*, *PATH*, *MAIL*, *SHELL*, and *TZ*. The name  
 104910       of the environment variable shall be the current name prefixed with the string *PBS\_O\_*.

104911       **Note:**     If the current value of the *HOME* variable in the environment space of the *qsub* utility is  
 104912       */aa/bb/cc*, then *qsub* shall place *PBS\_O\_HOME=/aa/bb/cc* in the *Variable\_List* attribute of the  
 104913       batch job.

104914       In addition to the variables described above, the *qsub* utility shall add the following variables  
 104915       with the indicated values to the variable list:

104916       *PBS\_O\_WORKDIR*     The absolute path of the current working directory of the *qsub* utility  
 104917       process.

104918       *PBS\_O\_HOST*        The name of the host on which the *qsub* utility is running.

104919 **OPTIONS**

104920       The *qsub* utility shall conform to XBD [Section 12.2](#) (on page 215).

104921       The following options shall be supported by the implementation:

104922       **-a date\_time**    Define the time at which a batch job becomes eligible for execution.

104923       The *qsub* utility shall accept an option-argument that conforms to the syntax of the  
 104924       *time* operand of the *touch* utility.

Table 4-19 Environment Variable Values (Utilities)

| Variable Name        | Value at qsub Time        |
|----------------------|---------------------------|
| <i>PBS_O_HOME</i>    | <i>HOME</i>               |
| <i>PBS_O_HOST</i>    | Client host name          |
| <i>PBS_O_LANG</i>    | <i>LANG</i>               |
| <i>PBS_O_LOGNAME</i> | <i>LOGNAME</i>            |
| <i>PBS_O_PATH</i>    | <i>PATH</i>               |
| <i>PBS_O_MAIL</i>    | <i>MAIL</i>               |
| <i>PBS_O_SHELL</i>   | <i>SHELL</i>              |
| <i>PBS_O_TZ</i>      | <i>TZ</i>                 |
| <i>PBS_O_WORKDIR</i> | Current working directory |

**Note:** The server that initiates execution of the batch job will add other variables to the batch job's environment; see [Section 3.2.2.1](#) (on page 2403).

The *qsub* utility shall set the *Execution\_Time* attribute of the batch job to the number of seconds since the Epoch that is equivalent to the local time expressed by the value of the *date\_time* option-argument. The Epoch is defined in XBD [Section 3.150](#) (on page 57).

If the *-a* option is not presented to the *qsub* utility, the utility shall set the *Execution\_Time* attribute of the batch job to a time (number of seconds since the Epoch) that is earlier than the time at which the utility exits.

#### *-A account\_string*

Define the account to which the resource consumption of the batch job should be charged.

The syntax of the *account\_string* option-argument is unspecified.

The *qsub* utility shall set the *Account\_Name* attribute of the batch job to the value of the *account\_string* option-argument.

If the *-A* option is not presented to the *qsub* utility, the utility shall omit the *Account\_Name* attribute from the attributes of the batch job.

#### *-c interval*

Define whether the batch job should be checkpointed, and if so, how often.

The *qsub* utility shall accept a value for the interval option-argument that is one of the following:

*n* No checkpointing shall be performed on the batch job (NO\_CHECKPOINT).

*s* Checkpointing shall be performed only when the batch server is shut down (CHECKPOINT\_AT\_SHUTDOWN).

*c* Automatic periodic checkpointing shall be performed at the *Minimum\_Cpu\_Interval* attribute of the batch queue, in units of CPU minutes (CHECKPOINT\_AT\_MIN\_CPU\_INTERVAL).

*c=minutes* Automatic periodic checkpointing shall be performed every *minutes* of CPU time, or every *Minimum\_Cpu\_Interval* minutes, whichever is greater. The *minutes* argument shall conform to the syntax for unsigned integers and shall be greater than zero.

The *qsub* utility shall set the *Checkpoint* attribute of the batch job to the value of the

104968 *interval* option-argument.

104969 If the `-c` option is not presented to the *qsub* utility, the utility shall set the *Checkpoint*  
 104970 attribute of the batch job to the single character `'u'`  
 104971 (`CHECKPOINT_UNSPECIFIED`).

104972 **-C** *directive\_prefix*  
 104973 Define the prefix that declares a directive to the *qsub* utility within the script.

104974 The *directive\_prefix* is not a batch job attribute; it affects the behavior of the *qsub*  
 104975 utility.

104976 If the `-C` option is presented to the *qsub* utility, and the value of the *directive\_prefix*  
 104977 option-argument is the null string, the utility shall not scan the script file for  
 104978 directives. If the `-C` option is not presented to the *qsub* utility, then the value of the  
 104979 *PBS\_DPREFIX* environment variable is used. If the environment variable is not  
 104980 defined, then `#PBS` encoded in the portable character set is the default.

104981 **-e** *path\_name*  
 104982 Define the path to be used for the standard error stream of the batch job.

104983 The *qsub* utility shall accept a *path\_name* option-argument which can be preceded  
 104984 by a host name element of the form *hostname*:

104985 If the *path\_name* option-argument constitutes an absolute pathname, the *qsub* utility  
 104986 shall set the *Error\_Path* attribute of the batch job to the value of the *path\_name*  
 104987 option-argument.

104988 If the *path\_name* option-argument constitutes a relative pathname and no host  
 104989 name element is specified, the *qsub* utility shall set the *Error\_Path* attribute of the  
 104990 batch job to the value of the absolute pathname derived by expanding the  
 104991 *path\_name* option-argument relative to the current directory of the process  
 104992 executing *qsub*.

104993 If the *path\_name* option-argument constitutes a relative pathname and a host name  
 104994 element is specified, the *qsub* utility shall set the *Error\_Path* attribute of the batch  
 104995 job to the value of the *path\_name* option-argument without expansion. The host  
 104996 name element shall be included.

104997 If the *path\_name* option-argument does not include a host name element, the *qsub*  
 104998 utility shall prefix the pathname with *hostname*:, where *hostname* is the name of the  
 104999 host upon which the *qsub* utility is being executed.

105000 If the `-e` option is not presented to the *qsub* utility, the utility shall set the  
 105001 *Error\_Path* attribute of the batch job to the host name and path of the current  
 105002 directory of the submitting process and the default filename.

105003 The default filename for standard error has the following format:  
 105004 *job\_name.esquence\_number*

105005 **-h** Specify that a USER hold is applied to the batch job.

105006 The *qsub* utility shall set the value of the *Hold\_Types* attribute of the batch job to the  
 105007 value USER.

105008 If the `-h` option is not presented to the *qsub* utility, the utility shall set the  
 105009 *Hold\_Types* attribute of the batch job to the value NO\_HOLD.

105010        **-j** *join\_list*    Define which streams of the batch job are to be merged. The *qsub* **-j** option shall  
 105011                            accept a value for the *join\_list* option-argument that is a string of alphanumeric  
 105012                            characters in the portable character set (see XBD [Section 6.1](#), on page 125).

105013                            The *qsub* utility shall accept a *join\_list* option-argument that consists of one or more  
 105014                            of the characters 'e' and 'o', or the single character 'n'.

105015                            All of the other batch job output streams specified will be merged into the output  
 105016                            stream represented by the character listed first in the *join\_list* option-argument.

105017                            For each unique character in the *join\_list* option-argument, the *qsub* utility shall  
 105018                            add a value to the *Join\_Path* attribute of the batch job as follows, each representing  
 105019                            a different batch job stream to join:

105020                            e    The standard error of the batch job (JOIN\_STD\_ERROR).

105021                            o    The standard output of the batch job (JOIN\_STD\_OUTPUT).

105022                            An existing *Join\_Path* attribute can be cleared by the following join type:

105023                            n    NO\_JOIN

105024                            If 'n' is specified, then no files are joined. The *qsub* utility shall consider it an error  
 105025                            if any join type other than 'n' is combined with join type 'n'.

105026                            Strictly conforming applications shall not repeat any of the characters 'e', 'o', or  
 105027                            'n' within the *join\_list* option-argument. The *qsub* utility shall permit the  
 105028                            repetition of characters, but shall not assign additional meaning to the repeated  
 105029                            characters.

105030                            An implementation may define other join types. The conformance document for an  
 105031                            implementation shall describe any additional batch job streams, how they are  
 105032                            specified, their internal behavior, and how they affect the behavior of the utility.

105033                            If the **-j** option is not presented to the *qsub* utility, the utility shall set the value of  
 105034                            the *Join\_Path* attribute of the batch job to NO\_JOIN.

105035        **-k** *keep\_list*    Define which output of the batch job to retain on the execution host.

105036                            The *qsub* **-k** option shall accept a value for the *keep\_list* option-argument that is a  
 105037                            string of alphanumeric characters in the portable character set (see XBD [Section](#)  
 105038                            [6.1](#), on page 125).

105039                            The *qsub* utility shall accept a *keep\_list* option-argument that consists of one or  
 105040                            more of the characters 'e' and 'o', or the single character 'n'.

105041                            For each unique character in the *keep\_list* option-argument, the *qsub* utility shall  
 105042                            add a value to the *Keep\_Files* attribute of the batch job as follows, each representing  
 105043                            a different batch job stream to keep:

105044                            e    The standard error of the batch job (KEEP\_STD\_ERROR).

105045                            o    The standard output of the batch job (KEEP\_STD\_OUTPUT).

105046                            If both 'e' and 'o' are specified, then both files are retained. An existing  
 105047                            *Keep\_Files* attribute can be cleared by the following keep type:

105048                            n    NO\_KEEP

105049                            If 'n' is specified, then no files are retained. The *qsub* utility shall consider it an  
 105050                            error if any keep type other than 'n' is combined with keep type 'n'.

105051 Strictly conforming applications shall not repeat any of the characters 'e', 'o', or  
 105052 'n' within the *keep\_list* option-argument. The *qsub* utility shall permit the  
 105053 repetition of characters, but shall not assign additional meaning to the repeated  
 105054 characters.

105055 An implementation may define other keep types. The conformance document for  
 105056 an implementation shall describe any additional keep types, how they are  
 105057 specified, their internal behavior, and how they affect the behavior of the utility. If  
 105058 the **-k** option is not presented to the *qsub* utility, the utility shall set the *Keep\_Files*  
 105059 attribute of the batch job to the value NO\_KEEP.

105060 **-m** *mail\_options*

105061 Define the points in the execution of the batch job at which the batch server that  
 105062 manages the batch job shall send mail about a change in the state of the batch job.

105063 The *qsub* **-m** option shall accept a value for the *mail\_options* option-argument that  
 105064 is a string of alphanumeric characters in the portable character set (see XBD [Section](#)  
 105065 [6.1](#), on page 125).

105066 The *qsub* utility shall accept a value for the *mail\_options* option-argument that is a  
 105067 string of one or more of the characters 'e', 'b', and 'a', or the single character  
 105068 'n'.

105069 For each unique character in the *mail\_options* option-argument, the *qsub* utility shall  
 105070 add a value to the *Mail\_Users* attribute of the batch job as follows, each  
 105071 representing a different time during the life of a batch job at which to send mail:

105072 e MAIL\_AT\_EXIT

105073 b MAIL\_AT\_BEGINNING

105074 a MAIL\_AT\_ABORT

105075 If any of these characters are duplicated in the *mail\_options* option-argument, the  
 105076 duplicates shall be ignored.

105077 An existing *Mail\_Points* attribute can be cleared by the following mail type:

105078 n NO\_MAIL

105079 If 'n' is specified, then mail is not sent. The *qsub* utility shall consider it an error if  
 105080 any mail type other than 'n' is combined with mail type 'n'.

105081 Strictly conforming applications shall not repeat any of the characters 'e', 'b',  
 105082 'a', or 'n' within the *mail\_options* option-argument.

105083 The *qsub* utility shall permit the repetition of characters, but shall not assign  
 105084 additional meaning to the repeated characters. An implementation may define  
 105085 other mail types. The conformance document for an implementation shall describe  
 105086 any additional mail types, how they are specified, their internal behavior, and how  
 105087 they affect the behavior of the utility.

105088 If the **-m** option is not presented to the *qsub* utility, the utility shall set the  
 105089 *Mail\_Points* attribute to the value MAIL\_AT\_ABORT.

105090 **-M** *mail\_list* Define the list of users to which a batch server that executes the batch job shall  
 105091 send mail, if the server sends mail about the batch job.

105092 The syntax of the *mail\_list* option-argument is unspecified.

105093 If the implementation of the *qsub* utility uses a name service to locate users, the

|        |                     |                                                                                                                |
|--------|---------------------|----------------------------------------------------------------------------------------------------------------|
| 105094 |                     | utility should accept the syntax used by the name service.                                                     |
| 105095 |                     | If the implementation of the <i>qsub</i> utility does not use a name service to locate users,                  |
| 105096 |                     | the implementation should accept the following syntax for user names:                                          |
| 105097 |                     | <i>mail_address</i> [ <i>,</i> <i>mail_address</i> <i>,</i> . . .]                                             |
| 105098 |                     | The interpretation of <i>mail_address</i> is implementation-defined.                                           |
| 105099 |                     | The <i>qsub</i> utility shall set the <i>Mail_Users</i> attribute of the batch job to the value of the         |
| 105100 |                     | <i>mail_list</i> option-argument.                                                                              |
| 105101 |                     | If the <b>-M</b> option is not presented to the <i>qsub</i> utility, the utility shall place only the          |
| 105102 |                     | user name and host name for the current process in the <i>Mail_Users</i> attribute of the                      |
| 105103 |                     | batch job.                                                                                                     |
| 105104 | <b>-N name</b>      | Define the name of the batch job.                                                                              |
| 105105 |                     | The <i>qsub</i> <b>-N</b> option shall accept a value for the <i>name</i> option-argument that is a            |
| 105106 |                     | string of up to 15 alphanumeric characters in the portable character set (see XBD                              |
| 105107 |                     | <a href="#">Section 6.1</a> , on page 125) where the first character is alphabetic.                            |
| 105108 |                     | The <i>qsub</i> utility shall set the value of the <i>Job_Name</i> attribute of the batch job to the           |
| 105109 |                     | value of the <i>name</i> option-argument.                                                                      |
| 105110 |                     | If the <b>-N</b> option is not presented to the <i>qsub</i> utility, the utility shall set the <i>Job_Name</i> |
| 105111 |                     | attribute of the batch job to the name of the <i>script</i> argument from which the                            |
| 105112 |                     | directory specification if any, has been removed.                                                              |
| 105113 |                     | If the <b>-N</b> option is not presented to the <i>qsub</i> utility, and the script is read from               |
| 105114 |                     | standard input, the utility shall set the <i>Job_Name</i> attribute of the batch job to the                    |
| 105115 |                     | value STDIN.                                                                                                   |
| 105116 | <b>-o path_name</b> |                                                                                                                |
| 105117 |                     | Define the path for the standard output of the batch job.                                                      |
| 105118 |                     | The <i>qsub</i> utility shall accept a <i>path_name</i> option-argument that conforms to the                   |
| 105119 |                     | syntax of the <i>path_name</i> element defined in the System Interfaces volume of                              |
| 105120 |                     | POSIX.1-2008, which can be preceded by a host name element of the form                                         |
| 105121 |                     | <i>hostname</i> ..                                                                                             |
| 105122 |                     | If the <i>path_name</i> option-argument constitutes an absolute pathname, the <i>qsub</i> utility              |
| 105123 |                     | shall set the <i>Output_Path</i> attribute of the batch job to the value of the <i>path_name</i>               |
| 105124 |                     | option-argument without expansion.                                                                             |
| 105125 |                     | If the <i>path_name</i> option-argument constitutes a relative pathname and no host                            |
| 105126 |                     | name element is specified, the <i>qsub</i> utility shall set the <i>Output_Path</i> attribute of the           |
| 105127 |                     | batch job to the pathname derived by expanding the value of the <i>path_name</i> option-                       |
| 105128 |                     | argument relative to the current directory of the process executing the <i>qsub</i> .                          |
| 105129 |                     | If the <i>path_name</i> option-argument constitutes a relative pathname and a host name                        |
| 105130 |                     | element is specified, the <i>qsub</i> utility shall set the <i>Output_Path</i> attribute of the batch          |
| 105131 |                     | job to the value of the <i>path_name</i> option-argument without expansion.                                    |
| 105132 |                     | If the <i>path_name</i> option-argument does not specify a host name element, the <i>qsub</i>                  |
| 105133 |                     | utility shall prefix the pathname with <i>hostname</i> ., where <i>hostname</i> is the name of the             |
| 105134 |                     | host upon which the <i>qsub</i> utility is executing.                                                          |
| 105135 |                     | If the <b>-o</b> option is not presented to the <i>qsub</i> utility, the utility shall set the                 |
| 105136 |                     | <i>Output_Path</i> attribute of the batch job to the host name and path of the current                         |



|        |                          |                                                                                                                                                                                                                                      |
|--------|--------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 105137 |                          | directory of the submitting process and the default filename.                                                                                                                                                                        |
| 105138 |                          | The default filename for standard output has the following format:                                                                                                                                                                   |
| 105139 |                          | <i>job_name.osequence_number</i>                                                                                                                                                                                                     |
| 105140 | <b>-p priority</b>       | Define the priority the batch job should have relative to other batch jobs owned by the batch server.                                                                                                                                |
| 105141 |                          |                                                                                                                                                                                                                                      |
| 105142 |                          | The <i>qsub</i> utility shall set the <i>Priority</i> attribute of the batch job to the value of the <i>priority</i> option-argument.                                                                                                |
| 105143 |                          |                                                                                                                                                                                                                                      |
| 105144 |                          | If the <b>-p</b> option is not presented to the <i>qsub</i> utility, the value of the <i>Priority</i> attribute is implementation-defined.                                                                                           |
| 105145 |                          |                                                                                                                                                                                                                                      |
| 105146 |                          | The <i>qsub</i> utility shall accept a value for the <i>priority</i> option-argument that conforms to the syntax for signed decimal integers, and which is not less than -1 024 and not greater than 1 023.                          |
| 105147 |                          |                                                                                                                                                                                                                                      |
| 105148 |                          |                                                                                                                                                                                                                                      |
| 105149 | <b>-q destination</b>    | Define the destination of the batch job.                                                                                                                                                                                             |
| 105150 |                          |                                                                                                                                                                                                                                      |
| 105151 |                          | The destination is not a batch job attribute; it determines the batch server, and possibly the batch queue, to which the <i>qsub</i> utility batch queues the batch job.                                                             |
| 105152 |                          |                                                                                                                                                                                                                                      |
| 105153 |                          | The <i>qsub</i> utility shall submit the script to the batch server named by the <i>destination</i> option-argument or the server that owns the batch queue named in the <i>destination</i> option-argument.                         |
| 105154 |                          |                                                                                                                                                                                                                                      |
| 105155 |                          |                                                                                                                                                                                                                                      |
| 105156 |                          | The <i>qsub</i> utility shall accept an option-argument for the <b>-q</b> option that conforms to the syntax for a destination (see <a href="#">Section 3.3.2</a> , on page 2420).                                                   |
| 105157 |                          |                                                                                                                                                                                                                                      |
| 105158 |                          | If the <b>-q</b> option is not presented to the <i>qsub</i> utility, the <i>qsub</i> utility shall submit the batch job to the default destination. The mechanism for determining the default destination is implementation-defined. |
| 105159 |                          |                                                                                                                                                                                                                                      |
| 105160 |                          |                                                                                                                                                                                                                                      |
| 105161 | <b>-r y   n</b>          | Define whether the batch job is rerunnable.                                                                                                                                                                                          |
| 105162 |                          | If the value of the option-argument is <i>y</i> , the <i>qsub</i> utility shall set the <i>Rerunable</i> attribute of the batch job to TRUE.                                                                                         |
| 105163 |                          |                                                                                                                                                                                                                                      |
| 105164 |                          | If the value of the option-argument is <i>n</i> , the <i>qsub</i> utility shall set the <i>Rerunable</i> attribute of the batch job to FALSE.                                                                                        |
| 105165 |                          |                                                                                                                                                                                                                                      |
| 105166 |                          | If the <b>-r</b> option is not presented to the <i>qsub</i> utility, the utility shall set the <i>Rerunable</i> attribute of the batch job to TRUE.                                                                                  |
| 105167 |                          |                                                                                                                                                                                                                                      |
| 105168 | <b>-S path_name_list</b> | Define the pathname to the shell under which the batch job is to execute.                                                                                                                                                            |
| 105169 |                          |                                                                                                                                                                                                                                      |
| 105170 |                          | The <i>qsub</i> utility shall accept a <i>path_name_list</i> option-argument that conforms to the following syntax:                                                                                                                  |
| 105171 |                          |                                                                                                                                                                                                                                      |
| 105172 |                          | <i>pathname</i> [ <i>@host</i> ][ <i>,</i> <i>,</i> <i>pathname</i> [ <i>@host</i> ] <i>,</i> <i>,</i> . . .]                                                                                                                        |
| 105173 |                          | The <i>qsub</i> utility shall allow only one pathname for a given host name. The <i>qsub</i> utility shall allow only one pathname that is missing a corresponding host name.                                                        |
| 105174 |                          |                                                                                                                                                                                                                                      |
| 105175 |                          | The <i>qsub</i> utility shall add a value to the <i>Shell_Path_List</i> attribute of the batch job for each entry in the <i>path_name_list</i> option-argument.                                                                      |
| 105176 |                          |                                                                                                                                                                                                                                      |
| 105177 |                          | If the <b>-S</b> option is not presented to the <i>qsub</i> utility, the utility shall set the                                                                                                                                       |

|        |                                |                                                                                                                          |
|--------|--------------------------------|--------------------------------------------------------------------------------------------------------------------------|
| 105178 |                                | <i>Shell_Path_List</i> attribute of the batch job to the null string.                                                    |
| 105179 |                                | The conformance document for an implementation shall describe the mechanism                                              |
| 105180 |                                | used to set the default shell and determine the current value of the default shell.                                      |
| 105181 |                                | An implementation shall provide a means for the installation to set the default                                          |
| 105182 |                                | shell to the login shell of the user under which the batch job is to execute. See                                        |
| 105183 |                                | <a href="#">Section 3.3.3</a> (on page 2421) for a means of removing <i>keyword=value</i> (and                           |
| 105184 |                                | <i>value@keyword</i> ) pairs and other general rules for list-oriented batch job attributes.                             |
| 105185 | <b>-u <i>user_list</i></b>     | Define the user name under which the batch job is to execute.                                                            |
| 105186 |                                | The <i>qsub</i> utility shall accept a <i>user_list</i> option-argument that conforms to the                             |
| 105187 |                                | following syntax:                                                                                                        |
| 105188 |                                | <i>username</i> [ <i>@host</i> ][ <i>,</i> <i>username</i> [ <i>@host</i> ] <i>,</i> <i>...</i> ]                        |
| 105189 |                                | The <i>qsub</i> utility shall accept only one user name that is missing a corresponding                                  |
| 105190 |                                | host name. The <i>qsub</i> utility shall accept only one user name per named host.                                       |
| 105191 |                                | The <i>qsub</i> utility shall add a value to the <i>User_List</i> attribute of the batch job for each                    |
| 105192 |                                | entry in the <i>user_list</i> option-argument.                                                                           |
| 105193 |                                | If the <b>-u</b> option is not presented to the <i>qsub</i> utility, the utility shall set the <i>User_List</i>          |
| 105194 |                                | attribute of the batch job to the user name from which the utility is executing. See                                     |
| 105195 |                                | <a href="#">Section 3.3.3</a> (on page 2421) for a means of removing <i>keyword=value</i> (and                           |
| 105196 |                                | <i>value@keyword</i> ) pairs and other general rules for list-oriented batch job attributes.                             |
| 105197 | <b>-v <i>variable_list</i></b> |                                                                                                                          |
| 105198 |                                | Add to the list of variables that are exported to the session leader of the batch job.                                   |
| 105199 |                                | A <i>variable_list</i> is a set of strings of either the form <i>&lt;variable&gt;</i> or <i>&lt;variable=value&gt;</i> , |
| 105200 |                                | delimited by <i>&lt;comma&gt;</i> characters.                                                                            |
| 105201 |                                | If the <b>-v</b> option is presented to the <i>qsub</i> utility, the utility shall also add, to the                      |
| 105202 |                                | environment <i>Variable_List</i> attribute of the batch job, every variable named in the                                 |
| 105203 |                                | environment <i>variable_list</i> option-argument and, optionally, values of specified                                    |
| 105204 |                                | variables.                                                                                                               |
| 105205 |                                | If a value is not provided on the command line, the <i>qsub</i> utility shall set the value                              |
| 105206 |                                | of each variable in the environment <i>Variable_List</i> attribute of the batch job to the                               |
| 105207 |                                | value of the corresponding environment variable for the process in which the                                             |
| 105208 |                                | utility is executing; see <a href="#">Table 4-19</a> (on page 3142).                                                     |
| 105209 |                                | A conforming application shall not repeat a variable in the environment                                                  |
| 105210 |                                | <i>variable_list</i> option-argument.                                                                                    |
| 105211 |                                | The <i>qsub</i> utility shall not repeat a variable in the environment <i>Variable_List</i> attribute                    |
| 105212 |                                | of the batch job. See <a href="#">Section 3.3.3</a> (on page 2421) for a means of removing                               |
| 105213 |                                | <i>keyword=value</i> (and <i>value@keyword</i> ) pairs and other general rules for list-oriented                         |
| 105214 |                                | batch job attributes.                                                                                                    |
| 105215 | <b>-V</b>                      |                                                                                                                          |
| 105216 |                                | Specify that all of the environment variables of the process are exported to the                                         |
|        |                                | context of the batch job.                                                                                                |
| 105217 |                                | The <i>qsub</i> utility shall place every environment variable in the process in which the                               |
| 105218 |                                | utility is executing in the list and shall set the value of each variable in the attribute                               |
| 105219 |                                | to the value of that variable in the process.                                                                            |

105220            **-z**            Specify that the utility does not write the batch *job\_identifier* of the created batch job  
105221            to standard output.

105222            If the **-z** option is presented to the *qsub* utility, the utility shall not write the batch  
105223            *job\_identifier* of the created batch job to standard output.

105224            If the **-z** option is not presented to the *qsub* utility, the utility shall write the  
105225            identifier of the created batch job to standard output.

#### 105226 OPERANDS

105227            The *qsub* utility shall accept a *script* operand that indicates the path to the script of the batch job.

105228            If the *script* operand is not presented to the *qsub* utility, or if the operand is the single-character  
105229            string **'-'**, the utility shall read the script from standard input.

105230            If the script represents a partial path, the *qsub* utility shall expand the path relative to the current  
105231            directory of the process executing the utility.

#### 105232 STDIN

105233            The *qsub* utility reads the script of the batch job from standard input if the script operand is  
105234            omitted or is the single character **'-'**.

#### 105235 INPUT FILES

105236            In addition to binding the file indicated by the *script* operand to the batch job, the *qsub* utility  
105237            reads the script file and acts on directives in the script.

#### 105238 ENVIRONMENT VARIABLES

105239            The following environment variables shall affect the execution of *qsub*:

105240            **LANG**            Provide a default value for the internationalization variables that are unset or null.  
105241            (See XBD [Section 8.2](#) (on page 174) the precedence of internationalization variables  
105242            used to determine the values of locale categories.)

105243            **LC\_ALL**            If set to a non-empty string value, override the values of all the other  
105244            internationalization variables.

105245            **LC\_CTYPE**          Determine the locale for the interpretation of sequences of bytes of text data as  
105246            characters (for example, single-byte as opposed to multi-byte characters in  
105247            arguments).

105248            **LC\_MESSAGES**

105249            Determine the locale that should be used to affect the format and contents of  
105250            diagnostic messages written to standard error.

105251            **LOGNAME**          Determine the login name of the user.

105252            **PBS\_DPREFIX**

105253            Determine the default prefix for directives within the script.

105254            **SHELL**            Determine the pathname of the preferred command language interpreter of the  
105255            user.

105256            **TZ**                Determine the timezone used to interpret the *date-time* option-argument. If *TZ* is  
105257            unset or null, an unspecified default timezone shall be used.

#### 105258 ASYNCHRONOUS EVENTS

105259            Once created, a batch job exists until it exits, aborts, or is deleted.

105260            After a batch job is created by the *qsub* utility, batch servers might route, execute, modify, or  
105261            delete the batch job.

105262 **STDOUT**

105263 The *qsub* utility writes the batch *job\_identifier* assigned to the batch job to standard output, unless  
 105264 the **-z** option is specified.

105265 **STDERR**

105266 The standard error shall be used only for diagnostic messages.

105267 **OUTPUT FILES**

105268 None.

105269 **EXTENDED DESCRIPTION**105270 **Script Preservation**

105271 The *qsub* utility shall make the script available to the server executing the batch job in such a  
 105272 way that the server executes the script as it exists at the time of submission.

105273 The *qsub* utility can send a copy of the script to the server with the *Queue Job Request* or store a  
 105274 temporary copy of the script in a location specified to the server.

105275 **Option Specification**

105276 A script can contain directives to the *qsub* utility.

105277 The *qsub* utility shall scan the lines of the script for directives, skipping blank lines, until the first  
 105278 line that begins with a string other than the directive string; if directives occur on subsequent  
 105279 lines, the utility shall ignore those directives.

105280 Lines are separated by a <newline>. If the first line of the script begins with "#!" or a <colon>  
 105281 (' : '), then it is skipped. The *qsub* utility shall process a line in the script as a directive if and  
 105282 only if the string of characters from the first non-white-space character on the line until the first  
 105283 <space> or <tab> on the line match the directive prefix. If a line in the script contains a directive  
 105284 and the final characters of the line are <backslash> and <newline>, then the next line shall be  
 105285 interpreted as a continuation of that directive.

105286 The *qsub* utility shall process the options and option-arguments contained on the directive prefix  
 105287 line using the same syntax as if the options were input on the *qsub* utility.

105288 The *qsub* utility shall continue to process a directive prefix line until after a <newline> is  
 105289 encountered. An implementation may ignore lines which, according to the syntax of the shell  
 105290 that will interpret the script, are comments. An implementation shall describe in the  
 105291 conformance document the format of any shell comments that it will recognize.

105292 If an option is present in both a directive and the arguments to the *qsub* utility, the utility shall  
 105293 ignore the option and the corresponding option-argument, if any, in the directive.

105294 If an option that is present in the directive is not present in the arguments to the *qsub* utility, the  
 105295 utility shall process the option and the option-argument, if any.

105296 In order of preference, the *qsub* utility shall select the directive prefix from one of the following  
 105297 sources:

- 105298 • If the **-C** option is presented to the utility, the value of the *directive\_prefix* option-argument
- 105299 • If the environment variable *PBS\_DPPREFIX* is defined, the value of that variable
- 105300 • The four-character string "#PBS" encoded in the portable character set

105301 If the **-C** option is present in the script file it shall be ignored.

**105302 EXIT STATUS**

105303 The following exit values shall be returned:

105304 0 Successful completion.

105305 >0 An error occurred.

**105306 CONSEQUENCES OF ERRORS**

105307 Default.

**105308 APPLICATION USAGE**

105309 None.

**105310 EXAMPLES**

105311 None.

**105312 RATIONALE**

105313 The *qsub* utility allows users to create a batch job that will process the script specified as the  
105314 operand of the utility.

105315 The options of the *qsub* utility allow users to control many aspects of the queuing and execution  
105316 of a batch job.

105317 The **-a** option allows users to designate the time after which the batch job will become eligible to  
105318 run. By specifying an execution time, users can take advantage of resources at off-peak hours,  
105319 synchronize jobs with chronologically predictable events, and perhaps take advantage of off-  
105320 peak pricing of computing time. For these reasons and others, a timing option is existing  
105321 practice on the part of almost every batch system, including NQS.

105322 The **-A** option allows users to specify the account that will be charged for the batch job. Support  
105323 for account is not mandatory for conforming batch servers.

105324 The **-C** option allows users to prescribe the prefix for directives within the script file. The default  
105325 prefix "**#PBS**" may be inappropriate if the script will be interpreted with an alternate shell, as  
105326 specified by the **-S** option.

105327 The **-c** option allows users to establish the checkpointing interval for their jobs. A checkpointing  
105328 system, which is not defined by this volume of POSIX.1-2008, allows recovery of a batch job at  
105329 the most recent checkpoint in the event of a crash. Checkpointing is typically used for jobs that  
105330 consume expensive computing time or must meet a critical schedule. Users should be allowed to  
105331 make the tradeoff between the overhead of checkpointing and the risk to the timely completion  
105332 of the batch job; therefore, this volume of POSIX.1-2008 provides the checkpointing interval  
105333 option. Support for checkpointing is optional for batch servers.

105334 The **-e** option allows users to redirect the standard error streams of their jobs to a non-default  
105335 path. For example, if the submitted script generally produces a great deal of useless error  
105336 output, a user might redirect the standard error output to the null device. Or, if the file system  
105337 holding the default location (the home directory of the user) has too little free space, the user  
105338 might redirect the standard error stream to a file in another file system.

105339 The **-h** option allows users to create a batch job that is held until explicitly released. The ability  
105340 to create a held job is useful when some external event must complete before the batch job can  
105341 execute. For example, the user might submit a held job and release it when the system load has  
105342 dropped.

105343 The **-j** option allows users to merge the standard error of a batch job into its standard output  
105344 stream, which has the advantage of showing the sequential relationship between output and  
105345 error messages.

The **-m** option allows users to designate those points in the execution of a batch job at which mail will be sent to the submitting user, or to the account(s) indicated by the **-M** option. By requesting mail notification at points of interest in the life of a job, the submitting user, or other designated users, can track the progress of a batch job.

The **-N** option allows users to associate a name with the batch job. The job name in no way affects the processing of the batch job, but rather serves as a mnemonic handle for users. For example, the batch job name can help the user distinguish between multiple jobs listed by the *qstat* utility.

The **-o** option allows users to redirect the standard output stream. A user might, for example, wish to redirect to the null device the standard output stream of a job that produces copious yet superfluous output.

The **-P** option allows users to designate the relative priority of a batch job for selection from a queue.

The **-q** option allows users to specify an initial queue for the batch job. If the user specifies a routing queue, the batch server routes the batch job to another queue for execution or further routing. If the user specifies a non-routing queue, the batch server of the queue eventually executes the batch job.

The **-r** option allows users to control whether the submitted job will be rerun if the controlling batch node fails during execution of the batch job. The **-r** option likewise allows users to indicate whether or not the batch job is eligible to be rerun by the *qrerun* utility. Some jobs cannot be correctly rerun because of changes they make in the state of databases or other aspects of their environment. This volume of POSIX.1-2008 specifies that the default, if the **-r** option is not presented to the utility, will be that the batch job cannot be rerun, since the result of rerunning a non-rerunnable job might be catastrophic.

The **-S** option allows users to specify the program (usually a shell) that will be invoked to process the script of the batch job. This option has been modified to allow a list of shell names and locations associated with different hosts.

The **-u** option is useful when the submitting user is authorized to use more than one account on a given host, in which case the **-u** option allows the user to select from among those accounts. The option-argument is a list of user-host pairs, so that the submitting user can provide different user identifiers for different nodes in the event the batch job is routed. The **-u** option provides a lot of flexibility to accommodate sites with complex account structures. Users that have the same user identifier on all the hosts they are authorized to use will not need to use the **-u** option.

The **-V** option allows users to export all their current environment variables, as of the time the batch job is submitted, to the context of the processes of the batch job.

The **-v** option allows users to export specific environment variables from their current process to the processes of the batch job.

The **-z** option allows users to suppress the writing of the batch job identifier to standard output. The **-z** option is an existing NQS practice that has been standardized.

Historically, the *qsub* utility has served the batch job-submission function in the NQS system, the existing practice on which it is based. Some changes and additions have been made to the *qsub* utility in this volume of POSIX.1-2008, *vis-a-vis* NQS, as a result of the growing pool of experience with distributed batch systems.

The set of features of the *qsub* utility as defined in this volume of POSIX.1-2008 appears to incorporate all the common existing practice on potentially conforming platforms.

**105391 FUTURE DIRECTIONS**

105392 The *qsub* utility may be removed in a future version.

**105393 SEE ALSO**

105394 [Chapter 3](#) (on page 2397), [qrerun](#), [qstat](#), [touch](#)

105395 XBD [Section 3.150](#) (on page 57), [Section 6.1](#) (on page 125), [Chapter 8](#) (on page 173), [Section 12.2](#)  
105396 (on page 215)

**105397 CHANGE HISTORY**

105398 Derived from IEEE Std 1003.2d-1994.

**105399 Issue 6**

105400 The `-I` option has been removed as there is no portable description of the resources that are  
105401 allowed or required by the batch job.

**105402 Issue 7**

105403 The *qsub* utility is marked obsolescent.

105404 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

105405 **NAME**

105406       read — read a line from standard input

105407 **SYNOPSIS**

105408       read [-r] var...

105409 **DESCRIPTION**105410       The *read* utility shall read a single line from standard input.

105411       By default, unless the *-r* option is specified, <backslash> shall act as an escape character. An  
 105412       unescaped <backslash> shall preserve the literal value of the following character, with the  
 105413       exception of a <newline>. If a <newline> follows the <backslash>, the *read* utility shall interpret  
 105414       this as line continuation. The <backslash> and <newline> shall be removed before splitting the  
 105415       input into fields. All other unescaped <backslash> characters shall be removed after splitting the  
 105416       input into fields.

105417       If standard input is a terminal device and the invoking shell is interactive, *read* shall prompt for a  
 105418       continuation line when it reads an input line ending with a <backslash> <newline>, unless the  
 105419       *-r* option is specified.

105420       The terminating <newline> (if any) shall be removed from the input and the results shall be split  
 105421       into fields as in the shell for the results of parameter expansion (see [Section 2.6.5](#), on page 2333);  
 105422       the first field shall be assigned to the first variable *var*, the second field to the second variable  
 105423       *var*, and so on. If there are fewer fields than there are *var* operands, the remaining *vars* shall be  
 105424       set to empty strings. If there are fewer *vars* than fields, the last *var* shall be set to a value  
 105425       comprising the following elements:

- 105426       • The field that corresponds to the last *var* in the normal assignment sequence described  
 105427       above
- 105428       • The delimiter(s) that follow the field corresponding to the last *var*
- 105429       • The remaining fields and their delimiters, with trailing *IFS* white space ignored

105430       The setting of variables specified by the *var* operands shall affect the current shell execution  
 105431       environment; see [Section 2.12](#) (on page 2353). If it is called in a subshell or separate utility  
 105432       execution environment, such as one of the following:

```
105433 (read foo)
105434 nohup read ...
105435 find . -exec read ... \;
```

105436       it shall not affect the shell variables in the caller's environment.

105437 **OPTIONS**105438       The *read* utility shall conform to XBD [Section 12.2](#) (on page 215).

105439       The following option is supported:

105440       *-r*       Do not treat a <backslash> character in any special way. Consider each  
 105441       <backslash> to be part of the input line.

105442 **OPERANDS**

105443       The following operand shall be supported:

105444       *var*       The name of an existing or nonexisting shell variable.



105445 **STDIN**

105446 The standard input shall be a text file.

105447 **INPUT FILES**

105448 None.

105449 **ENVIRONMENT VARIABLES**105450 The following environment variables shall affect the execution of *read*:

105451 *IFS* Determine the internal field separators used to delimit fields; see [Section 2.5.3](#) (on  
105452 page 2324).

105453 *LANG* Provide a default value for the internationalization variables that are unset or null.  
105454 (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
105455 variables used to determine the values of locale categories.)

105456 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
105457 internationalization variables.

105458 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
105459 characters (for example, single-byte as opposed to multi-byte characters in  
105460 arguments).

105461 *LC\_MESSAGES*

105462 Determine the locale that should be used to affect the format and contents of  
105463 diagnostic messages written to standard error.

105464 XSI *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

105465 *PS2* Provide the prompt string that an interactive shell shall write to standard error  
105466 when a line ending with a <backslash> <newline> is read and the *-r* option was  
105467 not specified.

105468 **ASYNCHRONOUS EVENTS**

105469 Default.

105470 **STDOUT**

105471 Not used.

105472 **STDERR**

105473 The standard error shall be used for diagnostic messages and prompts for continued input.

105474 **OUTPUT FILES**

105475 None.

105476 **EXTENDED DESCRIPTION**

105477 None.

105478 **EXIT STATUS**

105479 The following exit values shall be returned:

105480 0 Successful completion.

105481 &gt;0 End-of-file was detected or an error occurred.

105482 **CONSEQUENCES OF ERRORS**

105483 Default.

**105484 APPLICATION USAGE**

105485 The `-r` option is included to enable *read* to subsume the purpose of the *line* utility, which is not  
 105486 included in POSIX.1-2008.

**105487 EXAMPLES**

105488 The following command:

```
105489 while read -r xx yy
105490 do
105491 printf "%s %s\n" "$yy" "$xx"
105492 done < input_file
```

105493 prints a file with the first field of each line moved to the end of the line.

**105494 RATIONALE**

105495 The *read* utility historically has been a shell built-in. It was separated off into its own utility to  
 105496 take advantage of the richer description of functionality introduced by this volume of  
 105497 POSIX.1-2008.

105498 Since *read* affects the current shell execution environment, it is generally provided as a shell  
 105499 regular built-in. If it is called in a subshell or separate utility execution environment, such as one  
 105500 of the following:

```
105501 (read foo)
105502 nohup read ...
105503 find . -exec read ... \;
```

105504 it does not affect the shell variables in the environment of the caller.

105505 Although the standard input is required to be a text file, and therefore will always end with a  
 105506 <newline> (unless it is an empty file), the processing of continuation lines when the `-r` option is  
 105507 not used can result in the input not ending with a <newline>. This occurs if the last line of the  
 105508 input file ends with a <backslash> <newline>. It is for this reason that “if any” is used in “The  
 105509 terminating <newline> (if any) shall be removed from the input” in the description. It is not a  
 105510 relaxation of the requirement for standard input to be a text file.

**105511 FUTURE DIRECTIONS**

105512 None.

**105513 SEE ALSO**

105514 [Chapter 2](#) (on page 2319)

105515 XBD [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 215)

**105516 CHANGE HISTORY**

105517 First released in Issue 2.

**105518 Issue 7**

105519 Austin Group Interpretation 1003.1-2001 #194 is applied, clarifying the handling of the  
 105520 <backslash> escape character.

105521 SD5-XCU-ERN-126 is applied, clarifying that input lines end with a <newline>.

105522 The description of here-documents is removed from the *read* reference page.

105523 **NAME**105524        **renice** — set nice values of running processes105525 **SYNOPSIS**105526        **renice** [-g|-p|-u] -n *increment* *ID*...105527 **DESCRIPTION**

105528        The *renice* utility shall request that the nice values (see XBD [Section 3.240](#), on page 71) of one or  
 105529        more running processes be changed. By default, the applicable processes are specified by their  
 105530        process IDs. When a process group is specified (see **-g**), the request shall apply to all processes  
 105531        in the process group.

105532        The nice value shall be bounded in an implementation-defined manner. If the requested  
 105533        *increment* would raise or lower the nice value of the executed utility beyond implementation-  
 105534        defined limits, then the limit whose value was exceeded shall be used.

105535        When a user is *reniced*, the request applies to all processes whose saved set-user-ID matches the  
 105536        user ID corresponding to the user.

105537        Regardless of which options are supplied or any other factor, *renice* shall not alter the nice values  
 105538        of any process unless the user requesting such a change has appropriate privileges to do so for  
 105539        the specified process. If the user lacks appropriate privileges to perform the requested action, the  
 105540        utility shall return an error status.

105541        The saved set-user-ID of the user's process shall be checked instead of its effective user ID when  
 105542        *renice* attempts to determine the user ID of the process in order to determine whether the user  
 105543        has appropriate privileges.

105544 **OPTIONS**105545        The *renice* utility shall conform to XBD [Section 12.2](#) (on page 215), except for Guideline 9.

105546        The following options shall be supported:

105547        **-g**            Interpret the following operands as unsigned decimal integer process group IDs.

105548        **-n** *increment* Specify how the nice value of the specified process or processes is to be adjusted.  
 105549        The *increment* option-argument is a positive or negative decimal integer that shall  
 105550        be used to modify the nice value of the specified process or processes.

105551        Positive *increment* values shall cause a lower nice value. Negative *increment* values  
 105552        may require appropriate privileges and shall cause a higher nice value.

105553        **-p**            Interpret the following operands as unsigned decimal integer process IDs. The **-p**  
 105554        option is the default if no options are specified.

105555        **-u**            Interpret the following operands as users. If a user exists with a user name equal to  
 105556        the operand, then the user ID of that user is used in further processing. Otherwise,  
 105557        if the operand represents an unsigned decimal integer, it shall be used as the  
 105558        numeric user ID of the user.

105559 **OPERANDS**

105560        The following operands shall be supported:

105561        *ID*            A process ID, process group ID, or user name/user ID, depending on the option  
 105562        selected.

105563 **STDIN**

105564        Not used.

105565 **INPUT FILES**

105566 None.

105567 **ENVIRONMENT VARIABLES**105568 The following environment variables shall affect the execution of *renice*:

105569 *LANG* Provide a default value for the internationalization variables that are unset or null.  
 105570 (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 105571 variables used to determine the values of locale categories.)

105572 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
 105573 internationalization variables.

105574 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
 105575 characters (for example, single-byte as opposed to multi-byte characters in  
 105576 arguments).

105577 *LC\_MESSAGES*

105578 Determine the locale that should be used to affect the format and contents of  
 105579 diagnostic messages written to standard error.

105580 XSI *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

105581 **ASYNCHRONOUS EVENTS**

105582 Default.

105583 **STDOUT**

105584 Not used.

105585 **STDERR**

105586 The standard error shall be used only for diagnostic messages.

105587 **OUTPUT FILES**

105588 None.

105589 **EXTENDED DESCRIPTION**

105590 None.

105591 **EXIT STATUS**

105592 The following exit values shall be returned:

105593 0 Successful completion.

105594 &gt;0 An error occurred.

105595 **CONSEQUENCES OF ERRORS**

105596 Default.

105597 **APPLICATION USAGE**

105598 None.

105599 **EXAMPLES**

105600 1. Adjust the nice value so that process IDs 987 and 32 would have a lower nice value:

105601 `renice -n 5 -p 987 32`

105602 2. Adjust the nice value so that group IDs 324 and 76 would have a higher nice value, if the  
 105603 user has appropriate privileges to do so:

105604 `renice -n -4 -g 324 76`

3. Adjust the nice value so that numeric user ID 8 and user **sas** would have a lower nice value:

```
renice -n 4 -u 8 sas
```

Useful nice value increments on historical systems include 19 or 20 (the affected processes run only when nothing else in the system attempts to run) and any negative number (to make processes run faster).

#### RATIONALE

The *gid*, *pid*, and *user* specifications do not fit either the definition of operand or option-argument. However, for clarity, they have been included in the OPTIONS section, rather than the OPERANDS section.

The definition of nice value is not intended to suggest that all processes in a system have priorities that are comparable. Scheduling policy extensions such as the realtime priorities in the System Interfaces volume of POSIX.1-2008 make the notion of a single underlying priority for all scheduling policies problematic. Some implementations may implement the *nice*-related features to affect all processes on the system, others to affect just the general time-sharing activities implied by this volume of POSIX.1-2008, and others may have no effect at all. Because of the use of “implementation-defined” in *nice* and *renice*, a wide range of implementation strategies are possible.

Originally, this utility was written in the historical manner, using the term “nice value”. This was always a point of concern with users because it was never intuitively obvious what this meant. With a newer version of *renice*, which used the term “system scheduling priority”, it was hoped that novice users could better understand what this utility was meant to do. Also, it would be easier to document what the utility was meant to do. Unfortunately, the addition of the POSIX realtime scheduling capabilities introduced the concepts of process and thread scheduling priorities that were totally unaffected by the *nice*/*renice* utilities or the *nice()*/*setpriority()* functions. Continuing to use the term “system scheduling priority” would have incorrectly suggested that these utilities and functions were indeed affecting these realtime priorities. It was decided to revert to the historical term “nice value” to reference this unrelated process attribute.

Although this utility has use by system administrators (and in fact appears in the system administration portion of the BSD documentation), the standard developers considered that it was very useful for individual end users to control their own processes.

Earlier versions of this standard allowed the following forms in the SYNOPSIS:

```
renice nice_value[-p] pid...[-g gid...][-p pid...][-u user...]
renice nice_value -g gid...[-g gid...]-p pid...[-u user...]
renice nice_value -u user...[-g gid...]-p pid...[-u user...]
```

These forms are no longer specified by POSIX.1-2008 but may be present in some implementations.

#### FUTURE DIRECTIONS

None.

#### SEE ALSO

*nice*

XBD [Section 3.240](#) (on page 71), [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 215)

**CHANGE HISTORY**

105648  
105649 First released in Issue 4.

**Issue 5**

105650  
105651 In the SYNOPSIS, an ellipsis is added to the **-u** option in all three obsolescent forms.

**Issue 6**

105652  
105653 This utility is marked as part of the User Portability Utilities option.

105654 The APPLICATION USAGE section is added.

105655 The obsolescent forms of the SYNOPSIS are removed.

105656 Text previously conditional on POSIX\_SAVED\_IDS is mandatory in this version. This is a FIPS  
105657 requirement.

**Issue 7**

105658  
105659 Austin Group Interpretation 1003.1-2001 #027 is applied, clarifying that Guideline 9 of the Utility  
105660 Syntax Guidelines does not apply.

105661 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

105662 The *renice* utility is moved from the User Portability Utilities option to the Base. User Portability  
105663 Utilities is now an option for interactive utilities.

105664 **NAME**

105665       rm — remove directory entries

105666 **SYNOPSIS**105667       rm [-fiRr] *file*...105668 **DESCRIPTION**105669       The *rm* utility shall remove the directory entry specified by each *file* argument.

105670       If either of the files dot or dot-dot are specified as the basename portion of an operand (that is,  
 105671       the final pathname component) or if an operand resolves to the root directory, *rm* shall write a  
 105672       diagnostic message to standard error and do nothing more with such operands.

105673       For each *file* the following steps shall be taken:

- 105674       1. If the *file* does not exist:
  - 105675           a. If the **-f** option is not specified, *rm* shall write a diagnostic message to standard  
 105676           error.
  - 105677           b. Go on to any remaining *files*.
- 105678       2. If *file* is of type directory, the following steps shall be taken:
  - 105679           a. If neither the **-R** option nor the **-r** option is specified, *rm* shall write a diagnostic  
 105680           message to standard error, do nothing more with *file*, and go on to any remaining  
 105681           files.
  - 105682           b. If the **-f** option is not specified, and either the permissions of *file* do not permit  
 105683           writing and the standard input is a terminal or the **-i** option is specified, *rm* shall  
 105684           write a prompt to standard error and read a line from the standard input. If the  
 105685           response is not affirmative, *rm* shall do nothing more with the current file and go  
 105686           on to any remaining files.
  - 105687           c. For each entry contained in *file*, other than dot or dot-dot, the four steps listed here  
 105688           (1 to 4) shall be taken with the entry as if it were a *file* operand. The *rm* utility shall  
 105689           not traverse directories by following symbolic links into other parts of the  
 105690           hierarchy, but shall remove the links themselves.
  - 105691           d. If the **-i** option is specified, *rm* shall write a prompt to standard error and read a  
 105692           line from the standard input. If the response is not affirmative, *rm* shall do nothing  
 105693           more with the current file, and go on to any remaining files.
- 105694       3. If *file* is not of type directory, the **-f** option is not specified, and either the permissions of  
 105695       *file* do not permit writing and the standard input is a terminal or the **-i** option is specified,  
 105696       *rm* shall write a prompt to the standard error and read a line from the standard input. If  
 105697       the response is not affirmative, *rm* shall do nothing more with the current file and go on  
 105698       to any remaining files.
- 105699       4. If the current file is a directory, *rm* shall perform actions equivalent to the *rmdir*() function  
 105700       defined in the System Interfaces volume of POSIX.1-2008 called with a pathname of the  
 105701       current file used as the *path* argument. If the current file is not a directory, *rm* shall  
 105702       perform actions equivalent to the *unlink*() function defined in the System Interfaces  
 105703       volume of POSIX.1-2008 called with a pathname of the current file used as the *path*  
 105704       argument.

105705       If this fails for any reason, *rm* shall write a diagnostic message to standard error, do  
 105706       nothing more with the current file, and go on to any remaining files.

105707       The *rm* utility shall be able to descend to arbitrary depths in a file hierarchy, and shall not fail

105708 due to path length limitations (unless an operand specified by the user exceeds system  
105709 limitations).

#### 105710 OPTIONS

105711 The *rm* utility shall conform to XBD [Section 12.2](#) (on page 215).

105712 The following options shall be supported:

105713 **-f** Do not prompt for confirmation. Do not write diagnostic messages or modify the  
105714 exit status in the case of nonexistent operands. Any previous occurrences of the **-i**  
105715 option shall be ignored.

105716 **-i** Prompt for confirmation as described previously. Any previous occurrences of the  
105717 **-f** option shall be ignored.

105718 **-R** Remove file hierarchies. See the DESCRIPTION.

105719 **-r** Equivalent to **-R**.

#### 105720 OPERANDS

105721 The following operand shall be supported:

105722 *file* A pathname of a directory entry to be removed.

#### 105723 STDIN

105724 The standard input shall be used to read an input line in response to each prompt specified in  
105725 the STDOUT section. Otherwise, the standard input shall not be used.

#### 105726 INPUT FILES

105727 None.

#### 105728 ENVIRONMENT VARIABLES

105729 The following environment variables shall affect the execution of *rm*:

105730 **LANG** Provide a default value for the internationalization variables that are unset or null.  
105731 (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
105732 variables used to determine the values of locale categories.)

105733 **LC\_ALL** If set to a non-empty string value, override the values of all the other  
105734 internationalization variables.

105735 **LC\_COLLATE**  
105736 Determine the locale for the behavior of ranges, equivalence classes, and multi-  
105737 character collating elements used in the extended regular expression defined for  
105738 the **yesexpr** locale keyword in the **LC\_MESSAGES** category.

105739 **LC\_CTYPE** Determine the locale for the interpretation of sequences of bytes of text data as  
105740 characters (for example, single-byte as opposed to multi-byte characters in  
105741 arguments) and the behavior of character classes within regular expressions used  
105742 in the extended regular expression defined for the **yesexpr** locale keyword in the  
105743 **LC\_MESSAGES** category.

105744 **LC\_MESSAGES**  
105745 Determine the locale used to process affirmative responses, and the locale used to  
105746 affect the format and contents of diagnostic messages and prompts written to  
105747 standard error.

105748 **XSIX** **NLSPATH** Determine the location of message catalogs for the processing of **LC\_MESSAGES**.



105749 **ASYNCHRONOUS EVENTS**

105750 Default.

105751 **STDOUT**

105752 Not used.

105753 **STDERR**

105754 Prompts shall be written to standard error under the conditions specified in the DESCRIPTION  
 105755 and OPTIONS sections. The prompts shall contain the *file* pathname, but their format is  
 105756 otherwise unspecified. The standard error also shall be used for diagnostic messages.

105757 **OUTPUT FILES**

105758 None.

105759 **EXTENDED DESCRIPTION**

105760 None.

105761 **EXIT STATUS**

105762 The following exit values shall be returned:

105763 0 Each directory entry was successfully removed, unless its removal was canceled by a non-  
 105764 affirmative response to a prompt for confirmation.

105765 &gt;0 An error occurred.

105766 **CONSEQUENCES OF ERRORS**

105767 Default.

105768 **APPLICATION USAGE**

105769 The *rm* utility is forbidden to remove the names dot and dot-dot in order to avoid the  
 105770 consequences of inadvertently doing something like:

105771 `rm -r .*`

105772 Some implementations do not permit the removal of the last link to an executable binary file that  
 105773 is being executed; see the [EBUSY] error in the *unlink()* function defined in the System Interfaces  
 105774 volume of POSIX.1-2008. Thus, the *rm* utility can fail to remove such files.

105775 The *-i* option causes *rm* to prompt and read the standard input even if the standard input is not  
 105776 a terminal, but in the absence of *-i* the mode prompting is not done when the standard input is  
 105777 not a terminal.

105778 **EXAMPLES**

105779 1. The following command:

105780 `rm a.out core`105781 removes the directory entries: **a.out** and **core**.

105782 2. The following command:

105783 `rm -Rf junk`105784 removes the directory **junk** and all its contents, without prompting.105785 **RATIONALE**

105786 For absolute clarity, paragraphs (2b) and (3) in the DESCRIPTION of *rm* describing the behavior  
 105787 when prompting for confirmation, should be interpreted in the following manner:

```
105788 if ((NOT f_option) AND
105789 ((not_writable AND input_is_terminal) OR i_option))
```

The exact format of the interactive prompts is unspecified. Only the general nature of the contents of prompts are specified because implementations may desire more descriptive prompts than those used on historical implementations. Therefore, an application not using the `-f` option, or using the `-i` option, relies on the system to provide the most suitable dialog directly with the user, based on the behavior specified.

The `-r` option is historical practice on all known systems. The synonym `-R` option is provided for consistency with the other utilities in this volume of POSIX.1-2008 that provide options requesting recursive descent through the file hierarchy.

The behavior of the `-f` option in historical versions of *rm* is inconsistent. In general, along with “forcing” the unlink without prompting for permission, it always causes diagnostic messages to be suppressed and the exit status to be unmodified for nonexistent operands and files that cannot be unlinked. In some versions, however, the `-f` option suppresses usage messages and system errors as well. Suppressing such messages is not a service to either shell scripts or users.

It is less clear that error messages regarding files that cannot be unlinked (removed) should be suppressed. Although this is historical practice, this volume of POSIX.1-2008 does not permit the `-f` option to suppress such messages.

When given the `-r` and `-i` options, historical versions of *rm* prompt the user twice for each directory, once before removing its contents and once before actually attempting to delete the directory entry that names it. This allows the user to “prune” the file hierarchy walk. Historical versions of *rm* were inconsistent in that some did not do the former prompt for directories named on the command line and others had obscure prompting behavior when the `-i` option was specified and the permissions of the file did not permit writing. The POSIX Shell and Utilities *rm* differs little from historic practice, but does require that prompts be consistent. Historical versions of *rm* were also inconsistent in that prompts were done to both standard output and standard error. This volume of POSIX.1-2008 requires that prompts be done to standard error, for consistency with *cp* and *mv*, and to allow historical extensions to *rm* that provide an option to list deleted files on standard output.

The *rm* utility is required to descend to arbitrary depths so that any file hierarchy may be deleted. This means, for example, that the *rm* utility cannot run out of file descriptors during its descent (that is, if the number of file descriptors is limited, *rm* cannot be implemented in the historical fashion where one file descriptor is used per directory level). Also, *rm* is not permitted to fail because of path length restrictions, unless an operand specified by the user is longer than `{PATH_MAX}`.

The *rm* utility removes symbolic links themselves, not the files they refer to, as a consequence of the dependence on the *unlink()* functionality, per the DESCRIPTION. When removing hierarchies with `-r` or `-R`, the prohibition on following symbolic links has to be made explicit.

#### FUTURE DIRECTIONS

None.

#### SEE ALSO

*rmdir*

XBD [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 215)

XSH [remove\(\)](#), [rmdir\(\)](#), [unlink\(\)](#)

#### CHANGE HISTORY

First released in Issue 2.

105834 **Issue 5**

105835 The FUTURE DIRECTIONS section is added.

105836 **Issue 6**

105837 Text is added to clarify actions relating to symbolic links as specified in the IEEE P1003.2b draft  
105838 standard.

105839 **Issue 7**

105840 Austin Group Interpretations 1003.1-2001 #019 and #091 are applied.

105841 Austin Group Interpretation 1003.1-2001 #126 is applied, changing the description of the  
105842 *LC\_MESSAGES* environment variable.

105843 **NAME**105844 `rmdel` — remove a delta from an SCCS file (**DEVELOPMENT**)105845 **SYNOPSIS**105846 XSI `rmdel -r SID file...`105847 **DESCRIPTION**

105848 The *rm~~del~~* utility shall remove the delta specified by the *SID* from each named SCCS file. The  
 105849 delta to be removed shall be the most recent delta in its branch in the delta chain of each named  
 105850 SCCS file. In addition, the application shall ensure that the *SID* specified is not that of a version  
 105851 being edited for the purpose of making a delta; that is, if a *p-file* (see [get](#)) exists for the named  
 105852 SCCS file, the *SID* specified shall not appear in any entry of the *p-file*.

105853 Removal of a delta shall be restricted to:

- 105854 1. The user who made the delta
- 105855 2. The owner of the SCCS file
- 105856 3. The owner of the directory containing the SCCS file

105857 **OPTIONS**105858 The *rm~~del~~* utility shall conform to XBD [Section 12.2](#) (on page 215).

105859 The following option shall be supported:

105860 `-r SID` Specify the SCCS identification string (*SID*) of the delta to be deleted.105861 **OPERANDS**

105862 The following operand shall be supported:

105863 *file* A pathname of an existing SCCS file or a directory. If *file* is a directory, the *rm~~del~~*  
 105864 utility shall behave as though each file in the directory were specified as a named  
 105865 file, except that non-SCCS files (last component of the pathname does not begin  
 105866 with *s*.) and unreadable files shall be silently ignored.

105867 If exactly one *file* operand appears, and it is `'-'`, the standard input shall be read;  
 105868 each line of the standard input is taken to be the name of an SCCS file to be  
 105869 processed. Non-SCCS files and unreadable files shall be silently ignored.

105870 **STDIN**

105871 The standard input shall be a text file used only when the *file* operand is specified as `'-'`. Each  
 105872 line of the text file shall be interpreted as an SCCS pathname.

105873 **INPUT FILES**

105874 The SCCS files shall be files of unspecified format.

105875 **ENVIRONMENT VARIABLES**105876 The following environment variables shall affect the execution of *rm~~del~~*:

105877 *LANG* Provide a default value for the internationalization variables that are unset or null.  
 105878 (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 105879 variables used to determine the values of locale categories.)

105880 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
 105881 internationalization variables.

105882 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
 105883 characters (for example, single-byte as opposed to multi-byte characters in  
 105884 arguments and input files).

105885 *LC\_MESSAGES*  
105886 Determine the locale that should be used to affect the format and contents of  
105887 diagnostic messages written to standard error.

105888 *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

105889 **ASYNCHRONOUS EVENTS**  
105890 Default.

105891 **STDOUT**  
105892 Not used.

105893 **STDERR**  
105894 The standard error shall be used only for diagnostic messages.

105895 **OUTPUT FILES**  
105896 The SCCS files shall be files of unspecified format. During processing of a *file*, a temporary *x-file*,  
105897 as described in *admin*, may be created and deleted; a locking *z-file*, as described in *get*, may be  
105898 created and deleted.

105899 **EXTENDED DESCRIPTION**  
105900 None.

105901 **EXIT STATUS**  
105902 The following exit values shall be returned:  
105903 0 Successful completion.  
105904 >0 An error occurred.

105905 **CONSEQUENCES OF ERRORS**  
105906 Default.

105907 **APPLICATION USAGE**  
105908 None.

105909 **EXAMPLES**  
105910 None.

105911 **RATIONALE**  
105912 None.

105913 **FUTURE DIRECTIONS**  
105914 None.

105915 **SEE ALSO**  
105916 *admin, delta, get, prs*  
105917 XBD [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 215)

105918 **CHANGE HISTORY**  
105919 First released in Issue 2.

105920 **Issue 6**  
105921 The normative text is reworded to avoid use of the term “must” for application requirements.

105922 **NAME**

105923       rmdir — remove directories

105924 **SYNOPSIS**105925       rmdir [-p] *dir*...105926 **DESCRIPTION**105927       The *rmdir* utility shall remove the directory entry specified by each *dir* operand.105928       For each *dir* operand, the *rmdir* utility shall perform actions equivalent to the *rmdir()* function  
105929       called with the *dir* operand as its only argument.105930       Directories shall be processed in the order specified. If a directory and a subdirectory of that  
105931       directory are specified in a single invocation of the *rmdir* utility, the application shall specify the  
105932       subdirectory before the parent directory so that the parent directory will be empty when the  
105933       *rmdir* utility tries to remove it.105934 **OPTIONS**105935       The *rmdir* utility shall conform to XBD [Section 12.2](#) (on page 215).

105936       The following option shall be supported:

105937       **-p**       Remove all directories in a pathname. For each *dir* operand:

- 105938               1. The directory entry it names shall be removed.
- 
- 105939               2. If the
- dir*
- operand includes more than one pathname component, effects
- 
- 105940               equivalent to the following command shall occur:

105941               rmdir -p \$(dirname *dir*)105942 **OPERANDS**

105943       The following operand shall be supported:

105944       *dir*       A pathname of an empty directory to be removed.105945 **STDIN**

105946       Not used.

105947 **INPUT FILES**

105948       None.

105949 **ENVIRONMENT VARIABLES**105950       The following environment variables shall affect the execution of *rmdir*:105951       **LANG**       Provide a default value for the internationalization variables that are unset or null.  
105952       (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
105953       variables used to determine the values of locale categories.)105954       **LC\_ALL**       If set to a non-empty string value, override the values of all the other  
105955       internationalization variables.105956       **LC\_CTYPE**   Determine the locale for the interpretation of sequences of bytes of text data as  
105957       characters (for example, single-byte as opposed to multi-byte characters in  
105958       arguments).105959       **LC\_MESSAGES**105960       Determine the locale that should be used to affect the format and contents of  
105961       diagnostic messages written to standard error.

105962 XSI **NLSPATH** Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

105963 **ASYNCHRONOUS EVENTS**

105964 Default.

105965 **STDOUT**

105966 Not used.

105967 **STDERR**

105968 The standard error shall be used only for diagnostic messages.

105969 **OUTPUT FILES**

105970 None.

105971 **EXTENDED DESCRIPTION**

105972 None.

105973 **EXIT STATUS**

105974 The following exit values shall be returned:

105975 0 Each directory entry specified by a *dir* operand was removed successfully.

105976 >0 An error occurred.

105977 **CONSEQUENCES OF ERRORS**

105978 Default.

105979 **APPLICATION USAGE**

105980 The definition of an empty directory is one that contains, at most, directory entries for dot and dot-dot.

105981

105982 **EXAMPLES**

105983 If a directory **a** in the current directory is empty except it contains a directory **b** and **a/b** is empty

105984 except it contains a directory **c**:

105985 `rmdir -p a/b/c`

105986 removes all three directories.

105987 **RATIONALE**

105988 On historical System V systems, the **-p** option also caused a message to be written to the

105989 standard output. The message indicated whether the whole path was removed or whether part

105990 of the path remained for some reason. The **STDERR** section requires this diagnostic when the

105991 entire path specified by a *dir* operand is not removed, but does not allow the status message

105992 reporting success to be written as a diagnostic.

105993 The *rmdir* utility on System V also included a **-s** option that suppressed the informational

105994 message output by the **-p** option. This option has been omitted because the informational

105995 message is not specified by this volume of POSIX.1-2008.

105996 **FUTURE DIRECTIONS**

105997 None.

105998 **SEE ALSO**

105999 *rm*

106000 XBD [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 215)

106001 XSH *remove()*, *rmdir()*, *unlink()*

|        |                                                                                              |
|--------|----------------------------------------------------------------------------------------------|
| 106002 | <b>CHANGE HISTORY</b>                                                                        |
| 106003 | First released in Issue 2.                                                                   |
| 106004 | <b>Issue 6</b>                                                                               |
| 106005 | The normative text is reworded to avoid use of the term “must” for application requirements. |



106006 **NAME**106007       sact — print current SCCS file-editing activity (**DEVELOPMENT**)106008 **SYNOPSIS**106009 XSI       sact *file*...106010 **DESCRIPTION**

106011       The *sact* utility shall inform the user of any impending deltas to a named SCCS file by writing a  
 106012       list to standard output. This situation occurs when *get -e* has been executed previously without  
 106013       a subsequent execution of *delta*, *unget*, or *scs unedit*.

106014 **OPTIONS**

106015       None.

106016 **OPERANDS**

106017       The following operand shall be supported:

106018       *file*       A pathname of an existing SCCS file or a directory. If *file* is a directory, the *sact*  
 106019       utility shall behave as though each file in the directory were specified as a named  
 106020       file, except that non-SCCS files (last component of the pathname does not begin  
 106021       with **s**.) and unreadable files shall be silently ignored.

106022       If exactly one *file* operand appears, and it is *'-'*, the standard input shall be read;  
 106023       each line of the standard input shall be taken to be the name of an SCCS file to be  
 106024       processed. Non-SCCS files and unreadable files shall be silently ignored.

106025 **STDIN**

106026       The standard input shall be a text file used only when the *file* operand is specified as *'-'*. Each  
 106027       line of the text file shall be interpreted as an SCCS pathname.

106028 **INPUT FILES**

106029       Any SCCS files interrogated are files of an unspecified format.

106030 **ENVIRONMENT VARIABLES**106031       The following environment variables shall affect the execution of *sact*:

106032       *LANG*       Provide a default value for the internationalization variables that are unset or null.  
 106033       (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 106034       variables used to determine the values of locale categories.)

106035       *LC\_ALL*     If set to a non-empty string value, override the values of all the other  
 106036       internationalization variables.

106037       *LC\_CTYPE*   Determine the locale for the interpretation of sequences of bytes of text data as  
 106038       characters (for example, single-byte as opposed to multi-byte characters in  
 106039       arguments and input files).

106040       *LC\_MESSAGES*

106041       Determine the locale that should be used to affect the format and contents of  
 106042       diagnostic messages written to standard error.

106043       *NLSPATH*    Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

106044 **ASYNCHRONOUS EVENTS**

106045       Default.

**106046 STDOUT**

106047 The output for each named file shall consist of a line in the following format:

106048 "%sΔ%sΔ%sΔ%s\n", <SID>, <new SID>, <login>, <date>, <time>

106049 <SID> Specifies the SID of a delta that currently exists in the SCCS file to which changes  
106050 are made to make the new delta.

106051 <new SID> Specifies the SID for the new delta to be created.

106052 <login> Contains the login name of the user who makes the delta (that is, who executed a  
106053 *get* for editing).

106054 <date> Contains the date that *get -e* was executed, in the format used by the *prs :D:* data  
106055 keyword.

106056 <time> Contains the time that *get -e* was executed, in the format used by the *prs :T:* data  
106057 keyword.

106058 If there is more than one named file or if a directory or standard input is named, each pathname  
106059 shall be written before each of the preceding lines:

106060 "\n%s:\n", <pathname>

**106061 STDERR**

106062 The standard error shall be used only for optional informative messages concerning SCCS files  
106063 with no impending deltas, and for diagnostic messages.

**106064 OUTPUT FILES**

106065 None.

**106066 EXTENDED DESCRIPTION**

106067 None.

**106068 EXIT STATUS**

106069 The following exit values shall be returned:

106070 0 Successful completion.

106071 >0 An error occurred.

**106072 CONSEQUENCES OF ERRORS**

106073 Default.

**106074 APPLICATION USAGE**

106075 None.

**106076 EXAMPLES**

106077 None.

**106078 RATIONALE**

106079 None.

**106080 FUTURE DIRECTIONS**

106081 None.

**106082 SEE ALSO**

106083 *delta*, *get*, *sccs*, *unget*

106084 XBD Chapter 8 (on page 173)

106085 **CHANGE HISTORY**  
106086 First released in Issue 2.

106087 **NAME**106088 `sccs` — front end for the SCCS subsystem (**DEVELOPMENT**)106089 **SYNOPSIS**106090 XSI `sccs [-r] [-d path] [-p path] command [options...] [operands...]`106091 **DESCRIPTION**106092 The *sccs* utility is a front end to the SCCS programs. It also includes the capability to run set-  
106093 user-id to another user to provide additional protection.106094 The *sccs* utility shall invoke the specified *command* with the specified *options* and *operands*. By  
106095 default, each of the *operands* shall be modified by prefixing it with the string "SCCS/s.". 106096 The *command* can be the name of one of the SCCS utilities in this volume of POSIX.1-2008 (*admin*,  
106097 *delta*, *get*, *prs*, *rmDEL*, *sact*, *unget*, *val*, or *what*) or one of the pseudo-utilities listed in the  
106098 EXTENDED DESCRIPTION section.106099 **OPTIONS**106100 The *sccs* utility shall conform to XBD [Section 12.2](#) (on page 215), except that *options* operands are  
106101 actually options to be passed to the utility named by *command*. When the portion of the  
106102 command:106103 `command [options ...] [operands ...]`106104 is considered, all of the pseudo-utilities used as *command* shall support the Utility Syntax  
106105 Guidelines. Any of the other SCCS utilities that can be invoked in this manner support the  
106106 Guidelines to the extent indicated by their individual OPTIONS sections.106107 The following options shall be supported preceding the *command* operand:106108 **-d *path*** A pathname of a directory to be used as a root directory for the SCCS files. The  
106109 default shall be the current directory. The **-d** option shall take precedence over the  
106110 *PROJECTDIR* variable. See **-p**.106111 **-p *path*** A pathname of a directory in which the SCCS files are located. The default shall be  
106112 the **SCCS** directory.106113 The **-p** option differs from the **-d** option in that the **-d** option-argument shall be  
106114 prefixed to the entire pathname and the **-p** option-argument shall be inserted  
106115 before the final component of the pathname. For example:106116 `sccs -d /x -p y get a/b`

106117 converts to:

106118 `get /x/a/y/s.b`

106119 This allows the creation of aliases such as:

106120 `alias syssccs="sccs -d /usr/src"`

106121 which is used as:

106122 `syssccs get cmd/who.c`106123 **-r** Invoke *command* with the real user ID of the process, not any effective user ID that  
106124 the *sccs* utility is set to. Certain commands (*admin*, **check**, **clean**, **diffs**, **info**, *rmDEL*,  
106125 and **tell**) cannot be run set-user-ID by all users, since this would allow anyone to  
106126 change the authorizations. These commands are always run as the real user.

106127 **OPERANDS**

106128 The following operands shall be supported:

106129 *command* An SCCS utility name or the name of one of the pseudo-utilities listed in the  
 106130 EXTENDED DESCRIPTION section.

106131 *options* An option or option-argument to be passed to *command*.

106132 *operands* An operand to be passed to *command*.

106133 **STDIN**106134 See the utility description for the specified *command*.106135 **INPUT FILES**106136 See the utility description for the specified *command*.106137 **ENVIRONMENT VARIABLES**106138 The following environment variables shall affect the execution of *sccs*:

106139 *LANG* Provide a default value for the internationalization variables that are unset or null.  
 106140 (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 106141 variables used to determine the values of locale categories.)

106142 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
 106143 internationalization variables.

106144 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
 106145 characters (for example, single-byte as opposed to multi-byte characters in  
 106146 arguments and input files).

106147 *LC\_MESSAGES*

106148 Determine the locale that should be used to affect the format and contents of  
 106149 diagnostic messages written to standard error.

106150 *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

106151 *PROJECTDIR*

106152 Provide a default value for the *-d path* option. If the value of *PROJECTDIR* begins  
 106153 with a <slash>, it shall be considered an absolute pathname; otherwise, the value  
 106154 of *PROJECTDIR* is treated as a user name and that user's initial working directory  
 106155 shall be examined for a subdirectory **src** or **source**. If such a directory is found, it  
 106156 shall be used. Otherwise, the value shall be used as a relative pathname.

106157 Additional environment variable effects may be found in the utility description for the specified  
 106158 *command*.

106159 **ASYNCHRONOUS EVENTS**

106160 Default.

106161 **STDOUT**106162 See the utility description for the specified *command*.106163 **STDERR**106164 See the utility description for the specified *command*.106165 **OUTPUT FILES**106166 See the utility description for the specified *command*.

## 106167 EXTENDED DESCRIPTION

106168 The following pseudo-utilities shall be supported as *command* operands. All options referred to  
 106169 in the following list are values given in the *options* operands following *command*.

- 106170 **check** Equivalent to **info**, except that nothing shall be printed if nothing is being edited, and a  
 106171 non-zero exit status shall be returned if anything is being edited. The intent is to have  
 106172 this included in an “install” entry in a makefile to ensure that everything is included  
 106173 into the SCCS file before a version is installed.
- 106174 **clean** Remove everything from the current directory that can be recreated from SCCS files,  
 106175 but do not remove any files being edited. If the **-b** option is given, branches shall be  
 106176 ignored in the determination of whether they are being edited; this is dangerous if  
 106177 branches are kept in the same directory.
- 106178 **create** Create an SCCS file, taking the initial contents from the file of the same name. Any  
 106179 options to *admin* are accepted. If the creation is successful, the original files shall be  
 106180 renamed by prefixing the basenames with a comma. These renamed files should be  
 106181 removed after it has been verified that the SCCS files have been created successfully.
- 106182 **delget** Perform a *delta* on the named files and then *get* new versions. The new versions shall  
 106183 have ID keywords expanded and shall not be editable. Any **-m**, **-p**, **-r**, **-s**, and **-y**  
 106184 options shall be passed to *delta*, and any **-b**, **-c**, **-e**, **-i**, **-k**, **-l**, **-s**, and **-x** options shall be  
 106185 passed to *get*.
- 106186 **deledit** Equivalent to **delget**, except that the *get* phase shall include the **-e** option. This option is  
 106187 useful for making a checkpoint of the current editing phase. The same options shall be  
 106188 passed to *delta* as described above, and all the options listed for *get* above except **-e**  
 106189 shall be passed to **edit**.
- 106190 **diffs** Write a difference listing between the current version of the files checked out for editing  
 106191 and the versions in SCCS format. Any **-r**, **-c**, **-i**, **-x**, and **-t** options shall be passed to  
 106192 *get*; any **-l**, **-s**, **-e**, **-f**, **-h**, and **-b** options shall be passed to *diff*. A **-C** option shall be  
 106193 passed to *diff* as **-c**.
- 106194 **edit** Equivalent to *get -e*.
- 106195 **fix** Remove the named delta, but leave a copy of the delta with the changes that were in it.  
 106196 It is useful for fixing small compiler bugs, and so on. The application shall ensure that it  
 106197 is followed by a **-r** *SID* option. Since **fix** does not leave audit trails, it should be used  
 106198 carefully.
- 106199 **info** Write a listing of all files being edited. If the **-b** option is given, branches (that is, SIDs  
 106200 with two or fewer components) shall be ignored. If a **-u** *user* option is given, then only  
 106201 files being edited by the named user shall be listed. A **-U** option shall be equivalent to  
 106202 **-u***<current user>*.
- 106203 **print** Write out verbose information about the named files, equivalent to *sccs prs*.
- 106204 **tell** Write a <newline>-separated list of the files being edited to standard output. Takes the  
 106205 **-b**, **-u**, and **-U** options like **info** and **check**.
- 106206 **unedit** This is the opposite of an **edit** or a *get -e*. It should be used with caution, since any  
 106207 changes made since the *get* are lost.

106208 **EXIT STATUS**

106209 The following exit values shall be returned:

106210 0 Successful completion.

106211 &gt;0 An error occurred.

106212 **CONSEQUENCES OF ERRORS**

106213 Default.

106214 **APPLICATION USAGE**

106215 Many of the SCCS utilities take directory names as operands as well as specific filenames. The  
 106216 pseudo-utilities supported by *sccs* are not described as having this capability, but are not  
 106217 prohibited from doing so.

106218 **EXAMPLES**

106219 1. To get a file for editing, edit it and produce a new delta:

106220 `sccs get -e file.c`106221 `ex file.c`106222 `sccs delta file.c`

106223 2. To get a file from another directory:

106224 `sccs -p /usr/src/sccs/s. get cc.c`

106225 or:

106226 `sccs get /usr/src/sccs/s.cc.c`

106227 3. To make a delta of a large number of files in the current directory:

106228 `sccs delta *.c`

106229 4. To get a list of files being edited that are not on branches:

106230 `sccs info -b`

106231 5. To delta everything being edited by the current user:

106232 `sccs delta $(sccs tell -U)`

106233 6. In a makefile, to get source files from an SCCS file if it does not already exist:

106234 `SRCS = <list of source files>`106235 `$(SRCS) :`106236 `sccs get $(REL) $@`106237 **RATIONALE**

106238 *sccs* and its associated utilities are part of the XSI Development Utilities option within the XSI  
 106239 option.

106240 SCCS is an abbreviation for Source Code Control System. It is a maintenance and enhancement  
 106241 tracking tool. When a file is put under SCCS, the source code control system maintains the file  
 106242 and, when changes are made, identifies and stores them in the file with the original source code  
 106243 and/or documentation. As other changes are made, they too are identified and retained in the  
 106244 file.

106245 Retrieval of the original and any set of changes is possible. Any version of the file as it develops  
 106246 can be reconstructed for inspection or additional modification. History data can be stored with  
 106247 each version, documenting why the changes were made, who made them, and when they were  
 106248 made.

106249 **FUTURE DIRECTIONS**

106250 None.

106251 **SEE ALSO**106252 *admin, delta, get, make, prs, rmdel, sact, unget, val, what*106253 XBD [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 215)106254 **CHANGE HISTORY**

106255 First released in Issue 4.

106256 **Issue 6**

106257 In the ENVIRONMENT VARIABLES section, the *PROJECTDIR* description is updated from  
106258 “otherwise, the home directory of a user of that name is examined” to “otherwise, the value of  
106259 *PROJECTDIR* is treated as a user name and that user’s initial working directory is examined”.

106260 The normative text is reworded to avoid use of the term “must” for application requirements.

106261 **Issue 7**

106262 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.



106263 **NAME**106264 `sed` — stream editor106265 **SYNOPSIS**106266 `sed [-n] script [file...]`106267 `sed [-n] -e script [-e script]... [-f script_file]... [file...]`106268 `sed [-n] [-e script]... -f script_file [-f script_file]... [file...]`106269 **DESCRIPTION**

106270 The *sed* utility is a stream editor that shall read one or more text files, make editing changes  
 106271 according to a script of editing commands, and write the results to standard output. The script  
 106272 shall be obtained from either the *script* operand string or a combination of the option-arguments  
 106273 from the `-e script` and `-f script_file` options.

106274 **OPTIONS**

106275 The *sed* utility shall conform to XBD [Section 12.2](#) (on page 215), except that the order of  
 106276 presentation of the `-e` and `-f` options is significant.

106277 The following options shall be supported:

106278 `-e script` Add the editing commands specified by the *script* option-argument to the end of  
 106279 the script of editing commands.

106280 `-f script_file` Add the editing commands in the file *script\_file* to the end of the script of editing  
 106281 commands.

106282 `-n` Suppress the default output (in which each line, after it is examined for editing, is  
 106283 written to standard output). Only lines explicitly selected for output are written.

106284 If any `-e` or `-f` options are specified, the script of editing commands shall initially be empty. The  
 106285 commands specified by each `-e` or `-f` option shall be added to the script in the order specified.  
 106286 When each addition is made, if the previous addition (if any) was from a `-e` option, a <newline>  
 106287 shall be inserted before the new addition. The resulting script shall have the same properties as  
 106288 the *script* operand, described in the OPERANDS section.

106289 **OPERANDS**

106290 The following operands shall be supported:

106291 *file* A pathname of a file whose contents are read and edited. If multiple *file* operands  
 106292 are specified, the named files shall be read in the order specified and the  
 106293 concatenation shall be edited. If no *file* operands are specified, the standard input  
 106294 shall be used.

106295 *script* A string to be used as the script of editing commands. The application shall not  
 106296 present a *script* that violates the restrictions of a text file except that the final  
 106297 character need not be a <newline>.

106298 **STDIN**

106299 The standard input shall be used if no *file* operands are specified, and shall be used if a *file*  
 106300 operand is `'-'` and the implementation treats the `'-'` as meaning standard input. Otherwise,  
 106301 the standard input shall not be used. See the INPUT FILES section.

106302 **INPUT FILES**

106303 The input files shall be text files. The *script\_files* named by the `-f` option shall consist of editing  
 106304 commands.

106305 **ENVIRONMENT VARIABLES**

106306 The following environment variables shall affect the execution of *sed*:

106307 *LANG* Provide a default value for the internationalization variables that are unset or null.  
 106308 (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 106309 variables used to determine the values of locale categories.)

106310 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
 106311 internationalization variables.

106312 *LC\_COLLATE*  
 106313 Determine the locale for the behavior of ranges, equivalence classes, and multi-  
 106314 character collating elements within regular expressions.

106315 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
 106316 characters (for example, single-byte as opposed to multi-byte characters in  
 106317 arguments and input files), and the behavior of character classes within regular  
 106318 expressions.

106319 *LC\_MESSAGES*  
 106320 Determine the locale that should be used to affect the format and contents of  
 106321 diagnostic messages written to standard error.

106322 XSI *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

106323 **ASYNCHRONOUS EVENTS**

106324 Default.

106325 **STDOUT**

106326 The input files shall be written to standard output, with the editing commands specified in the  
 106327 script applied. If the *-n* option is specified, only those input lines selected by the script shall be  
 106328 written to standard output.

106329 **STDERR**

106330 The standard error shall be used only for diagnostic messages.

106331 **OUTPUT FILES**

106332 The output files shall be text files whose formats are dependent on the editing commands given.

106333 **EXTENDED DESCRIPTION**

106334 The *script* shall consist of editing commands of the following form:

106335 `[address[, address]] function`

106336 where *function* represents a single-character command verb from the list in [Editing Commands](#)  
 106337 [in sed](#) (on page 3181), followed by any applicable arguments.

106338 The command can be preceded by <blank> characters and/or <semicolon> characters. The  
 106339 function can be preceded by <blank> characters. These optional characters shall have no effect.

106340 In default operation, *sed* cyclically shall append a line of input, less its terminating <newline>  
 106341 character, into the pattern space. Reading from input shall be skipped if a <newline> was in the  
 106342 pattern space prior to a **D** command ending the previous cycle. The *sed* utility shall then apply in  
 106343 sequence all commands whose addresses select that pattern space, until a command starts the  
 106344 next cycle or quits. If no commands explicitly started a new cycle, then at the end of the script  
 106345 the pattern space shall be copied to standard output (except when *-n* is specified) and the  
 106346 pattern space shall be deleted. Whenever the pattern space is written to standard output or a  
 106347 named file, *sed* shall immediately follow it with a <newline>.

106348 Some of the editing commands use a hold space to save all or part of the pattern space for

106349 subsequent retrieval. The pattern and hold spaces shall each be able to hold at least 8 192 bytes.

## 106350 **Addresses in sed**

106351 An address is either a decimal number that counts input lines cumulatively across files, a '\$' character that addresses the last line of input, or a context address (which consists of a BRE, as described in [Regular Expressions in sed](#), preceded and followed by a delimiter, usually a <slash>).

106355 An editing command with no addresses shall select every pattern space.

106356 An editing command with one address shall select each pattern space that matches the address.

106357 An editing command with two addresses shall select the inclusive range from the first pattern space that matches the first address through the next pattern space that matches the second. (If the second address is a number less than or equal to the line number first selected, only one line shall be selected.) Starting at the first line following the selected range, *sed* shall look again for the first address. Thereafter, the process shall be repeated. Omitting either or both of the address components in the following form produces undefined results:

106363 `[address[, address]]`

## 106364 **Regular Expressions in sed**

106365 The *sed* utility shall support the BREs described in XBD [Section 9.3](#) (on page 183), with the following additions:

- 106367 • In a context address, the construction "`\cBREc`", where *c* is any character other than <backslash> or <newline>, shall be identical to "`/BRE/`". If the character designated by *c* appears following a <backslash>, then it shall be considered to be that literal character, which shall not terminate the BRE. For example, in the context address "`\xabc\xdefx`", the second *x* stands for itself, so that the BRE is "`abcxdef`".
- 106372 • The escape sequence '`\n`' shall match a <newline> embedded in the pattern space. A literal <newline> shall not be used in the BRE of a context address or in the substitute function.
- 106375 • If an RE is empty (that is, no pattern is specified) *sed* shall behave as if the last RE used in the last command applied (either as an address or as part of a substitute command) was specified.

## 106378 **Editing Commands in sed**

106379 In the following list of editing commands, the maximum number of permissible addresses for each function is indicated by `[0addr]`, `[1addr]`, or `[2addr]`, representing zero, one, or two addresses.

106382 The argument *text* shall consist of one or more lines. Each embedded <newline> in the text shall be preceded by a <backslash>. Other <backslash> characters in text shall be removed, and the following character shall be treated literally.

106385 The **r** and **w** command verbs, and the *w* flag to the **s** command, take an *rfile* (or *wfile*) parameter, separated from the command verb letter or flag by one or more <blank> characters; implementations may allow zero separation as an extension.

106388 The argument *rfile* or the argument *wfile* shall terminate the editing command. Each *wfile* shall be created before processing begins. Implementations shall support at least ten *wfile* arguments in the script; the actual number (greater than or equal to 10) that is supported by the

implementation is unspecified. The use of the *wfile* parameter shall cause that file to be initially created, if it does not exist, or shall replace the contents of an existing file.

The **b**, **r**, **s**, **t**, **w**, **y**, and **:** command verbs shall accept additional arguments. The following synopses indicate which arguments shall be separated from the command verbs by a single <space>.

The **a** and **r** commands schedule text for later output. The text specified for the **a** command, and the contents of the file specified for the **r** command, shall be written to standard output just before the next attempt to fetch a line of input when executing the **N** or **n** commands, or when reaching the end of the script. If written when reaching the end of the script, and the **-n** option was not specified, the text shall be written after copying the pattern space to standard output. The contents of the file specified for the **r** command shall be as of the time the output is written, not the time the **r** command is applied. The text shall be output in the order in which the **a** and **r** commands were applied to the input.

Command verbs other than **{**, **a**, **b**, **c**, **i**, **r**, **t**, **w**, **:**, and **#** can be followed by a <semicolon>, optional <blank> characters, and another command verb. However, when the **s** command verb is used with the *w* flag, following it with another command in this manner produces undefined results.

A function can be preceded by one or more **' ! '** characters, in which case the function shall be applied if the addresses do not select the pattern space. Zero or more <blank> characters shall be accepted before the first **' ! '** character. It is unspecified whether <blank> characters can follow a **' ! '** character, and conforming applications shall not follow a **' ! '** character with <blank> characters.

[2addr] {editing command  
editing command  
...  
}  
Execute a list of *sed* editing commands only when the pattern space is selected. The list of *sed* editing commands shall be surrounded by braces and separated by <newline> characters, and conform to the following rules. The braces can be preceded or followed by <blank> characters. The editing commands can be preceded by <blank> characters, but shall not be followed by <blank> characters. The <right-brace> shall be preceded by a <newline> and can be preceded or followed by <blank> characters.

[1addr]a\  
text  
Write text to standard output as described previously.

[2addr]b [label]  
Branch to the **:** function bearing the *label*. If *label* is not specified, branch to the end of the script. The implementation shall support *labels* recognized as unique up to at least 8 characters; the actual length (greater than or equal to 8) that shall be supported by the implementation is unspecified. It is unspecified whether exceeding a label length causes an error or a silent truncation.

[2addr]c\  
text  
Delete the pattern space. With a 0 or 1 address or at the end of a 2-address range, place *text* on the output and start the next cycle.

[2addr]d  
Delete the pattern space and start the next cycle.

[2addr]D  
If the pattern space contains no <newline>, delete the pattern space and start a normal new cycle as if the **d** command was issued. Otherwise, delete the initial segment of the pattern space through the first <newline>, and start the next cycle

|        |                                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
|--------|---------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 106438 |                                       | with the resultant pattern space and without reading any new input.                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 106439 | <b>[2addr]g</b>                       | Replace the contents of the pattern space by the contents of the hold space.                                                                                                                                                                                                                                                                                                                                                                                                                           |
| 106440 | <b>[2addr]G</b>                       | Append to the pattern space a <newline> followed by the contents of the hold space.                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 106441 |                                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 106442 | <b>[2addr]h</b>                       | Replace the contents of the hold space with the contents of the pattern space.                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 106443 | <b>[2addr]H</b>                       | Append to the hold space a <newline> followed by the contents of the pattern space.                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 106444 |                                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 106445 | <b>[1addr]i\</b>                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 106446 | <i>text</i>                           | Write <i>text</i> to standard output.                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 106447 | <b>[2addr]l</b>                       | (The letter ell.) Write the pattern space to standard output in a visually unambiguous form. The characters listed in XBD <a href="#">Table 5-1</a> (on page 121) ('\\', '\a', '\b', '\f', '\r', '\t', '\v') shall be written as the corresponding escape sequence; the '\n' in that table is not applicable. Non-printable characters not in that table shall be written as one three-digit octal number (with a preceding <backslash>) for each byte in the character (most significant byte first). |
| 106448 |                                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 106449 |                                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 106450 |                                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 106451 |                                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 106452 |                                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 106453 |                                       | Long lines shall be folded, with the point of folding indicated by writing a <backslash> followed by a <newline>; the length at which folding occurs is unspecified, but should be appropriate for the output device. The end of each line shall be marked with a ' \$ '.                                                                                                                                                                                                                              |
| 106454 |                                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 106455 |                                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 106456 |                                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 106457 | <b>[2addr]n</b>                       | Write the pattern space to standard output if the default output has not been suppressed, and replace the pattern space with the next line of input, less its terminating <newline>.                                                                                                                                                                                                                                                                                                                   |
| 106458 |                                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 106459 |                                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 106460 |                                       | If no next line of input is available, the <b>n</b> command verb shall branch to the end of the script and quit without starting a new cycle.                                                                                                                                                                                                                                                                                                                                                          |
| 106461 |                                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 106462 | <b>[2addr]N</b>                       | Append the next line of input, less its terminating <newline>, to the pattern space, using an embedded <newline> to separate the appended material from the original material. Note that the current line number changes.                                                                                                                                                                                                                                                                              |
| 106463 |                                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 106464 |                                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 106465 |                                       | If no next line of input is available, the <b>N</b> command verb shall branch to the end of the script and quit without starting a new cycle or copying the pattern space to standard output.                                                                                                                                                                                                                                                                                                          |
| 106466 |                                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 106467 |                                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 106468 | <b>[2addr]p</b>                       | Write the pattern space to standard output.                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 106469 | <b>[2addr]P</b>                       | Write the pattern space, up to the first <newline>, to standard output.                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 106470 | <b>[1addr]q</b>                       | Branch to the end of the script and quit without starting a new cycle.                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| 106471 | <b>[1addr]r <i>rfile</i></b>          | Copy the contents of <i>rfile</i> to standard output as described previously. If <i>rfile</i> does not exist or cannot be read, it shall be treated as if it were an empty file, causing no error condition.                                                                                                                                                                                                                                                                                           |
| 106472 |                                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 106473 |                                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 106474 | <b>[2addr]s/BRE/replacement/flags</b> |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 106475 |                                       | Substitute the replacement string for instances of the BRE in the pattern space. Any character other than <backslash> or <newline> can be used instead of a <slash> to delimit the BRE and the replacement. Within the BRE and the replacement, the BRE delimiter itself can be used as a literal character if it is preceded by a <backslash>.                                                                                                                                                        |
| 106476 |                                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 106477 |                                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 106478 |                                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 106479 |                                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 106480 |                                       | The replacement string shall be scanned from beginning to end. An <ampersand>                                                                                                                                                                                                                                                                                                                                                                                                                          |

106481 (' & ') appearing in the replacement shall be replaced by the string matching the  
 106482 BRE. The special meaning of ' & ' in this context can be suppressed by preceding it  
 106483 by a <backslash>. The characters "\n", where *n* is a digit, shall be replaced by the  
 106484 text matched by the corresponding back-reference expression. If the corresponding  
 106485 back-reference expression does not match, then the characters "\n" shall be  
 106486 replaced by the empty string. The special meaning of "\n" where *n* is a digit in  
 106487 this context, can be suppressed by preceding it by a <backslash>. For each other  
 106488 <backslash> encountered, the following character shall lose its special meaning (if  
 106489 any). The meaning of a <backslash> immediately followed by any character other  
 106490 than ' & ', <backslash>, a digit, or the delimiter character used for this command, is  
 106491 unspecified.

106492 A line can be split by substituting a <newline> into it. The application shall escape  
 106493 the <newline> in the replacement by preceding it by a <backslash>. A substitution  
 106494 shall be considered to have been performed even if the replacement string is  
 106495 identical to the string that it replaces. Any <backslash> used to alter the default  
 106496 meaning of a subsequent character shall be discarded from the BRE or the  
 106497 replacement before evaluating the BRE or using the replacement.

106498 The value of *flags* shall be zero or more of:

106499 *n* Substitute for the *n*th occurrence only of the BRE found within the  
 106500 pattern space.

106501 *g* Globally substitute for all non-overlapping instances of the BRE  
 106502 rather than just the first one. If both *g* and *n* are specified, the results  
 106503 are unspecified.

106504 *p* Write the pattern space to standard output if a replacement was  
 106505 made.

106506 *w wfile* Write. Append the pattern space to *wfile* if a replacement was made.  
 106507 A conforming application shall precede the *wfile* argument with one  
 106508 or more <blank> characters. If the *w* flag is not the last flag value  
 106509 given in a concatenation of multiple flag values, the results are  
 106510 undefined.

106511 [2addr]**t** [*label*]  
 106512 Test. Branch to the : command verb bearing the *label* if any substitutions have been  
 106513 made since the most recent reading of an input line or execution of a **t**. If *label* is  
 106514 not specified, branch to the end of the script.

106515 [2addr]**w** *wfile*  
 106516 Append (write) the pattern space to *wfile*.

106517 [2addr]**x** Exchange the contents of the pattern and hold spaces.

106518 [2addr]**y**/*string1*/*string2*/  
 106519 Replace all occurrences of characters in *string1* with the corresponding characters  
 106520 in *string2*. If a <backslash> followed by an ' n ' appear in *string1* or *string2*, the two  
 106521 characters shall be handled as a single <newline>. If the number of characters in  
 106522 *string1* and *string2* are not equal, or if any of the characters in *string1* appear more  
 106523 than once, the results are undefined. Any character other than <backslash> or  
 106524 <newline> can be used instead of <slash> to delimit the strings. If the delimiter is  
 106525 not ' n ', within *string1* and *string2*, the delimiter itself can be used as a literal  
 106526 character if it is preceded by a <backslash>. If a <backslash> character is  
 106527 immediately followed by a <backslash> character in *string1* or *string2*, the two

106528 <backslash> characters shall be counted as a single literal <backslash> character.  
 106529 The meaning of a <backslash> followed by any character that is not 'n', a  
 106530 <backslash>, or the delimiter character is undefined.

106531 [0addr]:label Do nothing. This command bears a *label* to which the **b** and **t** commands branch.

106532 [1addr]= Write the following to standard output:

106533 "%d\n", <current line number>

106534 [0addr] Ignore this empty command.

106535 [0addr]# Ignore the '#' and the remainder of the line (treat them as a comment), with the  
 106536 single exception that if the first two characters in the script are "#n", the default  
 106537 output shall be suppressed; this shall be the equivalent of specifying **-n** on the  
 106538 command line.

## 106539 EXIT STATUS

106540 The following exit values shall be returned:

106541 0 Successful completion.

106542 >0 An error occurred.

## 106543 CONSEQUENCES OF ERRORS

106544 Default.

## 106545 APPLICATION USAGE

106546 Regular expressions match entire strings, not just individual lines, but a <newline> is matched  
 106547 by '\n' in a *sed* RE; a <newline> is not allowed by the general definition of regular expression  
 106548 in POSIX.1-2008. Also note that '\n' cannot be used to match a <newline> at the end of an  
 106549 arbitrary input line; <newline> characters appear in the pattern space as a result of the **N** editing  
 106550 command.

## 106551 EXAMPLES

106552 This *sed* script simulates the BSD *cat -s* command, squeezing excess empty lines from standard  
 106553 input.

```
106554 sed -n '
106555 # Write non-empty lines.
106556 ./ {
106557 p
106558 d
106559 }
106560 # Write a single empty line, then look for more empty lines.
106561 /^$/ p
106562 # Get next line, discard the held <newline> (empty line),
106563 # and look for more empty lines.
106564 :Empty
106565 /^$/ {
106566 N
106567 s/./ /
106568 b Empty
106569 }
106570 # Write the non-empty line before going back to search
106571 # for the first in a set of empty lines.
106572 p
106573 '
```

The following *sed* command is a much simpler method of squeezing empty lines, although it is not quite the same as *cat -s* since it removes any initial empty lines:

```
sed -n '/./,/^$/p'
```

#### RATIONALE

This volume of POSIX.1-2008 requires implementations to support at least ten distinct *wfiles*, matching historical practice on many implementations. Implementations are encouraged to support more, but conforming applications should not exceed this limit.

The exit status codes specified here are different from those in System V. System V returns 2 for garbled *sed* commands, but returns zero with its usage message or if the input file could not be opened. The standard developers considered this to be a bug.

The manner in which the *l* command writes non-printable characters was changed to avoid the historical backspace-overstrike method, and other requirements to achieve unambiguous output were added. See the RATIONALE for *ed* for details of the format chosen, which is the same as that chosen for *sed*.

This volume of POSIX.1-2008 requires implementations to provide pattern and hold spaces of at least 8 192 bytes, larger than the 4 000 bytes spaces used by some historical implementations, but less than the 20 480 bytes limit used in an early proposal. Implementations are encouraged to allocate dynamically larger pattern and hold spaces as needed.

The requirements for acceptance of <blank> and <space> characters in command lines has been made more explicit than in early proposals to describe clearly the historical practice and to remove confusion about the phrase “protect initial blanks [*sic*] and tabs from the stripping that is done on every script line” that appears in much of the historical documentation of the *sed* utility description of text. (Not all implementations are known to have stripped <blank> characters from text lines, although they all have allowed leading <blank> characters preceding the address on a command line.)

The treatment of ‘#’ comments differs from the SVID which only allows a comment as the first line of the script, but matches BSD-derived implementations. The comment character is treated as a command, and it has the same properties in terms of being accepted with leading <blank> characters; the BSD implementation has historically supported this.

Early proposals required that a *script\_file* have at least one non-comment line. Some historical implementations have behaved in unexpected ways if this were not the case. The standard developers considered that this was incorrect behavior and that application developers should not have to avoid this feature. A correct implementation of this volume of POSIX.1-2008 shall permit *script\_files* that consist only of comment lines.

Early proposals indicated that if *-e* and *-f* options were intermixed, all *-e* options were processed before any *-f* options. This has been changed to process them in the order presented because it matches historical practice and is more intuitive.

The treatment of the *p* flag to the *s* command differs between System V and BSD-based systems when the default output is suppressed. In the two examples:

```
echo a | sed 's/a/A/p'
```

```
echo a | sed -n 's/a/A/p'
```

this volume of POSIX.1-2008, BSD, System V documentation, and the SVID indicate that the first example should write two lines with *A*, whereas the second should write one. Some System V systems write the *A* only once in both examples because the *p* flag is ignored if the *-n* option is not specified.



This is a case of a diametrical difference between systems that could not be reconciled through the compromise of declaring the behavior to be unspecified. The SVID/BSD/System V documentation behavior was adopted for this volume of POSIX.1-2008 because:

- No known documentation for any historic system describes the interaction between the **p** flag and the **-n** option.
- The selected behavior is more correct as there is no technical justification for any interaction between the **p** flag and the **-n** option. A relationship between **-n** and the **p** flag might imply that they are only used together, but this ignores valid scripts that interrupt the cyclical nature of the processing through the use of the **D**, **d**, **q**, or branching commands. Such scripts rely on the **p** suffix to write the pattern space because they do not make use of the default output at the “bottom” of the script.
- Because the **-n** option makes the **p** flag unnecessary, any interaction would only be useful if *sed* scripts were written to run both with and without the **-n** option. This is believed to be unlikely. It is even more unlikely that programmers have coded the **p** flag expecting it to be unnecessary. Because the interaction was not documented, the likelihood of a programmer discovering the interaction and depending on it is further decreased.
- Finally, scripts that break under the specified behavior produce too much output instead of too little, which is easier to diagnose and correct.

The form of the substitute command that uses the **n** suffix was limited to the first 512 matches in an early proposal. This limit has been removed because there is no reason an editor processing lines of {LINE\_MAX} length should have this restriction. The command **s/a/A/2047** should be able to substitute the 2 047th occurrence of **a** on a line.

The **b**, **t**, and **:** commands are documented to ignore leading white space, but no mention is made of trailing white space. Historical implementations of *sed* assigned different locations to the labels **'x'** and **"x "**. This is not useful, and leads to subtle programming errors, but it is historical practice, and changing it could theoretically break working scripts. Implementors are encouraged to provide warning messages about labels that are never used or jumps to labels that do not exist.

Historically, the *sed* **!** and **}** editing commands did not permit multiple commands on a single line using a <semicolon> as a command delimiter. Implementations are permitted, but not required, to support this extension.

Earlier versions of this standard allowed for implementations with bytes other than eight bits, but this has been modified in this version.

## FUTURE DIRECTIONS

None.

## SEE ALSO

*awk*, *ed*, *grep*

XBD [Table 5-1](#) (on page 121), [Chapter 8](#) (on page 173), [Section 9.3](#) (on page 183), [Section 12.2](#) (on page 215)

## CHANGE HISTORY

First released in Issue 2.

## Issue 5

The FUTURE DIRECTIONS section is added.

106662 **Issue 6**

106663 The following new requirements on POSIX implementations derive from alignment with the  
106664 Single UNIX Specification:

- 106665 • Implementations are required to support at least ten *wfile* arguments in an editing  
106666 command.

106667 The EXTENDED DESCRIPTION is changed to align with the IEEE P1003.2b draft standard.

106668 IEEE PASC Interpretation 1003.2 #190 is applied.

106669 IEEE PASC Interpretation 1003.2 #203 is applied, clarifying the meaning of the  
106670 <backslash>-escape sequences in a replacement string for a BRE.

106671 IEEE Std 1003.1-2001/Cor 2-2004, item XCU/TC2/D6/28 is applied, removing text describing  
106672 behavior on systems with bytes consisting of more than eight bits.

106673 IEEE Std 1003.1-2001/Cor 2-2004, item XCU/TC2/D6/29 is applied, making an editorial  
106674 correction within the Editing Commands in *sed* section.

106675 **Issue 7**

106676 Austin Group Interpretations 1003.1-2001 #006, #036, and #092 are applied.

106677 SD5-XCU-ERN-97 and SD5-XCU-ERN-123 are applied, updating the SYNOPSIS.

106678 A second example is added.

106679 POSIX.1-2008, Technical Corrigendum 1, XCU/TC1-2008/0133 [262], XCU/TC1-2008/0134 +  
106680 [282,431], XCU/TC1-2008/0135 [269], and XCU/TC1-2008/0136 [282,431] are applied.

106681 **NAME**

106682 sh — shell, the standard command language interpreter

106683 **SYNOPSIS**

```
106684 sh [-abCefhimnuvx] [-o option]... [+abCefhimnuvx] [+o option]...
106685 [command_file [argument...]]
106686 sh -c [-abCefhimnuvx] [-o option]... [+abCefhimnuvx] [+o option]...
106687 command_string [command_name [argument...]]
106688 sh -s [-abCefhimnuvx] [-o option]... [+abCefhimnuvx] [+o option]...
106689 [argument...]
```

106690 **DESCRIPTION**

106691 The *sh* utility is a command language interpreter that shall execute commands read from a  
 106692 command line string, the standard input, or a specified file. The application shall ensure that the  
 106693 commands to be executed are expressed in the language described in [Chapter 2](#) (on page 2319).

106694 Pathname expansion shall not fail due to the size of a file.

106695 Shell input and output redirections have an implementation-defined offset maximum that is  
 106696 established in the open file description.

106697 **OPTIONS**

106698 The *sh* utility shall conform to XBD [Section 12.2](#) (on page 215), with an extension for support of a  
 106699 leading <plus-sign> ( ' + ' ) as noted below.

106700 The *-a*, *-b*, *-C*, *-e*, *-f*, *-m*, *-n*, *-o option*, *-u*, *-v*, and *-x* options are described as part of the *set*  
 106701 utility in [Section 2.14](#) (on page 2356). The option letters derived from the *set* special built-in shall  
 106702 also be accepted with a leading <plus-sign> ( ' + ' ) instead of a leading <hyphen> (meaning the  
 106703 reverse case of the option as described in this volume of POSIX.1-2008).

106704 The following additional options shall be supported:

106705 *-c* Read commands from the *command\_string* operand. Set the value of special  
 106706 parameter 0 (see [Section 2.5.2](#), on page 2324) from the value of the *command\_name*  
 106707 operand and the positional parameters (\$1, \$2, and so on) in sequence from the  
 106708 remaining *argument* operands. No commands shall be read from the standard  
 106709 input.

106710 *-i* Specify that the shell is *interactive*; see below. An implementation may treat  
 106711 specifying the *-i* option as an error if the real user ID of the calling process does  
 106712 not equal the effective user ID or if the real group ID does not equal the effective  
 106713 group ID.

106714 *-s* Read commands from the standard input.

106715 If there are no operands and the *-c* option is not specified, the *-s* option shall be assumed.

106716 If the *-i* option is present, or if there are no operands and the shell's standard input and  
 106717 standard error are attached to a terminal, the shell is considered to be *interactive*.

106718 **OPERANDS**

106719 The following operands shall be supported:

106720 *-* A single <hyphen> shall be treated as the first operand and then ignored. If both  
 106721 ' - ' and " -- " are given as arguments, or if other operands precede the single  
 106722 <hyphen>, the results are undefined.

106723 *argument* The positional parameters (\$1, \$2, and so on) shall be set to *arguments*, if any.

106724 *command\_file* The pathname of a file containing commands. If the pathname contains one or  
 106725 more <slash> characters, the implementation attempts to read that file; the file  
 106726 need not be executable. If the pathname does not contain a <slash> character:

- The implementation shall attempt to read that file from the current working directory; the file need not be executable.
- If the file is not in the current working directory, the implementation may perform a search for an executable file using the value of *PATH*, as described in [Section 2.9.1.1](#) (on page 2339).

106727

106728

106729

106730

106731

106732 Special parameter 0 (see [Section 2.5.2](#), on page 2324) shall be set to the value of  
 106733 *command\_file*. If *sh* is called using a synopsis form that omits *command\_file*, special  
 106734 parameter 0 shall be set to the value of the first argument passed to *sh* from its  
 106735 parent (for example, *argv*[0] for a C program), which is normally a pathname used  
 106736 to execute the *sh* utility.

106737 *command\_name*

106738 A string assigned to special parameter 0 when executing the commands in  
 106739 *command\_string*. If *command\_name* is not specified, special parameter 0 shall be set  
 106740 to the value of the first argument passed to *sh* from its parent (for example, *argv*[0]  
 106741 for a C program), which is normally a pathname used to execute the *sh* utility.

106742 *command\_string*

106743 A string that shall be interpreted by the shell as one or more commands, as if the  
 106744 string were the argument to the *system()* function defined in the System Interfaces  
 106745 volume of POSIX.1-2008. If the *command\_string* operand is an empty string, *sh* shall  
 106746 exit with a zero exit status.

106747 **STDIN**

106748 The standard input shall be used only if one of the following is true:

- The *-s* option is specified.
- The *-c* option is not specified and no operands are specified.
- The script executes one or more commands that require input from standard input (such as a *read* command that does not redirect its input).

106749

106750

106751

106752

106753 See the INPUT FILES section.

106754

106755 When the shell is using standard input and it invokes a command that also uses standard input,  
 106756 the shell shall ensure that the standard input file pointer points directly after the command it has  
 106757 read when the command begins execution. It shall not read ahead in such a manner that any  
 106758 characters intended to be read by the invoked command are consumed by the shell (whether  
 106759 interpreted by the shell or not) or that characters that are not read by the invoked command are  
 106760 not seen by the shell. When the command expecting to read standard input is started  
 106761 asynchronously by an interactive shell, it is unspecified whether characters are read by the  
 command or interpreted by the shell.

106762

106763 If the standard input to *sh* is a FIFO or terminal device and is set to non-blocking reads, then *sh*  
 106764 shall enable blocking reads on standard input. This shall remain in effect when the command  
 completes.

106765 **INPUT FILES**

106766 The input file shall be a text file, except that line lengths shall be unlimited. If the input file is  
 106767 empty or consists solely of blank lines or comments, or both, *sh* shall exit with a zero exit status.

106768 **ENVIRONMENT VARIABLES**

106769 The following environment variables shall affect the execution of *sh*:

106770 UP **ENV** This variable, when and only when an interactive shell is invoked, shall be  
 106771 subjected to parameter expansion (see [Section 2.6.2](#), on page 2328) by the shell, and  
 106772 the resulting value shall be used as a pathname of a file containing shell  
 106773 commands to execute in the current environment. The file need not be executable.  
 106774 If the expanded value of *ENV* is not an absolute pathname, the results are  
 106775 unspecified. *ENV* shall be ignored if the real and effective user IDs or real and  
 106776 effective group IDs of the process are different.

106777 UP **FCEDIT** This variable, when expanded by the shell, shall determine the default value for  
 106778 the *-e editor* option's *editor* option-argument. If *FCEDIT* is null or unset, *ed* shall be  
 106779 used as the editor.

106780 UP **HISTFILE** Determine a pathname naming a command history file. If the *HISTFILE* variable is  
 106781 not set, the shell may attempt to access or create a file *.sh\_history* in the directory  
 106782 referred to by the *HOME* environment variable. If the shell cannot obtain both read  
 106783 and write access to, or create, the history file, it shall use an unspecified  
 106784 mechanism that allows the history to operate properly. (References to history  
 106785 "file" in this section shall be understood to mean this unspecified mechanism in  
 106786 such cases.) An implementation may choose to access this variable only when  
 106787 initializing the history file; this initialization shall occur when *fc* or *sh* first attempt  
 106788 to retrieve entries from, or add entries to, the file, as the result of commands issued  
 106789 by the user, the file named by the *ENV* variable, or implementation-defined system  
 106790 start-up files. Implementations may choose to disable the history list mechanism  
 106791 for users with appropriate privileges who do not set *HISTFILE*; the specific  
 106792 circumstances under which this occurs are implementation-defined. If more than  
 106793 one instance of the shell is using the same history file, it is unspecified how  
 106794 updates to the history file from those shells interact. As entries are deleted from the  
 106795 history file, they shall be deleted oldest first. It is unspecified when history file  
 106796 entries are physically removed from the history file.

106797 **HISTSIZE** Determine a decimal number representing the limit to the number of previous  
 106798 commands that are accessible. If this variable is unset, an unspecified default  
 106799 greater than or equal to 128 shall be used. The maximum number of commands in  
 106800 the history list is unspecified, but shall be at least 128. An implementation may  
 106801 choose to access this variable only when initializing the history file, as described  
 106802 under *HISTFILE*. Therefore, it is unspecified whether changes made to *HISTSIZE*  
 106803 after the history file has been initialized are effective.

106804 UP **HOME** Determine the pathname of the user's home directory. The contents of *HOME* are  
 106805 used in tilde expansion as described in [Section 2.6.1](#) (on page 2327).

106806 **IFS** A string treated as a list of characters that is used for field splitting and to split  
 106807 lines into fields with the *read* command.

106808 If *IFS* is not set, it shall behave as normal for an unset variable, except that field  
 106809 splitting by the shell and line splitting by the *read* command shall be performed as  
 106810 if the value of *IFS* is <space><tab><newline>; see [Section 2.6.5](#) (on page 2333).

106811 Implementations may ignore the value of *IFS* in the environment, or the absence of

|        |                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|--------|---------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 106812 |                     | <i>IFS</i> from the environment, at the time the shell is invoked, in which case the shell shall set <i>IFS</i> to <space><tab><newline> when it is invoked.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 106813 |                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 106814 | <i>LANG</i>         | Provide a default value for the internationalization variables that are unset or null. (See XBD <a href="#">Section 8.2</a> (on page 174) for the precedence of internationalization variables used to determine the values of locale categories.)                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| 106815 |                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 106816 |                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 106817 | <i>LC_ALL</i>       | If set to a non-empty string value, override the values of all the other internationalization variables.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| 106818 |                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 106819 | <i>LC_COLLATE</i>   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 106820 |                     | Determine the behavior of range expressions, equivalence classes, and multi-character collating elements within pattern matching.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 106821 |                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 106822 | <i>LC_CTYPE</i>     | Determine the locale for the interpretation of sequences of bytes of text data as characters (for example, single-byte as opposed to multi-byte characters in arguments and input files), which characters are defined as letters (character class <b>alpha</b> ), and the behavior of character classes within pattern matching.                                                                                                                                                                                                                                                                                                                                                  |
| 106823 |                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 106824 |                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 106825 |                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 106826 | <i>LC_MESSAGES</i>  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 106827 |                     | Determine the locale that should be used to affect the format and contents of diagnostic messages written to standard error.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 106828 |                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 106829 | UP <i>MAIL</i>      | Determine a pathname of the user's mailbox file for purposes of incoming mail notification. If this variable is set, the shell shall inform the user if the file named by the variable is created or if its modification time has changed. Informing the user shall be accomplished by writing a string of unspecified format to standard error prior to the writing of the next primary prompt string. Such check shall be performed only after the completion of the interval defined by the <i>MAILCHECK</i> variable after the last such check. The user shall be informed only if <i>MAIL</i> is set and <i>MAILPATH</i> is not set.                                          |
| 106830 |                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 106831 |                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 106832 |                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 106833 |                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 106834 |                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 106835 |                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 106836 |                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 106837 | UP <i>MAILCHECK</i> |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 106838 |                     | Establish a decimal integer value that specifies how often (in seconds) the shell shall check for the arrival of mail in the files specified by the <i>MAILPATH</i> or <i>MAIL</i> variables. The default value shall be 600 seconds. If set to zero, the shell shall check before issuing each primary prompt.                                                                                                                                                                                                                                                                                                                                                                    |
| 106839 |                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 106840 |                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 106841 |                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 106842 | UP <i>MAILPATH</i>  | Provide a list of pathnames and optional messages separated by <colon> characters. If this variable is set, the shell shall inform the user if any of the files named by the variable are created or if any of their modification times change. (See the preceding entry for <i>MAIL</i> for descriptions of mail arrival and user informing.) Each pathname can be followed by ' % ' and a string that shall be subjected to parameter expansion and written to standard error when the modification time changes. If a ' % ' character in the pathname is preceded by a <backslash>, it shall be treated as a literal ' % ' in the pathname. The default message is unspecified. |
| 106843 |                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 106844 |                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 106845 |                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 106846 |                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 106847 |                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 106848 |                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 106849 |                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 106850 |                     | The <i>MAILPATH</i> environment variable takes precedence over the <i>MAIL</i> variable.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| 106851 | XSI <i>NLSPATH</i>  | Determine the location of message catalogs for the processing of <i>LC_MESSAGES</i> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 106852 | <i>PATH</i>         | Establish a string formatted as described in XBD <a href="#">Chapter 8</a> (on page 173), used to effect command interpretation; see <a href="#">Section 2.9.1.1</a> (on page 2339).                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| 106853 |                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 106854 | <i>PWD</i>          | This variable shall represent an absolute pathname of the current working directory. Assignments to this variable may be ignored.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 106855 |                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |

106856 **ASYNCHRONOUS EVENTS**

106857 The *sh* utility shall take the standard action for all signals (see [Section 1.4](#), on page 2310) with the  
 106858 following exceptions.

106859 If the shell is interactive, SIGINT signals received during command line editing shall be handled  
 106860 as described in the EXTENDED DESCRIPTION, and SIGINT signals received at other times  
 106861 shall be caught but no action performed.

106862 If the shell is interactive:

- 106863 • SIGQUIT and SIGTERM signals shall be ignored.
- 106864 • If the **-m** option is in effect, SIGTTIN, SIGTTOU, and SIGTSTP signals shall be ignored.
- 106865 • If the **-m** option is not in effect, it is unspecified whether SIGTTIN, SIGTTOU, and  
 106866 SIGTSTP signals are ignored, set to the default action, or caught. If they are caught, the  
 106867 shell shall, in the signal-catching function, set the signal to the default action and raise the  
 106868 signal (after taking any appropriate steps, such as restoring terminal settings).

106869 The standard actions, and the actions described above for interactive shells, can be overridden  
 106870 by use of the *trap* special built-in utility (see *trap* (on page 2391) and [Section 2.11](#), on page 2353).

106871 **STDOUT**

106872 See the STDERR section.

106873 **STDERR**

106874 Except as otherwise stated (by the descriptions of any invoked utilities or in interactive mode),  
 106875 standard error shall be used only for diagnostic messages.

106876 **OUTPUT FILES**

106877 None.

106878 **EXTENDED DESCRIPTION**

106879 UP See [Chapter 2](#). The functionality described in the rest of the EXTENDED DESCRIPTION section  
 106880 shall be provided on implementations that support the User Portability Utilities option (and the  
 106881 rest of this section is not further shaded for this option).

106882 **Command History List**

106883 When the *sh* utility is being used interactively, it shall maintain a list of commands previously  
 106884 entered from the terminal in the file named by the *HISTFILE* environment variable. The type,  
 106885 size, and internal format of this file are unspecified. Multiple *sh* processes can share access to the  
 106886 file for a user, if file access permissions allow this; see the description of the *HISTFILE*  
 106887 environment variable.

106888 **Command Line Editing**

106889 When *sh* is being used interactively from a terminal, the current command and the command  
 106890 history (see *fc*) can be edited using *vi*-mode command line editing. This mode uses commands,  
 106891 described below, similar to a subset of those described in the *vi* utility. Implementations may  
 106892 offer other command line editing modes corresponding to other editing utilities.

106893 The command *set -o vi* shall enable *vi*-mode editing and place *sh* into *vi* insert mode (see  
 106894 [Command Line Editing \(vi-mode\)](#), on page 3194). This command also shall disable any other  
 106895 editing mode that the implementation may provide. The command *set +o vi* disables *vi*-mode  
 106896 editing.

106897 Certain block-mode terminals may be unable to support shell command line editing. If a  
 106898 terminal is unable to provide either edit mode, it need not be possible to *set -o vi* when using the



106899 shell on this terminal.

106900 In the following sections, the characters *erase*, *interrupt*, *kill*, and *end-of-file* are those set by the  
106901 *stty* utility.

## 106902 **Command Line Editing (vi-mode)**

106903 In *vi* editing mode, there shall be a distinguished line, the edit line. All the editing operations  
106904 which modify a line affect the edit line. The edit line is always the newest line in the command  
106905 history buffer.

106906 With *vi*-mode enabled, *sh* can be switched between insert mode and command mode.

106907 When in insert mode, an entered character shall be inserted into the command line, except as  
106908 noted in [vi Line Editing Insert Mode](#). Upon entering *sh* and after termination of the previous  
106909 command, *sh* shall be in insert mode.

106910 Typing an escape character shall switch *sh* into command mode (see [vi Line Editing Command](#)  
106911 [Mode](#), on page 3195). In command mode, an entered character shall either invoke a defined  
106912 operation, be used as part of a multi-character operation, or be treated as an error. A character  
106913 that is not recognized as part of an editing command shall terminate any specific editing  
106914 command and shall alert the terminal. If *sh* receives a SIGINT signal in command mode |  
106915 (whether generated by typing the *interrupt* character or by other means), it shall terminate |  
106916 command line editing on the current command line, reissue the prompt on the next line of the |  
106917 terminal, and reset the command history (see [fc](#)) so that the most recently executed command is |  
106918 the previous command (that is, the command that was being edited when it was interrupted is |  
106919 not re-entered into the history).

106920 In the following sections, the phrase “move the cursor to the beginning of the word” shall mean  
106921 “move the cursor to the first character of the current word” and the phrase “move the cursor to  
106922 the end of the word” shall mean “move the cursor to the last character of the current word”. The  
106923 phrase “beginning of the command line” indicates the point between the end of the prompt  
106924 string issued by the shell (or the beginning of the terminal line, if there is no prompt string) and  
106925 the first character of the command text.

## 106926 **vi Line Editing Insert Mode**

106927 While in insert mode, any character typed shall be inserted in the current command line, unless  
106928 it is from the following set.

106929 <newline> Execute the current command line. If the current command line is not empty, this  
106930 line shall be entered into the command history (see [fc](#)).

106931 *erase* Delete the character previous to the current cursor position and move the current  
106932 cursor position back one character. In insert mode, characters shall be erased from  
106933 both the screen and the buffer when backspacing.

106934 *interrupt* If *sh* receives a SIGINT signal in insert mode (whether generated by typing the |  
106935 *interrupt* character or by other means), it shall terminate command line editing |  
106936 with the same effects as described for interrupting command mode; see [Command](#)  
106937 [Line Editing \(vi-mode\)](#).

106938 *kill* Clear all the characters from the input line.

106939 <control>-V Insert the next character input, even if the character is otherwise a special insert  
106940 mode character.



106941 <control>-W Delete the characters from the one preceding the cursor to the preceding word  
 106942 boundary. The word boundary in this case is the closer to the cursor of either the  
 106943 beginning of the line or a character that is in neither the **blank** nor **punct** character  
 106944 classification of the current locale.

106945 *end-of-file* Interpreted as the end of input in *sh*. This interpretation shall occur only at the  
 106946 beginning of an input line. If *end-of-file* is entered other than at the beginning of the  
 106947 line, the results are unspecified.

106948 <ESC> Place *sh* into command mode.

## 106949 **vi Line Editing Command Mode**

106950 In command mode for the command line editing feature, decimal digits not beginning with 0  
 106951 that precede a command letter shall be remembered. Some commands use these decimal digits  
 106952 as a count number that affects the operation.

106953 The term *motion command* represents one of the commands:

106954 <space> 0 b F l W ^ \$ ; E f T w | , B e h t

106955 If the current line is not the edit line, any command that modifies the current line shall cause the  
 106956 content of the current line to replace the content of the edit line, and the current line shall  
 106957 become the edit line. This replacement cannot be undone (see the **u** and **U** commands below).  
 106958 The modification requested shall then be performed to the edit line. When the current line is the  
 106959 edit line, the modification shall be done directly to the edit line.

106960 Any command that is preceded by *count* shall take a count (the numeric value of any preceding  
 106961 decimal digits). Unless otherwise noted, this count shall cause the specified operation to repeat  
 106962 by the number of times specified by the count. Also unless otherwise noted, a *count* that is out  
 106963 of range is considered an error condition and shall alert the terminal, but neither the cursor  
 106964 position, nor the command line, shall change.

106965 The terms *word* and *bigword* are used as defined in the *vi* description. The term *save buffer*  
 106966 corresponds to the term *unnamed buffer* in *vi*.

106967 The following commands shall be recognized in command mode:

106968 <newline> Execute the current command line. If the current command line is not empty, this  
 106969 line shall be entered into the command history (see *fc*).

106970 <control>-L Redraw the current command line. Position the cursor at the same location on the  
 106971 redrawn line.

106972 **#** Insert the character ' #' at the beginning of the current command line and treat the  
 106973 resulting edit line as a comment. This line shall be entered into the command  
 106974 history; see *fc*.

106975 **=** Display the possible shell word expansions (see [Section 2.6](#), on page 2327) of the  
 106976 bigword at the current command line position.

106977 **Note:** This does not modify the content of the current line, and therefore does not cause  
 106978 the current line to become the edit line.

106979 These expansions shall be displayed on subsequent terminal lines. If the bigword  
 106980 contains none of the characters '?', '\*', or '[', an <asterisk> ('\*') shall be  
 106981 implicitly assumed at the end. If any directories are matched, these expansions  
 106982 shall have a '/' character appended. After the expansion, the line shall be  
 106983 redrawn, the cursor repositioned at the current cursor position, and *sh* shall be  
 106984 placed in command mode.

|        |                |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|--------|----------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 106985 | \              | Perform pathname expansion (see <a href="#">Section 2.6.6</a> , on page 2333) on the current bigword, up to the largest set of characters that can be matched uniquely. If the bigword contains none of the characters '?', '*', or '[', an <asterisk> ('*') shall be implicitly assumed at the end. This maximal expansion then shall replace the original bigword in the command line, and the cursor shall be placed after this expansion. If the resulting bigword completely and uniquely matches a directory, a '/' character shall be inserted directly after the bigword. If some other file is completely matched, a single <space> shall be inserted after the bigword. After this operation, <i>sh</i> shall be placed in insert mode. |
| 106986 |                |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 106987 |                |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 106988 |                |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 106989 |                |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 106990 |                |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 106991 |                |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 106992 |                |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 106993 |                |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 106994 | *              | Perform pathname expansion on the current bigword and insert all expansions into the command to replace the current bigword, with each expansion separated by a single <space>. If at the end of the line, the current cursor position shall be moved to the first column position following the expansions and <i>sh</i> shall be placed in insert mode. Otherwise, the current cursor position shall be the last column position of the first character after the expansions and <i>sh</i> shall be placed in insert mode. If the current bigword contains none of the characters '?', '*', or '[', before the operation, an <asterisk> ('*') shall be implicitly assumed at the end.                                                           |
| 106995 |                |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 106996 |                |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 106997 |                |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 106998 |                |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 106999 |                |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 107000 |                |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 107001 |                |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 107002 | @letter        | Insert the value of the alias named <i>_letter</i> . The symbol <i>letter</i> represents a single alphabetic character from the portable character set; implementations may support additional characters as an extension. If the alias <i>_letter</i> contains other editing commands, these commands shall be performed as part of the insertion. If no alias <i>_letter</i> is enabled, this command shall have no effect.                                                                                                                                                                                                                                                                                                                     |
| 107003 |                |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 107004 |                |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 107005 |                |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 107006 |                |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 107007 | [count]~       | Convert, if the current character is a lowercase letter, to the equivalent uppercase letter and <i>vice versa</i> , as prescribed by the current locale. The current cursor position then shall be advanced by one character. If the cursor was positioned on the last character of the line, the case conversion shall occur, but the cursor shall not advance. If the '~' command is preceded by a <i>count</i> , that number of characters shall be converted, and the cursor shall be advanced to the character position after the last character converted. If the <i>count</i> is larger than the number of characters after the cursor, this shall not be considered an error; the cursor shall advance to the last character on the line. |
| 107008 |                |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 107009 |                |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 107010 |                |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 107011 |                |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 107012 |                |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 107013 |                |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 107014 |                |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 107015 |                |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 107016 | [count].       | Repeat the most recent non-motion command, even if it was executed on an earlier command line. If the previous command was preceded by a <i>count</i> , and no count is given on the '.' command, the count from the previous command shall be included as part of the repeated command. If the '.' command is preceded by a <i>count</i> , this shall override any <i>count</i> argument to the previous command. The <i>count</i> specified in the '.' command shall become the count for subsequent '.' commands issued without a count.                                                                                                                                                                                                       |
| 107017 |                |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 107018 |                |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 107019 |                |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 107020 |                |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 107021 |                |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 107022 |                |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 107023 | [number]v      | Invoke the <i>vi</i> editor to edit the current command line in a temporary file. When the editor exits, the commands in the temporary file shall be executed and placed in the command history. If a <i>number</i> is included, it specifies the command number in the command history to be edited, rather than the current command line.                                                                                                                                                                                                                                                                                                                                                                                                       |
| 107024 |                |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 107025 |                |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 107026 |                |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 107027 | [count]l (ell) |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 107028 | [count]<space> |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 107029 |                | Move the current cursor position to the next character position. If the cursor was positioned on the last character of the line, the terminal shall be alerted and the cursor shall not be advanced. If the <i>count</i> is larger than the number of characters after the cursor, this shall not be considered an error; the cursor shall advance to the last character on the line.                                                                                                                                                                                                                                                                                                                                                             |
| 107030 |                |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 107031 |                |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 107032 |                |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 107033 |                |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |

|        |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|--------|-----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 107034 | [count]h  | Move the current cursor position to the <i>count</i> th (default 1) previous character position. If the cursor was positioned on the first character of the line, the terminal shall be alerted and the cursor shall not be moved. If the count is larger than the number of characters before the cursor, this shall not be considered an error; the cursor shall move to the first character on the line.                                         |
| 107035 |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 107036 |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 107037 |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 107038 |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 107039 | [count]w  | Move to the start of the next word. If the cursor was positioned on the last character of the line, the terminal shall be alerted and the cursor shall not be advanced. If the <i>count</i> is larger than the number of words after the cursor, this shall not be considered an error; the cursor shall advance to the last character on the line.                                                                                                 |
| 107040 |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 107041 |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 107042 |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 107043 |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 107044 | [count]W  | Move to the start of the next bigword. If the cursor was positioned on the last character of the line, the terminal shall be alerted and the cursor shall not be advanced. If the <i>count</i> is larger than the number of bigwords after the cursor, this shall not be considered an error; the cursor shall advance to the last character on the line.                                                                                           |
| 107045 |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 107046 |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 107047 |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 107048 |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 107049 | [count]e  | Move to the end of the current word. If at the end of a word, move to the end of the next word. If the cursor was positioned on the last character of the line, the terminal shall be alerted and the cursor shall not be advanced. If the <i>count</i> is larger than the number of words after the cursor, this shall not be considered an error; the cursor shall advance to the last character on the line.                                     |
| 107050 |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 107051 |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 107052 |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 107053 |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 107054 | [count]E  | Move to the end of the current bigword. If at the end of a bigword, move to the end of the next bigword. If the cursor was positioned on the last character of the line, the terminal shall be alerted and the cursor shall not be advanced. If the <i>count</i> is larger than the number of bigwords after the cursor, this shall not be considered an error; the cursor shall advance to the last character on the line.                         |
| 107055 |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 107056 |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 107057 |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 107058 |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 107059 | [count]b  | Move to the beginning of the current word. If at the beginning of a word, move to the beginning of the previous word. If the cursor was positioned on the first character of the line, the terminal shall be alerted and the cursor shall not be moved. If the <i>count</i> is larger than the number of words preceding the cursor, this shall not be considered an error; the cursor shall return to the first character on the line.             |
| 107060 |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 107061 |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 107062 |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 107063 |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 107064 | [count]B  | Move to the beginning of the current bigword. If at the beginning of a bigword, move to the beginning of the previous bigword. If the cursor was positioned on the first character of the line, the terminal shall be alerted and the cursor shall not be moved. If the <i>count</i> is larger than the number of bigwords preceding the cursor, this shall not be considered an error; the cursor shall return to the first character on the line. |
| 107065 |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 107066 |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 107067 |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 107068 |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 107069 | ^         | Move the current cursor position to the first character on the input line that is not a <blank>.                                                                                                                                                                                                                                                                                                                                                    |
| 107070 |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 107071 |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 107072 |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 107073 |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 107074 | \$        | Move to the last character position on the current command line.                                                                                                                                                                                                                                                                                                                                                                                    |
| 107075 | 0         | (Zero.) Move to the first character position on the current command line.                                                                                                                                                                                                                                                                                                                                                                           |
| 107076 | [count] l | Move to the <i>count</i> th character position on the current command line. If no number is specified, move to the first position. The first character position shall be numbered 1. If the count is larger than the number of characters on the line, this shall not be considered an error; the cursor shall be placed on the last character on the line.                                                                                         |
| 107077 |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 107078 |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 107079 |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                     |

|        |                       |                                                                                                                                                                                                                                                                                                                                                                                                               |
|--------|-----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 107080 | <b>[count]fc</b>      | Move to the first occurrence of the character 'c' that occurs after the current cursor position. If the cursor was positioned on the last character of the line, the terminal shall be alerted and the cursor shall not be advanced. If the character 'c' does not occur in the line after the current cursor position, the terminal shall be alerted and the cursor shall not be moved.                      |
| 107081 |                       |                                                                                                                                                                                                                                                                                                                                                                                                               |
| 107082 |                       |                                                                                                                                                                                                                                                                                                                                                                                                               |
| 107083 |                       |                                                                                                                                                                                                                                                                                                                                                                                                               |
| 107084 |                       |                                                                                                                                                                                                                                                                                                                                                                                                               |
| 107085 | <b>[count]Fc</b>      | Move to the first occurrence of the character 'c' that occurs before the current cursor position. If the cursor was positioned on the first character of the line, the terminal shall be alerted and the cursor shall not be moved. If the character 'c' does not occur in the line before the current cursor position, the terminal shall be alerted and the cursor shall not be moved.                      |
| 107086 |                       |                                                                                                                                                                                                                                                                                                                                                                                                               |
| 107087 |                       |                                                                                                                                                                                                                                                                                                                                                                                                               |
| 107088 |                       |                                                                                                                                                                                                                                                                                                                                                                                                               |
| 107089 |                       |                                                                                                                                                                                                                                                                                                                                                                                                               |
| 107090 | <b>[count]tc</b>      | Move to the character before the first occurrence of the character 'c' that occurs after the current cursor position. If the cursor was positioned on the last character of the line, the terminal shall be alerted and the cursor shall not be advanced. If the character 'c' does not occur in the line after the current cursor position, the terminal shall be alerted and the cursor shall not be moved. |
| 107091 |                       |                                                                                                                                                                                                                                                                                                                                                                                                               |
| 107092 |                       |                                                                                                                                                                                                                                                                                                                                                                                                               |
| 107093 |                       |                                                                                                                                                                                                                                                                                                                                                                                                               |
| 107094 |                       |                                                                                                                                                                                                                                                                                                                                                                                                               |
| 107095 | <b>[count]Tc</b>      | Move to the character after the first occurrence of the character 'c' that occurs before the current cursor position. If the cursor was positioned on the first character of the line, the terminal shall be alerted and the cursor shall not be moved. If the character 'c' does not occur in the line before the current cursor position, the terminal shall be alerted and the cursor shall not be moved.  |
| 107096 |                       |                                                                                                                                                                                                                                                                                                                                                                                                               |
| 107097 |                       |                                                                                                                                                                                                                                                                                                                                                                                                               |
| 107098 |                       |                                                                                                                                                                                                                                                                                                                                                                                                               |
| 107099 |                       |                                                                                                                                                                                                                                                                                                                                                                                                               |
| 107100 | <b>[count];</b>       | Repeat the most recent <b>f</b> , <b>F</b> , <b>t</b> , or <b>T</b> command. Any number argument on that previous command shall be ignored. Errors are those described for the repeated command.                                                                                                                                                                                                              |
| 107101 |                       |                                                                                                                                                                                                                                                                                                                                                                                                               |
| 107102 |                       |                                                                                                                                                                                                                                                                                                                                                                                                               |
| 107103 | <b>[count],</b>       | Repeat the most recent <b>f</b> , <b>F</b> , <b>t</b> , or <b>T</b> command. Any number argument on that previous command shall be ignored. However, reverse the direction of that command.                                                                                                                                                                                                                   |
| 107104 |                       |                                                                                                                                                                                                                                                                                                                                                                                                               |
| 107105 |                       |                                                                                                                                                                                                                                                                                                                                                                                                               |
| 107106 | <b>a</b>              | Enter insert mode after the current cursor position. Characters that are entered shall be inserted before the next character.                                                                                                                                                                                                                                                                                 |
| 107107 |                       |                                                                                                                                                                                                                                                                                                                                                                                                               |
| 107108 | <b>A</b>              | Enter insert mode after the end of the current command line.                                                                                                                                                                                                                                                                                                                                                  |
| 107109 | <b>i</b>              | Enter insert mode at the current cursor position. Characters that are entered shall be inserted before the current character.                                                                                                                                                                                                                                                                                 |
| 107110 |                       |                                                                                                                                                                                                                                                                                                                                                                                                               |
| 107111 | <b>I</b>              | Enter insert mode at the beginning of the current command line.                                                                                                                                                                                                                                                                                                                                               |
| 107112 | <b>R</b>              | Enter insert mode, replacing characters from the command line beginning at the current cursor position.                                                                                                                                                                                                                                                                                                       |
| 107113 |                       |                                                                                                                                                                                                                                                                                                                                                                                                               |
| 107114 | <b>[count]cmotion</b> |                                                                                                                                                                                                                                                                                                                                                                                                               |
| 107115 |                       | Delete the characters between the current cursor position and the cursor position that would result from the specified motion command. Then enter insert mode before the first character following any deleted characters. If <i>count</i> is specified, it shall be applied to the motion command. A <i>count</i> shall be ignored for the following motion commands:                                        |
| 107116 |                       |                                                                                                                                                                                                                                                                                                                                                                                                               |
| 107117 |                       |                                                                                                                                                                                                                                                                                                                                                                                                               |
| 107118 |                       |                                                                                                                                                                                                                                                                                                                                                                                                               |
| 107119 |                       |                                                                                                                                                                                                                                                                                                                                                                                                               |
| 107120 |                       | 0      ^      \$      c                                                                                                                                                                                                                                                                                                                                                                                       |
| 107121 |                       | If the motion command is the character 'c', the current command line shall be cleared and insert mode shall be entered. If the motion command would move the current cursor position toward the beginning of the command line, the character under the current cursor position shall not be deleted. If the motion command                                                                                    |
| 107122 |                       |                                                                                                                                                                                                                                                                                                                                                                                                               |
| 107123 |                       |                                                                                                                                                                                                                                                                                                                                                                                                               |
| 107124 |                       |                                                                                                                                                                                                                                                                                                                                                                                                               |

|        |                               |                                                                                                   |
|--------|-------------------------------|---------------------------------------------------------------------------------------------------|
| 107125 |                               | would move the current cursor position toward the end of the command line, the                    |
| 107126 |                               | character under the current cursor position shall be deleted. If the <i>count</i> is larger       |
| 107127 |                               | than the number of characters between the current cursor position and the end of                  |
| 107128 |                               | the command line toward which the motion command would move the cursor, this                      |
| 107129 |                               | shall not be considered an error; all of the remaining characters in the                          |
| 107130 |                               | aforementioned range shall be deleted and insert mode shall be entered. If the                    |
| 107131 |                               | motion command is invalid, the terminal shall be alerted, the cursor shall not be                 |
| 107132 |                               | moved, and no text shall be deleted.                                                              |
| 107133 | <b>C</b>                      | Delete from the current character to the end of the line and enter insert mode at the             |
| 107134 |                               | new end-of-line.                                                                                  |
| 107135 | <b>S</b>                      | Clear the entire edit line and enter insert mode.                                                 |
| 107136 | <b>[count]rc</b>              | Replace the current character with the character 'c'. With a number <i>count</i> ,                |
| 107137 |                               | replace the current and the following <i>count</i> –1 characters. After this command, the         |
| 107138 |                               | current cursor position shall be on the last character that was changed. If the <i>count</i>      |
| 107139 |                               | is larger than the number of characters after the cursor, this shall not be considered            |
| 107140 |                               | an error; all of the remaining characters shall be changed.                                       |
| 107141 | <b>[count]_</b>               | Append a <space> after the current character position and then append the last                    |
| 107142 |                               | bigword in the previous input line after the <space>. Then enter insert mode after                |
| 107143 |                               | the last character just appended. With a number <i>count</i> , append the <i>count</i> th bigword |
| 107144 |                               | in the previous line.                                                                             |
| 107145 | <b>[count]x</b>               | Delete the character at the current cursor position and place the deleted characters              |
| 107146 |                               | in the save buffer. If the cursor was positioned on the last character of the line, the           |
| 107147 |                               | character shall be deleted and the cursor position shall be moved to the previous                 |
| 107148 |                               | character (the new last character). If the <i>count</i> is larger than the number of              |
| 107149 |                               | characters after the cursor, this shall not be considered an error; all the characters            |
| 107150 |                               | from the cursor to the end of the line shall be deleted.                                          |
| 107151 | <b>[count]X</b>               | Delete the character before the current cursor position and place the deleted                     |
| 107152 |                               | characters in the save buffer. The character under the current cursor position shall              |
| 107153 |                               | not change. If the cursor was positioned on the first character of the line, the                  |
| 107154 |                               | terminal shall be alerted, and the X command shall have no effect. If the line                    |
| 107155 |                               | contained a single character, the X command shall have no effect. If the line                     |
| 107156 |                               | contained no characters, the terminal shall be alerted and the cursor shall not be                |
| 107157 |                               | moved. If the <i>count</i> is larger than the number of characters before the cursor, this        |
| 107158 |                               | shall not be considered an error; all the characters from before the cursor to the                |
| 107159 |                               | beginning of the line shall be deleted.                                                           |
| 107160 | <b>[count]d</b> <i>motion</i> |                                                                                                   |
| 107161 |                               | Delete the characters between the current cursor position and the character                       |
| 107162 |                               | position that would result from the motion command. A number <i>count</i> repeats the             |
| 107163 |                               | motion command <i>count</i> times. If the motion command would move toward the                    |
| 107164 |                               | beginning of the command line, the character under the current cursor position                    |
| 107165 |                               | shall not be deleted. If the motion command is <b>d</b> , the entire current command line         |
| 107166 |                               | shall be cleared. If the <i>count</i> is larger than the number of characters between the         |
| 107167 |                               | current cursor position and the end of the command line toward which the motion                   |
| 107168 |                               | command would move the cursor, this shall not be considered an error; all of the                  |
| 107169 |                               | remaining characters in the aforementioned range shall be deleted. The deleted                    |
| 107170 |                               | characters shall be placed in the save buffer.                                                    |

|        |                       |                                                                                               |
|--------|-----------------------|-----------------------------------------------------------------------------------------------|
| 107171 | <b>D</b>              | Delete all characters from the current cursor position to the end of the line. The            |
| 107172 |                       | deleted characters shall be placed in the save buffer.                                        |
| 107173 | <b>[count]ymotion</b> |                                                                                               |
| 107174 |                       | Yank (that is, copy) the characters from the current cursor position to the position          |
| 107175 |                       | resulting from the motion command into the save buffer. A number <i>count</i> shall be        |
| 107176 |                       | applied to the motion command. If the motion command would move toward the                    |
| 107177 |                       | beginning of the command line, the character under the current cursor position                |
| 107178 |                       | shall not be included in the set of yanked characters. If the motion command is <b>y</b> ,    |
| 107179 |                       | the entire current command line shall be yanked into the save buffer. The current             |
| 107180 |                       | cursor position shall be unchanged. If the <i>count</i> is larger than the number of          |
| 107181 |                       | characters between the current cursor position and the end of the command line                |
| 107182 |                       | toward which the motion command would move the cursor, this shall not be                      |
| 107183 |                       | considered an error; all of the remaining characters in the aforementioned range              |
| 107184 |                       | shall be yanked.                                                                              |
| 107185 | <b>Y</b>              | Yank the characters from the current cursor position to the end of the line into the          |
| 107186 |                       | save buffer. The current character position shall be unchanged.                               |
| 107187 | <b>[count]p</b>       | Put a copy of the current contents of the save buffer after the current cursor                |
| 107188 |                       | position. The current cursor position shall be advanced to the last character put             |
| 107189 |                       | from the save buffer. A <i>count</i> shall indicate how many copies of the save buffer        |
| 107190 |                       | shall be put.                                                                                 |
| 107191 | <b>[count]P</b>       | Put a copy of the current contents of the save buffer before the current cursor               |
| 107192 |                       | position. The current cursor position shall be moved to the last character put from           |
| 107193 |                       | the save buffer. A <i>count</i> shall indicate how many copies of the save buffer shall be    |
| 107194 |                       | put.                                                                                          |
| 107195 | <b>u</b>              | Undo the last command that changed the edit line. This operation shall not undo               |
| 107196 |                       | the copy of any command line to the edit line.                                                |
| 107197 | <b>U</b>              | Undo all changes made to the edit line. This operation shall not undo the copy of             |
| 107198 |                       | any command line to the edit line.                                                            |
| 107199 | <b>[count]k</b>       |                                                                                               |
| 107200 | <b>[count]–</b>       | Set the current command line to be the <i>count</i> th previous command line in the shell     |
| 107201 |                       | command history. If <i>count</i> is not specified, it shall default to 1. The cursor shall be |
| 107202 |                       | positioned on the first character of the new command. If a <b>k</b> or <b>–</b> command would |
| 107203 |                       | retreat past the maximum number of commands in effect for this shell (affected by             |
| 107204 |                       | the <i>HISTSIZE</i> environment variable), the terminal shall be alerted, and the             |
| 107205 |                       | command shall have no effect.                                                                 |
| 107206 | <b>[count]j</b>       |                                                                                               |
| 107207 | <b>[count]+</b>       | Set the current command line to be the <i>count</i> th next command line in the shell         |
| 107208 |                       | command history. If <i>count</i> is not specified, it shall default to 1. The cursor shall be |
| 107209 |                       | positioned on the first character of the new command. If a <b>j</b> or <b>+</b> command       |
| 107210 |                       | advances past the edit line, the current command line shall be restored to the edit           |
| 107211 |                       | line and the terminal shall be alerted.                                                       |
| 107212 | <b>[number]G</b>      | Set the current command line to be the oldest command line stored in the shell                |
| 107213 |                       | command history. With a number <i>number</i> , set the current command line to be the         |
| 107214 |                       | command line <i>number</i> in the history. If command line <i>number</i> does not exist, the  |
| 107215 |                       | terminal shall be alerted and the command line shall not be changed.                          |

107216 */pattern<newline>*

107217 Move backwards through the command history, searching for the specified  
 107218 pattern, beginning with the previous command line. Patterns use the pattern  
 107219 matching notation described in [Section 2.13](#) (on page 2354), except that the '^'  
 107220 character shall have special meaning when it appears as the first character of  
 107221 *pattern*. In this case, the '^' is discarded and the characters after the '^' shall be  
 107222 matched only at the beginning of a line. Commands in the command history shall  
 107223 be treated as strings, not as filenames. If the pattern is not found, the current  
 107224 command line shall be unchanged and the terminal is alerted. If it is found in a  
 107225 previous line, the current command line shall be set to that line and the cursor  
 107226 shall be set to the first character of the new command line.

107227 If *pattern* is empty, the last non-empty pattern provided to / or ? shall be used. If  
 107228 there is no previous non-empty pattern, the terminal shall be alerted and the  
 107229 current command line shall remain unchanged.

107230 *?pattern<newline>*

107231 Move forwards through the command history, searching for the specified pattern,  
 107232 beginning with the next command line. Patterns use the pattern matching notation  
 107233 described in [Section 2.13](#) (on page 2354), except that the '^' character shall have  
 107234 special meaning when it appears as the first character of *pattern*. In this case, the  
 107235 '^' is discarded and the characters after the '^' shall be matched only at the  
 107236 beginning of a line. Commands in the command history shall be treated as strings,  
 107237 not as filenames. If the pattern is not found, the current command line shall be  
 107238 unchanged and the terminal alerted. If it is found in a following line, the current  
 107239 command line shall be set to that line and the cursor shall be set to the first  
 107240 character of the new command line.

107241 If *pattern* is empty, the last non-empty pattern provided to / or ? shall be used. If  
 107242 there is no previous non-empty pattern, the terminal shall be alerted and the  
 107243 current command line shall remain unchanged.

107244 **n** Repeat the most recent / or ? command. If there is no previous / or ?, the terminal  
 107245 shall be alerted and the current command line shall remain unchanged.

107246 **N** Repeat the most recent / or ? command, reversing the direction of the search. If  
 107247 there is no previous / or ?, the terminal shall be alerted and the current command  
 107248 line shall remain unchanged.

## 107249 EXIT STATUS

107250 The following exit values shall be returned:

107251 0 The script to be executed consisted solely of zero or more blank lines or comments, or  
 107252 both.

107253 1-125 A non-interactive shell detected an error other than *command\_file* not found, including  
 107254 but not limited to syntax, redirection, or variable assignment errors.

107255 127 A specified *command\_file* could not be found by a non-interactive shell.

107256 Otherwise, the shell shall return the exit status of the last command it invoked or attempted to  
 107257 invoke (see also the *exit* utility in [Section 2.14](#), on page 2356).

## 107258 CONSEQUENCES OF ERRORS

107259 See [Section 2.8.1](#) (on page 2337).

## APPLICATION USAGE

Standard input and standard error are the files that determine whether a shell is interactive when `-i` is not specified. For example:

```
sh > file
```

and:

```
sh 2> file
```

create interactive and non-interactive shells, respectively. Although both accept terminal input, the results of error conditions are different, as described in [Section 2.8.1](#) (on page 2337); in the second example a redirection error encountered by a special built-in utility aborts the shell.

A conforming application must protect its first operand, if it starts with a <plus-sign>, by preceding it with the `--` argument that denotes the end of the options.

Applications should note that the standard *PATH* to the shell cannot be assumed to be either `/bin/sh` or `/usr/bin/sh`, and should be determined by interrogation of the *PATH* returned by *getconf PATH*, ensuring that the returned pathname is an absolute pathname and not a shell built-in.

For example, to determine the location of the standard *sh* utility:

```
command -v sh
```

On some implementations this might return:

```
/usr/xpg4/bin/sh
```

Furthermore, on systems that support executable scripts (the `#!` construct), it is recommended that applications using executable scripts install them using *getconf PATH* to determine the shell pathname and update the `#!` script appropriately as it is being installed (for example, with *sed*). For example:

```
#
Installation time script to install correct POSIX shell pathname
#
Get list of paths to check
#
Sifs=$IFS
Sifs_set=${IFS+y}
IFS=:
set -- $(getconf PATH)
if ["$Sifs_set" = y]
then
 IFS=$Sifs
else
 unset IFS
fi
#
Check each path for 'sh'
#
for i
do
 if [-x "${i}/sh]
 then
 Pshell=${i}/sh
```



```

107306 fi
107307 done
107308 #
107309 # This is the list of scripts to update. They should be of the
107310 # form '${name}.source' and will be transformed to '${name}'.
107311 # Each script should begin:
107312 #
107313 # #!INSTALLSHELLPATH
107314 #
107315 scripts="a b c"
107316 #
107317 # Transform each script
107318 #
107319 for i in ${scripts}
107320 do
107321 sed -e "s|INSTALLSHELLPATH|${Pshell}|" < ${i}.source > ${i}
107322 done

```

### 107323 EXAMPLES

- 107324 1. Execute a shell command from a string:
- 107325     sh -c "cat myfile"
- 107326 2. Execute a shell script from a file in the current directory:
- 107327     sh my\_shell\_cmds

### 107328 RATIONALE

107329 The *sh* utility and the *set* special built-in utility share a common set of options.

107330 The name *IFS* was originally an abbreviation of “Input Field Separators”; however, this name is  
 107331 misleading as the *IFS* characters are actually used as field terminators. The KornShell ignores the  
 107332 contents of *IFS* upon entry to the script. A conforming application cannot rely on importing *IFS*.  
 107333 One justification for this, beyond security considerations, is to assist possible future shell  
 107334 compilers. Allowing *IFS* to be imported from the environment prevents many optimizations that  
 107335 might otherwise be performed via dataflow analysis of the script itself.

107336 The text in the STDIN section about non-blocking reads concerns an instance of *sh* that has been  
 107337 invoked, probably by a C-language program, with standard input that has been opened using  
 107338 the `O_NONBLOCK` flag; see `open()` in the System Interfaces volume of POSIX.1-2008. If the shell  
 107339 did not reset this flag, it would immediately terminate because no input data would be available  
 107340 yet and that would be considered the same as end-of-file.

107341 The options associated with a *restricted shell* (command name *rsh* and the `-r` option) were  
 107342 excluded because the standard developers considered that the implied level of security could  
 107343 not be achieved and they did not want to raise false expectations.

107344 On systems that support set-user-ID scripts, a historical trapdoor has been to link a script to the  
 107345 name `-i`. When it is called by a sequence such as:

107346 sh -

107347 or by:

107348 #! usr/bin/sh -

107349 the historical systems have assumed that no option letters follow. Thus, this volume of  
 107350 POSIX.1-2008 allows the single `<hyphen>` to mark the end of the options, in addition to the use

of the regular "--" argument, because it was considered that the older practice was so pervasive. An alternative approach is taken by the KornShell, where real and effective user/group IDs must match for an interactive shell; this behavior is specifically allowed by this volume of POSIX.1-2008.

**Note:** There are other problems with set-user-ID scripts that the two approaches described here do not resolve.

The initialization process for the history file can be dependent on the system start-up files, in that they may contain commands that effectively preempt the user's settings of *HISTFILE* and *HISTSIZE*. For example, function definition commands are recorded in the history file, unless the *set -o nolog* option is set. If the system administrator includes function definitions in some system start-up file called before the *ENV* file, the history file is initialized before the user gets a chance to influence its characteristics. In some historical shells, the history file is initialized just after the *ENV* file has been processed. Therefore, it is implementation-defined whether changes made to *HISTFILE* after the history file has been initialized are effective.

The default messages for the various *MAIL*-related messages are unspecified because they vary across implementations. Typical messages are:

"you have mail\n"

or:

"you have new mail\n"

It is important that the descriptions of command line editing refer to the same shell as that in POSIX.1-2008 so that interactive users can also be application programmers without having to deal with programmatic differences in their two environments. It is also essential that the utility name *sh* be specified because this explicit utility name is too firmly rooted in historical practice of application programs for it to change.

Consideration was given to mandating a diagnostic message when attempting to set *vi*-mode on terminals that do not support command line editing. However, it is not historical practice for the shell to be cognizant of all terminal types and thus be able to detect inappropriate terminals in all cases. Implementations are encouraged to supply diagnostics in this case whenever possible, rather than leaving the user in a state where editing commands work incorrectly.

In early proposals, the KornShell-derived *emacs* mode of command line editing was included, even though the *emacs* editor itself was not. The community of *emacs* proponents was adamant that the full *emacs* editor not be standardized because they were concerned that an attempt to standardize this very powerful environment would encourage vendors to ship strictly conforming versions lacking the extensibility required by the community. The author of the original *emacs* program also expressed his desire to omit the program. Furthermore, there were a number of historical systems that did not include *emacs*, or included it without supporting it, but there were very few that did not include and support *vi*. The shell *emacs* command line editing mode was finally omitted because it became apparent that the KornShell version and the editor being distributed with the GNU system had diverged in some respects. The author of *emacs* requested that the POSIX *emacs* mode either be deleted or have a significant number of unspecified conditions. Although the KornShell author agreed to consider changes to bring the shell into alignment, the standard developers decided to defer specification at that time. At the time, it was assumed that convergence on an acceptable definition would occur for a subsequent draft, but that has not happened, and there appears to be no impetus to do so. In any case, implementations are free to offer additional command line editing modes based on the exact models of editors their users are most comfortable with.

Early proposals had the following list entry in [vi Line Editing Insert Mode](#) (on page 3194):

107398 \ If followed by the *erase* or *kill* character, that character shall be inserted into the input line.  
 107399 Otherwise, the <backslash> itself shall be inserted into the input line.

107400 However, this is not actually a feature of *sh* command line editing insert mode, but one of some  
 107401 historical terminal line drivers. Some conforming implementations continue to do this when the  
 107402 *stty ixten* flag is set.

107403 In interactive shells, SIGTERM is ignored so that `kill 0` does not kill the shell, and SIGINT is +  
 107404 caught so that *wait* is interruptible. If the shell does not ignore SIGTTIN, SIGTTOU, and +  
 107405 SIGTSTP signals when it is interactive and the *-m* option is not in effect, these signals suspend +  
 107406 the shell if it is not a session leader. If it is a session leader, the signals are discarded if they +  
 107407 would stop the process, as required by XSH Section 2.4.3 (on page 490) for orphaned process +  
 107408 groups.

#### 107409 FUTURE DIRECTIONS

107410 None.

#### 107411 SEE ALSO

107412 Chapter 2 (on page 2319), *cd*, *echo*, *exit*, *fc*, *pwd*, *invalid*, *set*, *stty*, *test*, *trap*, *umask*, *vi*

107413 XBD Chapter 8 (on page 173), Section 12.2 (on page 215)

107414 XSH *dup()*, *exec*, *exit()*, *fork()*, *open()*, *pipe()*, *signal()*, *system()*, *ulimit()*, *umask()*, *wait()*

#### 107415 CHANGE HISTORY

107416 First released in Issue 2.

#### 107417 Issue 5

107418 The FUTURE DIRECTIONS section is added.

107419 Text is added to the DESCRIPTION for the Large File Summit proposal.

#### 107420 Issue 6

107421 The Open Group Corrigendum U029/2 is applied, correcting the second SYNOPSIS.

107422 The Open Group Corrigendum U027/3 is applied, correcting a typographical error.

107423 The following new requirements on POSIX implementations derive from alignment with the  
 107424 Single UNIX Specification:

107425 • The option letters derived from the *set* special built-in are also accepted with a leading  
 107426 <plus-sign> ('+').

107427 • Large file extensions are added:

107428 — Pathname expansion does not fail due to the size of a file.

107429 — Shell input and output redirections have an implementation-defined offset maximum  
 107430 that is established in the open file description.

107431 In the ENVIRONMENT VARIABLES section, the text “user’s home directory” is updated to  
 107432 “directory referred to by the *HOME* environment variable”.

107433 Descriptions for the *ENV* and *PWD* environment variables are included to align with the  
 107434 IEEE P1003.2b draft standard.

107435 The normative text is reworded to avoid use of the term “must” for application requirements.

107436 **Issue 7**

- 107437 Austin Group Interpretation 1003.1-2001 #098 is applied, changing the definition of *IFS*.
- 107438 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.
- 107439 Changes to the *pwd* utility and *PWD* environment variable have been made to match the  
107440 changes to the *getcwd()* function made for Austin Group Interpretation 1003.1-2001 #140.
- 107441 Minor editorial changes are made to the User Portability Utilities option shading. No normative  
107442 changes are implied.
- 107443 Minor changes are made to the install script example in the APPLICATION USAGE section.
- 107444 POSIX.1-2008, Technical Corrigendum 1, XCU/TC1-2008/0137 [152], XCU/TC1-2008/0138 +  
107445 [347], XCU/TC1-2008/0139 [347], XCU/TC1-2008/0140 [347], XCU/TC1-2008/0141 [299], and +  
107446 XCU/TC1-2008/0142 [347] are applied.

107447 **NAME**

107448           sleep — suspend execution for an interval

107449 **SYNOPSIS**107450           sleep *time*107451 **DESCRIPTION**

107452           The *sleep* utility shall suspend execution for at least the integral number of seconds specified by  
 107453           the *time* operand.

107454 **OPTIONS**

107455           None.

107456 **OPERANDS**

107457           The following operand shall be supported:

107458           *time*           A non-negative decimal integer specifying the number of seconds for which to  
 107459           suspend execution.

107460 **STDIN**

107461           Not used.

107462 **INPUT FILES**

107463           None.

107464 **ENVIRONMENT VARIABLES**107465           The following environment variables shall affect the execution of *sleep*:

107466           *LANG*           Provide a default value for the internationalization variables that are unset or null.  
 107467                               (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 107468                               variables used to determine the values of locale categories.)

107469           *LC\_ALL*       If set to a non-empty string value, override the values of all the other  
 107470           internationalization variables.

107471           *LC\_CTYPE*   Determine the locale for the interpretation of sequences of bytes of text data as  
 107472           characters (for example, single-byte as opposed to multi-byte characters in  
 107473           arguments).

107474           *LC\_MESSAGES*

107475                           Determine the locale that should be used to affect the format and contents of  
 107476           diagnostic messages written to standard error.

107477 XSI           *NLSPATH*   Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

107478 **ASYNCHRONOUS EVENTS**107479           If the *sleep* utility receives a SIGALRM signal, one of the following actions shall be taken:

- 107480           1. Terminate normally with a zero exit status.
- 107481           2. Effectively ignore the signal.
- 107482           3. Provide the default behavior for signals described in the ASYNCHRONOUS EVENTS  
 107483           section of [Section 1.4](#) (on page 2310). This could include terminating with a non-zero exit  
 107484           status.

107485           The *sleep* utility shall take the standard action for all other signals.

107486 **STDOUT**

107487 Not used.

107488 **STDERR**

107489 The standard error shall be used only for diagnostic messages.

107490 **OUTPUT FILES**

107491 None.

107492 **EXTENDED DESCRIPTION**

107493 None.

107494 **EXIT STATUS**

107495 The following exit values shall be returned:

107496 0 The execution was successfully suspended for at least *time* seconds, or a SIGALRM signal  
107497 was received. See the ASYNCHRONOUS EVENTS section.

107498 &gt;0 An error occurred.

107499 **CONSEQUENCES OF ERRORS**

107500 Default.

107501 **APPLICATION USAGE**

107502 None.

107503 **EXAMPLES**107504 The *sleep* utility can be used to execute a command after a certain amount of time, as in:107505 `(sleep 105; command) &`

107506 or to execute a command every so often, as in:

```
107507 while true
107508 do
107509 command
107510 sleep 37
107511 done
```

107512 **RATIONALE**

107513 The exit status is allowed to be zero when *sleep* is interrupted by the SIGALRM signal because  
107514 most implementations of this utility rely on the arrival of that signal to notify them that the  
107515 requested finishing time has been successfully attained. Such implementations thus do not  
107516 distinguish this situation from the successful completion case. Other implementations are  
107517 allowed to catch the signal and go back to sleep until the requested time expires or to provide  
107518 the normal signal termination procedures.

107519 As with all other utilities that take integral operands and do not specify subranges of allowed  
107520 values, *sleep* is required by this volume of POSIX.1-2008 to deal with *time* requests of up to  
107521 2 147 483 647 seconds. This may mean that some implementations have to make multiple calls to  
107522 the delay mechanism of the underlying operating system if its argument range is less than this.

107523 **FUTURE DIRECTIONS**

107524 None.

107525 **SEE ALSO**107526 *wait*

107527 XBD Chapter 8 (on page 173)

107528 XSH *alarm()*, *sleep()*

107529 **CHANGE HISTORY**  
107530 First released in Issue 2.

107531 **NAME**107532 `sort` — sort, merge, or sequence check text files107533 **SYNOPSIS**107534 `sort [-m] [-o output] [-bdfinru] [-t char] [-k keydef]... [file...]`107535 `sort [-c|-C] [-bdfinru] [-t char] [-k keydef] [file]`107536 **DESCRIPTION**107537 The *sort* utility shall perform one of the following functions:

- 107538 1. Sort lines of all the named files together and write the result to the specified output.
- 107539 2. Merge lines of all the named (presorted) files together and write the result to the specified output.
- 107540 3. Check that a single input file is correctly presorted.

107542 Comparisons shall be based on one or more sort keys extracted from each line of input (or, if no  
 107543 sort keys are specified, the entire line up to, but not including, the terminating <newline>), and  
 107544 shall be performed using the collating sequence of the current locale.

107545 **OPTIONS**

107546 The *sort* utility shall conform to XBD [Section 12.2](#) (on page 215), except for Guideline 9, and the  
 107547 `-k keydef` option should follow the `-b`, `-d`, `-f`, `-i`, `-n`, and `-r` options. In addition, `'+'` may be  
 107548 recognized as an option delimiter as well as `'-'`.

107549 The following options shall be supported:

- 107550 `-c` Check that the single input file is ordered as specified by the arguments and the  
 107551 collating sequence of the current locale. Output shall not be sent to standard  
 107552 output. The exit code shall indicate whether or not disorder was detected or an  
 107553 error occurred. If disorder (or, with `-u`, a duplicate key) is detected, a warning  
 107554 message shall be sent to standard error indicating where the disorder or duplicate  
 107555 key was found.
- 107556 `-C` Same as `-c`, except that a warning message shall not be sent to standard error if  
 107557 disorder or, with `-u`, a duplicate key is detected.
- 107558 `-m` Merge only; the input file shall be assumed to be already sorted.
- 107559 `-o output` Specify the name of an output file to be used instead of the standard output. This  
 107560 file can be the same as one of the input *files*.
- 107561 `-u` Unique: suppress all but one in each set of lines having equal keys. If used with  
 107562 the `-c` option, check that there are no lines with duplicate keys, in addition to  
 107563 checking that the input file is sorted.

107564 The following options shall override the default ordering rules. When ordering options appear  
 107565 independent of any key field specifications, the requested field ordering rules shall be applied  
 107566 globally to all sort keys. When attached to a specific key (see `-k`), the specified ordering options  
 107567 shall override all global ordering options for that key.

- 107568 `-d` Specify that only <blank> characters and alphanumeric characters, according to  
 107569 the current setting of *LC\_CTYPE*, shall be significant in comparisons. The behavior  
 107570 is undefined for a sort key to which `-i` or `-n` also applies.
- 107571 `-f` Consider all lowercase characters that have uppercase equivalents, according to  
 107572 the current setting of *LC\_CTYPE*, to be the uppercase equivalent for the purposes  
 107573 of comparison.



|        |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|--------|------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 107574 | <b>-i</b>        | Ignore all characters that are non-printable, according to the current setting of <i>LC_CTYPE</i> . The behavior is undefined for a sort key for which <b>-n</b> also applies.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| 107575 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 107576 | <b>-n</b>        | Restrict the sort key to an initial numeric string, consisting of optional <blank> characters, optional minus-sign, and zero or more digits with an optional radix character and thousands separators (as defined in the current locale), which shall be sorted by arithmetic value. An empty digit string shall be treated as zero. Leading zeros and signs on zeros shall not affect ordering.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 107577 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 107578 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 107579 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 107580 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 107581 | <b>-r</b>        | Reverse the sense of comparisons.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 107582 |                  | The treatment of field separators can be altered using the options:                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 107583 | <b>-b</b>        | Ignore leading <blank> characters when determining the starting and ending positions of a restricted sort key. If the <b>-b</b> option is specified before the first <b>-k</b> option, it shall be applied to all <b>-k</b> options. Otherwise, the <b>-b</b> option can be attached independently to each <b>-k</b> <i>field_start</i> or <i>field_end</i> option-argument (see below).                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| 107584 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 107585 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 107586 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 107587 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 107588 | <b>-t char</b>   | Use <i>char</i> as the field separator character; <i>char</i> shall not be considered to be part of a field (although it can be included in a sort key). Each occurrence of <i>char</i> shall be significant (for example, <char><char> delimits an empty field). If <b>-t</b> is not specified, <blank> characters shall be used as default field separators; each maximal non-empty sequence of <blank> characters that follows a non-<blank> shall be a field separator.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 107589 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 107590 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 107591 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 107592 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 107593 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 107594 |                  | Sort keys can be specified using the options:                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 107595 | <b>-k keydef</b> | The <i>keydef</i> argument is a restricted sort key field definition. The format of this definition is:                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 107596 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 107597 |                  | <i>field_start</i> [ <i>type</i> ][, <i>field_end</i> [ <i>type</i> ]]                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 107598 |                  | where <i>field_start</i> and <i>field_end</i> define a key field restricted to a portion of the line (see the EXTENDED DESCRIPTION section), and <i>type</i> is a modifier from the list of characters 'b', 'd', 'f', 'i', 'n', 'r'. The 'b' modifier shall behave like the <b>-b</b> option, but shall apply only to the <i>field_start</i> or <i>field_end</i> to which it is attached. The other modifiers shall behave like the corresponding options, but shall apply only to the key field to which they are attached; they shall have this effect if specified with <i>field_start</i> , <i>field_end</i> , or both. If any modifier is attached to a <i>field_start</i> or to a <i>field_end</i> , no option shall apply to either. Implementations shall support at least nine occurrences of the <b>-k</b> option, which shall be significant in command line order. If no <b>-k</b> option is specified, a default sort key of the entire line shall be used. |
| 107599 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 107600 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 107601 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 107602 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 107603 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 107604 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 107605 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 107606 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 107607 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 107608 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 107609 |                  | When there are multiple key fields, later keys shall be compared only after all earlier keys compare equal. Except when the <b>-u</b> option is specified, lines that otherwise compare equal shall be ordered as if none of the options <b>-d</b> , <b>-f</b> , <b>-i</b> , <b>-n</b> , or <b>-k</b> were present (but with <b>-r</b> still in effect, if it was specified) and with all bytes in the lines significant to the comparison. The order in which lines that still compare equal are written is unspecified.                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 107610 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 107611 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 107612 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 107613 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 107614 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 107615 | <b>OPERANDS</b>  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 107616 |                  | The following operand shall be supported:                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 107617 | <i>file</i>      | A pathname of a file to be sorted, merged, or checked. If no <i>file</i> operands are specified, or if a <i>file</i> operand is '-', the standard input shall be used.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 107618 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |

107619 **STDIN**

107620 The standard input shall be used only if no *file* operands are specified, or if a *file* operand is *'-'*.  
 107621 See the INPUT FILES section.

107622 **INPUT FILES**

107623 The input files shall be text files, except that the *sort* utility shall add a <newline> to the end of a  
 107624 file ending with an incomplete last line.

107625 **ENVIRONMENT VARIABLES**

107626 The following environment variables shall affect the execution of *sort*:

107627 *LANG* Provide a default value for the internationalization variables that are unset or null.  
 107628 (See XBD Section 8.2 (on page 174) for the precedence of internationalization  
 107629 variables used to determine the values of locale categories.)

107630 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
 107631 internationalization variables.

107632 *LC\_COLLATE*

107633 Determine the locale for ordering rules.

107634 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
 107635 characters (for example, single-byte as opposed to multi-byte characters in  
 107636 arguments and input files) and the behavior of character classification for the *-b*,  
 107637 *-d*, *-f*, *-i*, and *-n* options.

107638 *LC\_MESSAGES*

107639 Determine the locale that should be used to affect the format and contents of  
 107640 diagnostic messages written to standard error.

107641 *LC\_NUMERIC*

107642 Determine the locale for the definition of the radix character and thousands  
 107643 separator for the *-n* option.

107644 XSI *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

107645 **ASYNCHRONOUS EVENTS**

107646 Default.

107647 **STDOUT**

107648 Unless the *-o* or *-c* options are in effect, the standard output shall contain the sorted input.

107649 **STDERR**

107650 The standard error shall be used for diagnostic messages. When *-c* is specified, if disorder is  
 107651 detected (or if *-u* is also specified and a duplicate key is detected), a message shall be written to  
 107652 the standard error which identifies the input line at which disorder (or a duplicate key) was  
 107653 detected. A warning message about correcting an incomplete last line of an input file may be  
 107654 generated, but need not affect the final exit status.

107655 **OUTPUT FILES**

107656 If the *-o* option is in effect, the sorted input shall be written to the file *output*.

107657 **EXTENDED DESCRIPTION**

107658 The notation:

107659 *-k field\_start[type][, field\_end[type]]*

107660 shall define a key field that begins at *field\_start* and ends at *field\_end* inclusive, unless *field\_start*  
 107661 falls beyond the end of the line or after *field\_end*, in which case the key field is empty. A missing  
 107662 *field\_end* shall mean the last character of the line.

A field comprises a maximal sequence of non-separating characters and, in the absence of option **-t**, any preceding field separator.

The *field\_start* portion of the *keydef* option-argument shall have the form:

*field\_number*[.*first\_character*]

Fields and characters within fields shall be numbered starting with 1. The *field\_number* and *first\_character* pieces, interpreted as positive decimal integers, shall specify the first character to be used as part of a sort key. If *first\_character* is omitted, it shall refer to the first character of the field.

The *field\_end* portion of the *keydef* option-argument shall have the form:

*field\_number*[.*last\_character*]

The *field\_number* shall be as described above for *field\_start*. The *last\_character* piece, interpreted as a non-negative decimal integer, shall specify the last character to be used as part of the sort key. If *last\_character* evaluates to zero or *last\_character* is omitted, it shall refer to the last character of the field specified by *field\_number*.

If the **-b** option or **b** type modifier is in effect, characters within a field shall be counted from the first non-<blank> in the field. (This shall apply separately to *first\_character* and *last\_character*.)

## EXIT STATUS

The following exit values shall be returned:

- 0 All input files were output successfully, or **-c** was specified and the input file was correctly sorted.
- 1 Under the **-c** option, the file was not ordered as specified, or if the **-c** and **-u** options were both specified, two input lines were found with equal keys.
- >1 An error occurred.

## CONSEQUENCES OF ERRORS

Default.

## APPLICATION USAGE

The default value for **-t**, <blank>, has different properties from, for example, **-t"<space>"**. If a line contains:

<space><space>foo

the following treatment would occur with default separation as opposed to specifically selecting a <space>:

| Field | Default           | <b>-t "&lt;space&gt;"</b> |
|-------|-------------------|---------------------------|
| 1     | <space><space>foo | empty                     |
| 2     | empty             | empty                     |
| 3     | empty             | foo                       |

The leading field separator itself is included in a field when **-t** is not used. For example, this command returns an exit status of zero, meaning the input was already sorted:

```
sort -c -k 2 <<eof
y<tab>b
x<space>a
eof
```

(assuming that a <tab> precedes the <space> in the current collating sequence). The field

separator is not included in a field when it is explicitly set via **-t**. This is historical practice and allows usage such as:

```
sort -t "|" -k 2n <<eof
Atlanta|425022|Georgia
Birmingham|284413|Alabama
Columbia|100385|South Carolina
eof
```

where the second field can be correctly sorted numerically without regard to the non-numeric field separator.

The wording in the OPTIONS section clarifies that the **-b**, **-d**, **-f**, **-i**, **-n**, and **-r** options have to come before the first sort key specified if they are intended to apply to all specified keys. The way it is described in this volume of POSIX.1-2008 matches historical practice, not historical documentation. The results are unspecified if these options are specified after a **-k** option.

The **-f** option might not work as expected in locales where there is not a one-to-one mapping between an uppercase and a lowercase letter.

## EXAMPLES

1. The following command sorts the contents of **infile** with the second field as the sort key:

```
sort -k 2,2 infile
```

2. The following command sorts, in reverse order, the contents of **infile1** and **infile2**, placing the output in **outfile** and using the second character of the second field as the sort key (assuming that the first character of the second field is the field separator):

```
sort -r -o outfile -k 2.2,2.2 infile1 infile2
```

3. The following command sorts the contents of **infile1** and **infile2** using the second non-**<blank>** of the second field as the sort key:

```
sort -k 2.2b,2.2b infile1 infile2
```

4. The following command prints the System V password file (user database) sorted by the numeric user ID (the third **<colon>**-separated field):

```
sort -t : -k 3,3n /etc/passwd
```

5. The following command prints the lines of the already sorted file **infile**, suppressing all but one occurrence of lines having the same third field:

```
sort -um -k 3.1,3.0 infile
```

## RATIONALE

Examples in some historical documentation state that options **-um** with one input file keep the first in each set of lines with equal keys. This behavior was deemed to be an implementation artifact and was not standardized.

The **-z** option was omitted; it is not standard practice on most systems and is inconsistent with using *sort* to sort several files individually and then merge them together. The text concerning **-z** in historical documentation appeared to require implementations to determine the proper buffer length during the sort phase of operation, but not during the merge.

The **-y** option was omitted because of non-portability. The **-M** option, present in System V, was omitted because of non-portability in international usage.

An undocumented **-T** option exists in some implementations. It is used to specify a directory for

intermediate files. Implementations are encouraged to support the use of the *TMPDIR* environment variable instead of adding an option to support this functionality.

The **-k** option was added to satisfy two objections. First, the zero-based counting used by *sort* is not consistent with other utility conventions. Second, it did not meet syntax guideline requirements.

Historical documentation indicates that “setting **-n** implies **-b**”. The description of **-n** already states that optional leading <blank>s are tolerated in doing the comparison. If **-b** is enabled, rather than implied, by **-n**, this has unusual side-effects. When a character offset is used in a column of numbers (for example, to sort modulo 100), that offset is measured relative to the most significant digit, not to the column. Based upon a recommendation from the author of the original *sort* utility, the **-b** implication has been omitted from this volume of POSIX.1-2008, and an application wishing to achieve the previously mentioned side-effects has to code the **-b** flag explicitly.

Earlier versions of this standard allowed the **-o** option to appear after operands. Historical practice allowed all options to be interspersed with operands. This version of the standard allows implementations to accept options after operands but conforming applications should not use this form.

Earlier versions of this standard also allowed the *-number* and *+number* options. These options are no longer specified by POSIX.1-2008 but may be present in some implementations.

Historical implementations produced a message on standard error when **-c** was specified and disorder was detected, and when **-c** and **-u** were specified and a duplicate key was detected. An earlier version of this standard contained wording that did not make it clear that this message was allowed and some implementations removed this message to be sure that they conformed to the standard’s requirements. Confronted with this difference in behavior, interactive users that wanted to be sure that they got visual feedback instead of just exit code 1 could have used a command like:

```
sort -c file || echo disorder
```

whether or not the *sort* utility provided a message in this case. But, it was not easy for a user to find where the disorder or duplicate key occurred on implementations that do not produce a message, especially when some parts of the input line were not part of the key and when one or more of the **-b**, **-d**, **-f**, **-i**, **-n**, or **-r** options or *keydef* type modifiers were in use. POSIX.1-2008 requires a message to be produced in this case. POSIX.1-2008 also contains the **-C** option giving users the ability to choose either behavior.

When a disorder or duplicate is found when the **-c** option is specified, some implementations print a message containing the first line that is out of order or contains a duplicate key; others print a message specifying the line number of the offending line. This standard allows either type of message.

#### FUTURE DIRECTIONS

None.

#### SEE ALSO

*comm*, *join*, *uniq*

XBD Chapter 8 (on page 173), Section 12.2 (on page 215)

XSH *toupper()*

**CHANGE HISTORY**

107790  
107791 First released in Issue 2.

**Issue 6**

107792  
107793 IEEE PASC Interpretation 1003.2 #174 is applied, updating the DESCRIPTION of comparisons.

107794 IEEE PASC Interpretation 1003.2 #168 is applied.

**Issue 7**

107795  
107796 Austin Group Interpretation 1003.1-2001 #027 is applied, clarifying that Guideline 9 of the Utility  
107797 Syntax Guidelines does not apply and noting that ' +' may be recognized as an option delimiter.

107798 Austin Group Interpretation 1003.1-2001 #120 is applied, clarifying the use of the -c option and  
107799 introducing the -C option.

107800 XCU-ERN-81 is applied, modifying the description of the -i option.

107801 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

107802 **NAME**107803 `split` — split files into pieces107804 **SYNOPSIS**107805 `split [-l line_count] [-a suffix_length] [file[name]]`107806 `split -b n[k|m] [-a suffix_length] [file[name]]`107807 **DESCRIPTION**

107808 The *split* utility shall read an input file and write one or more output files. The default size of  
 107809 each output file shall be 1 000 lines. The size of the output files can be modified by specification  
 107810 of the `-b` or `-l` options. Each output file shall be created with a unique suffix. The suffix shall  
 107811 consist of exactly *suffix\_length* lowercase letters from the POSIX locale. The letters of the suffix  
 107812 shall be used as if they were a base-26 digit system, with the first suffix to be created consisting  
 107813 of all 'a' characters, the second with a 'b' replacing the last 'a', and so on, until a name of all  
 107814 'z' characters is created. By default, the names of the output files shall be 'x', followed by a  
 107815 two-character suffix from the character set as described above, starting with "aa", "ab", "ac",  
 107816 and so on, and continuing until the suffix "zz", for a maximum of 676 files.

107817 If the number of files required exceeds the maximum allowed by the suffix length provided,  
 107818 such that the last allowable file would be larger than the requested size, the *split* utility shall fail  
 107819 after creating the last file with a valid suffix; *split* shall not delete the files it created with valid  
 107820 suffixes. If the file limit is not exceeded, the last file created shall contain the remainder of the  
 107821 input file, and may be smaller than the requested size.

107822 **OPTIONS**107823 The *split* utility shall conform to XBD [Section 12.2](#) (on page 215).

107824 The following options shall be supported:

107825 `-a suffix_length`

107826 Use *suffix\_length* letters to form the suffix portion of the filenames of the split file. If  
 107827 `-a` is not specified, the default suffix length shall be two. If the sum of the *name*  
 107828 operand and the *suffix\_length* option-argument would create a filename exceeding  
 107829 {NAME\_MAX} bytes, an error shall result; *split* shall exit with a diagnostic message  
 107830 and no files shall be created.

107831 `-b n` Split a file into pieces *n* bytes in size.107832 `-b nk` Split a file into pieces *n*\*1 024 bytes in size.107833 `-b nm` Split a file into pieces *n*\*1 048 576 bytes in size.

107834 `-l line_count` Specify the number of lines in each resulting file piece. The *line\_count* argument is  
 107835 an unsigned decimal integer. The default is 1 000. If the input does not end with a  
 107836 <newline>, the partial line shall be included in the last output file.

107837 **OPERANDS**

107838 The following operands shall be supported:

107839 *file* The pathname of the ordinary file to be split. If no input file is given or *file* is '-',  
 107840 the standard input shall be used.

107841 *name* The prefix to be used for each of the files resulting from the split operation. If no  
 107842 *name* argument is given, 'x' shall be used as the prefix of the output files. The  
 107843 combined length of the basename of *prefix* and *suffix\_length* cannot exceed  
 107844 {NAME\_MAX} bytes. See the OPTIONS section.

107845 **STDIN**

107846 See the INPUT FILES section.

107847 **INPUT FILES**

107848 Any file can be used as input.

107849 **ENVIRONMENT VARIABLES**107850 The following environment variables shall affect the execution of *split*:

107851 **LANG** Provide a default value for the internationalization variables that are unset or null.  
 107852 (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 107853 variables used to determine the values of locale categories.)

107854 **LC\_ALL** If set to a non-empty string value, override the values of all the other  
 107855 internationalization variables.

107856 **LC\_CTYPE** Determine the locale for the interpretation of sequences of bytes of text data as  
 107857 characters (for example, single-byte as opposed to multi-byte characters in  
 107858 arguments and input files).

107859 **LC\_MESSAGES**

107860 Determine the locale that should be used to affect the format and contents of  
 107861 diagnostic messages written to standard error.

107862 **XSI NLSPATH** Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

107863 **ASYNCHRONOUS EVENTS**

107864 Default.

107865 **STDOUT**

107866 Not used.

107867 **STDERR**

107868 The standard error shall be used only for diagnostic messages.

107869 **OUTPUT FILES**

107870 The output files contain portions of the original input file; otherwise, unchanged.

107871 **EXTENDED DESCRIPTION**

107872 None.

107873 **EXIT STATUS**

107874 The following exit values shall be returned:

107875 0 Successful completion.

107876 &gt;0 An error occurred.

107877 **CONSEQUENCES OF ERRORS**

107878 Default.



## 107879 APPLICATION USAGE

107880 None.

## 107881 EXAMPLES

107882 In the following examples **foo** is a text file that contains 5 000 lines.107883 1. Create five files, **xaa**, **xab**, **xac**, **xad**, and **xae**:107884 `split foo`107885 2. Create five files, but the suffixed portion of the created files consists of three letters, **xaaa**,  
107886 **xaab**, **xaac**, **xaad**, and **xaae**:107887 `split -a 3 foo`107888 3. Create three files with four-letter suffixes and a supplied prefix, **bar\_aaaa**, **bar\_aaab**, and  
107889 **bar\_aaac**:107890 `split -a 4 -l 2000 foo bar_`107891 4. Create as many files as are necessary to contain at most 20\*1 024 bytes, each with the  
107892 default prefix of **x** and a five-letter suffix:107893 `split -a 5 -b 20k foo`

## 107894 RATIONALE

107895 The **-b** option was added to provide a mechanism for splitting files other than by lines. While  
107896 most uses of the **-b** option are for transmitting files over networks, some believed it would have  
107897 additional uses.107898 The **-a** option was added to overcome the limitation of being able to create only 676 files.107899 Consideration was given to deleting this utility, using the rationale that the functionality  
107900 provided by this utility is available via the *csplit* utility (see *csplit*). Upon reconsideration of the  
107901 purpose of the User Portability Utilities option, it was decided to retain both this utility and the  
107902 *csplit* utility because users use both utilities and have historical expectations of their behavior.  
107903 Furthermore, the splitting on byte boundaries in *split* cannot be duplicated with the historical  
107904 *csplit*.107905 The text “*split* shall not delete the files it created with valid suffixes” would normally be  
107906 assumed, but since the related utility, *csplit*, does delete files under some circumstances, the  
107907 historical behavior of *split* is made explicit to avoid misinterpretation.107908 Earlier versions of this standard allowed a *-line\_count* option. This form is no longer specified by  
107909 POSIX.1-2008 but may be present in some implementations.

## 107910 FUTURE DIRECTIONS

107911 None.

## 107912 SEE ALSO

107913 *csplit*107914 XBD [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 215)

## 107915 CHANGE HISTORY

107916 First released in Issue 2.

## 107917 Issue 6

107918 This utility is marked as part of the User Portability Utilities option.

107919 The APPLICATION USAGE section is added.

107920 The obsolescent SYNOPSIS is removed.

107921 **Issue 7**

107922 Austin Group Interpretation 1003.1-2001 #027 is applied.

107923 The *split* utility is moved from the User Portability Utilities option to the Base. User Portability  
107924 Utilities is now an option for interactive utilities.

107925 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

107926 **NAME**

107927 strings — find printable strings in files

107928 **SYNOPSIS**107929 strings [-a] [-t *format*] [-n *number*] [*file...*]107930 **DESCRIPTION**

107931 The *strings* utility shall look for printable strings in regular files and shall write those strings to  
 107932 standard output. A printable string is any sequence of four (by default) or more printable  
 107933 characters terminated by a <newline> or NUL character. Additional implementation-defined  
 107934 strings may be written; see *localedef*.

107935 If the first argument is '-', the results are unspecified.

107936 **OPTIONS**

107937 The *strings* utility shall conform to XBD [Section 12.2](#) (on page 215), except for the unspecified  
 107938 usage of '-'.

107939 The following options shall be supported:

107940 -a Scan files in their entirety. If -a is not specified, it is implementation-defined what  
 107941 portion of each file is scanned for strings.

107942 -n *number* Specify the minimum string length, where the *number* argument is a positive  
 107943 decimal integer. The default shall be 4.

107944 -t *format* Write each string preceded by its byte offset from the start of the file. The format  
 107945 shall be dependent on the single character used as the *format* option-argument:

107946 d The offset shall be written in decimal.

107947 o The offset shall be written in octal.

107948 x The offset shall be written in hexadecimal.

107949 **OPERANDS**

107950 The following operand shall be supported:

107951 *file* A pathname of a regular file to be used as input. If no *file* operand is specified, the  
 107952 *strings* utility shall read from the standard input.

107953 **STDIN**

107954 See the INPUT FILES section.

107955 **INPUT FILES**

107956 The input files named by the utility arguments or the standard input shall be regular files of any  
 107957 format.

107958 **ENVIRONMENT VARIABLES**107959 The following environment variables shall affect the execution of *strings*:

107960 LANG Provide a default value for the internationalization variables that are unset or null.  
 107961 (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 107962 variables used to determine the values of locale categories.)

107963 LC\_ALL If set to a non-empty string value, override the values of all the other  
 107964 internationalization variables.

107965 LC\_CTYPE Determine the locale for the interpretation of sequences of bytes of text data as  
 107966 characters (for example, single-byte as opposed to multi-byte characters in  
 107967 arguments and input files) and to identify printable strings.

107968 *LC\_MESSAGES*  
 107969 Determine the locale that should be used to affect the format and contents of  
 107970 diagnostic messages written to standard error.

107971 XSI *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

107972 **ASYNCHRONOUS EVENTS**  
 107973 Default.

107974 **STDOUT**  
 107975 Strings found shall be written to the standard output, one per line.  
 107976 When the **-t** option is not specified, the format of the output shall be:  
 107977 "%s", <string>  
 107978 With the **-t o** option, the format of the output shall be:  
 107979 "%o %s", <byte offset>, <string>  
 107980 With the **-t x** option, the format of the output shall be:  
 107981 "%x %s", <byte offset>, <string>  
 107982 With the **-t d** option, the format of the output shall be:  
 107983 "%d %s", <byte offset>, <string>

107984 **STDERR**  
 107985 The standard error shall be used only for diagnostic messages.

107986 **OUTPUT FILES**  
 107987 None.

107988 **EXTENDED DESCRIPTION**  
 107989 None.

107990 **EXIT STATUS**  
 107991 The following exit values shall be returned:  
 107992 0 Successful completion.  
 107993 >0 An error occurred.

107994 **CONSEQUENCES OF ERRORS**  
 107995 Default.

107996 **APPLICATION USAGE**  
 107997 By default the data area (as opposed to the text, "bss", or header areas) of a binary executable  
 107998 file is scanned. Implementations document which areas are scanned.  
 107999 Some historical implementations do not require NUL or <newline> terminators for strings to  
 108000 permit those languages that do not use NUL as a string terminator to have their strings written.

108001 **EXAMPLES**  
 108002 None.

108003 **RATIONALE**  
 108004 Apart from rationalizing the option syntax and slight difficulties with object and executable  
 108005 binary files, *strings* is specified to match historical practice closely. The **-a** and **-n** options were  
 108006 introduced to replace the non-conforming **-** and **-number** options. These options are no longer  
 108007 specified by POSIX.1-2008 but may be present in some implementations.

- 108008 The **-o** option historically means different things on different implementations. Some use it to  
108009 mean “*offset* in decimal”, while others use it as “*offset* in octal”. Instead of trying to decide which  
108010 way would be least objectionable, the **-t** option was added. It was originally named **-O** to mean  
108011 “*offset*”, but was changed to **-t** to be consistent with *od*.
- 108012 The ISO C standard function *isprint()* is restricted to a domain of **unsigned char**. This volume of  
108013 POSIX.1-2008 requires implementations to write strings as defined by the current locale.
- 108014 **FUTURE DIRECTIONS**
- 108015 None.
- 108016 **SEE ALSO**
- 108017 *localedef*, *nm*
- 108018 XBD [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 215)
- 108019 **CHANGE HISTORY**
- 108020 First released in Issue 4.
- 108021 **Issue 6**
- 108022 This utility is marked as part of the User Portability Utilities option.
- 108023 The obsolescent SYNOPSIS is removed.
- 108024 The normative text is reworded to avoid use of the term “must” for application requirements.
- 108025 **Issue 7**
- 108026 Austin Group Interpretation 1003.1-2001 #027 is applied, clarifying the behavior if the first  
108027 argument is ‘-’.
- 108028 The *strings* utility is moved from the User Portability Utilities option to the Base. User  
108029 Portability Utilities is now an option for interactive utilities.
- 108030 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

108031 **NAME**108032 strip — remove unnecessary information from strippable files (**DEVELOPMENT**)108033 **SYNOPSIS**

108034 SD strip file...

108035 **DESCRIPTION**

108036 XSI A strippable file is defined as a relocatable, object, or executable file. On XSI-conformant  
 108037 systems, a strippable file can also be an archive of object or relocatable files.

108038 The *strip* utility shall remove from strippable files named by the *file* operands any information  
 108039 the implementor deems unnecessary for execution of those files. The nature of that information  
 108040 is unspecified. The effect of *strip* on object and executable files shall be similar to the use of the  
 108041 XSI *-s* option to *c99* or *fort77*. The effect of *strip* on an archive of object files shall be similar to the  
 108042 use of the *-s* option to *c99* or *fort77* for each object file in the archive.

108043 **OPTIONS**

108044 None.

108045 **OPERANDS**

108046 The following operand shall be supported:

108047 *file* A pathname referring to a strippable file.

108048 **STDIN**

108049 Not used.

108050 **INPUT FILES**

108051 The input files shall be in the form of strippable files successfully produced by any compiler  
 108052 XSI defined by this volume of POSIX.1-2008 or produced by creating or updating an archive of such  
 108053 files using the *ar* utility.

108054 **ENVIRONMENT VARIABLES**108055 The following environment variables shall affect the execution of *strip*:

108056 *LANG* Provide a default value for the internationalization variables that are unset or null.  
 108057 (See XBD Section 8.2 (on page 174) for the precedence of internationalization  
 108058 variables used to determine the values of locale categories.)

108059 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
 108060 internationalization variables.

108061 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
 108062 characters (for example, single-byte as opposed to multi-byte characters in  
 108063 arguments).

108064 *LC\_MESSAGES*

108065 Determine the locale that should be used to affect the format and contents of  
 108066 diagnostic messages written to standard error.

108067 XSI *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

108068 **ASYNCHRONOUS EVENTS**

108069 Default.

108070 **STDOUT**

108071 Not used.

**108072 STDERR**

108073 The standard error shall be used only for diagnostic messages.

**108074 OUTPUT FILES**

108075 The *strip* utility shall produce strippable files of unspecified format.

**108076 EXTENDED DESCRIPTION**

108077 None.

**108078 EXIT STATUS**

108079 The following exit values shall be returned:

108080 0 Successful completion.

108081 >0 An error occurred.

**108082 CONSEQUENCES OF ERRORS**

108083 Default.

**108084 APPLICATION USAGE**

108085 None.

**108086 EXAMPLES**

108087 None.

**108088 RATIONALE**

108089 Historically, this utility has been used to remove the symbol table from a strippable file. It was  
108090 included since it is known that the amount of symbolic information can amount to several  
108091 megabytes; the ability to remove it in a portable manner was deemed important, especially for  
108092 smaller systems.

108093 The behavior of *strip* on object and executable files is said to be the same as the *-s* option to a  
108094 compiler. While the end result is essentially the same, it is not required to be identical.

108095 XSI-conformant systems support use of *strip* on archive files containing object files or relocatable  
108096 files.

**108097 FUTURE DIRECTIONS**

108098 None.

**108099 SEE ALSO**

108100 *ar*, *c99*, *fort77*

108101 XBD Chapter 8 (on page 173)

**108102 CHANGE HISTORY**

108103 First released in Issue 2.

**108104 Issue 6**

108105 This utility is marked as part of the Software Development Utilities option.

**108106 Issue 7**

108107 Austin Group Interpretation 1003.1-2001 #103 is applied.

108108 **NAME**

108109       stty — set the options for a terminal

108110 **SYNOPSIS**

108111       stty [-a|-g]

108112       stty *operand*...108113 **DESCRIPTION**

108114       The *stty* utility shall set or report on terminal I/O characteristics for the device that is its  
 108115       standard input. Without options or operands specified, it shall report the settings of certain  
 108116       characteristics, usually those that differ from implementation-defined defaults. Otherwise, it  
 108117       shall modify the terminal state according to the specified operands. Detailed information about  
 108118       the modes listed in the first five groups below are described in XBD [Chapter 11](#) (on page 199).  
 108119       Operands in the Combination Modes group (see [Combination Modes](#), on page 3231) are  
 108120       implemented using operands in the previous groups. Some combinations of operands are  
 108121       mutually-exclusive on some terminal types; the results of using such combinations are  
 108122       unspecified.

108123       Typical implementations of this utility require a communications line configured to use the  
 108124       **termios** interface defined in the System Interfaces volume of POSIX.1-2008. On systems where  
 108125       none of these lines are available, and on lines not currently configured to support the **termios**  
 108126       interface, some of the operands need not affect terminal characteristics.

108127 **OPTIONS**108128       The *stty* utility shall conform to XBD [Section 12.2](#) (on page 215).

108129       The following options shall be supported:

108130       **-a**           Write to standard output all the current settings for the terminal.

108131       **-g**           Write to standard output all the current settings in an unspecified form that can be  
 108132       used as arguments to another invocation of the *stty* utility on the same system. The  
 108133       form used shall not contain any characters that would require quoting to avoid  
 108134       word expansion by the shell; see [Section 2.6](#) (on page 2327).

108135 **OPERANDS**

108136       The following operands shall be supported to set the terminal characteristics.

108137       **Control Modes**

108138       **parenb** (**-parenb**) Enable (disable) parity generation and detection. This shall have the effect of  
 108139       setting (not setting) PARENB in the **termios** *c\_flag* field, as defined in XBD  
 108140       [Chapter 11](#) (on page 199).

108141       **parodd** (**-parodd**) Select odd (even) parity. This shall have the effect of setting (not setting)  
 108142       PARODD in the **termios** *c\_flag* field, as defined in XBD [Chapter 11](#) (on page  
 108143       199).  
 108144

108145       **cs5 cs6 cs7 cs8** Select character size, if possible. This shall have the effect of setting CS5, CS6,  
 108146       CS7, and CS8, respectively, in the **termios** *c\_flag* field, as defined in XBD  
 108147       [Chapter 11](#) (on page 199).

108148       **number**       Set terminal baud rate to the number given, if possible. If the baud rate is set  
 108149       to zero, the modem control lines shall no longer be asserted. This shall have  
 108150       the effect of setting the input and output **termios** baud rate values as defined  
 108151       in XBD [Chapter 11](#) (on page 199).



|        |                                  |                                                                                                              |
|--------|----------------------------------|--------------------------------------------------------------------------------------------------------------|
| 108152 | <b>ispeed</b> <i>number</i>      | Set terminal input baud rate to the number given, if possible. If the input baud                             |
| 108153 |                                  | rate is set to zero, the input baud rate shall be specified by the value of the                              |
| 108154 |                                  | output baud rate. This shall have the effect of setting the input <b>termios</b> baud                        |
| 108155 |                                  | rate values as defined in XBD <a href="#">Chapter 11</a> (on page 199).                                      |
| 108156 | <b>ospeed</b> <i>number</i>      | Set terminal output baud rate to the number given, if possible. If the output                                |
| 108157 |                                  | baud rate is set to zero, the modem control lines shall no longer be asserted.                               |
| 108158 |                                  | This shall have the effect of setting the output <b>termios</b> baud rate values as                          |
| 108159 |                                  | defined in XBD <a href="#">Chapter 11</a> (on page 199).                                                     |
| 108160 | <b>hupcl</b> ( <b>-hupcl</b> )   | Stop asserting modem control lines (do not stop asserting modem control                                      |
| 108161 |                                  | lines) on last close. This shall have the effect of setting (not setting) HUPCL in                           |
| 108162 |                                  | the <b>termios</b> <i>c_cflag</i> field, as defined in XBD <a href="#">Chapter 11</a> (on page 199).         |
| 108163 | <b>hup</b> ( <b>-hup</b> )       | Equivalent to <b>hupcl</b> ( <b>-hupcl</b> ).                                                                |
| 108164 | <b>cstopb</b> ( <b>-cstopb</b> ) | Use two (one) stop bits per character. This shall have the effect of setting (not                            |
| 108165 |                                  | setting) CSTOPB in the <b>termios</b> <i>c_cflag</i> field, as defined in XBD <a href="#">Chapter 11</a> (on |
| 108166 |                                  | page 199).                                                                                                   |
| 108167 | <b>cread</b> ( <b>-cread</b> )   | Enable (disable) the receiver. This shall have the effect of setting (not setting)                           |
| 108168 |                                  | CREAD in the <b>termios</b> <i>c_cflag</i> field, as defined in XBD <a href="#">Chapter 11</a> (on page      |
| 108169 |                                  | 199).                                                                                                        |
| 108170 | <b>clocal</b> ( <b>-clocal</b> ) | Assume a line without (with) modem control. This shall have the effect of                                    |
| 108171 |                                  | setting (not setting) CLOCAL in the <b>termios</b> <i>c_cflag</i> field, as defined in XBD                   |
| 108172 |                                  | <a href="#">Chapter 11</a> (on page 199).                                                                    |

It is unspecified whether *stty* shall report an error if an attempt to set a Control Mode fails.

## Input Modes

|        |                                  |                                                                                                              |
|--------|----------------------------------|--------------------------------------------------------------------------------------------------------------|
| 108174 |                                  |                                                                                                              |
| 108175 | <b>ignbrk</b> ( <b>-ignbrk</b> ) | Ignore (do not ignore) break on input. This shall have the effect of setting (not                            |
| 108176 |                                  | setting) IGNBRK in the <b>termios</b> <i>c_iflag</i> field, as defined in XBD <a href="#">Chapter 11</a> (on |
| 108177 |                                  | page 199).                                                                                                   |
| 108178 | <b>brkint</b> ( <b>-brkint</b> ) | Signal (do not signal) INTR on break. This shall have the effect of setting (not                             |
| 108179 |                                  | setting) BRKINT in the <b>termios</b> <i>c_iflag</i> field, as defined in XBD <a href="#">Chapter 11</a> (on |
| 108180 |                                  | page 199).                                                                                                   |
| 108181 | <b>ignpar</b> ( <b>-ignpar</b> ) | Ignore (do not ignore) bytes with parity errors. This shall have the effect of                               |
| 108182 |                                  | setting (not setting) IGNPAR in the <b>termios</b> <i>c_iflag</i> field, as defined in XBD                   |
| 108183 |                                  | <a href="#">Chapter 11</a> (on page 199).                                                                    |
| 108184 | <b>parmrk</b> ( <b>-parmrk</b> ) |                                                                                                              |
| 108185 |                                  | Mark (do not mark) parity errors. This shall have the effect of setting (not                                 |
| 108186 |                                  | setting) PARMRK in the <b>termios</b> <i>c_iflag</i> field, as defined in XBD <a href="#">Chapter 11</a> (on |
| 108187 |                                  | page 199).                                                                                                   |
| 108188 | <b>inpck</b> ( <b>-inpck</b> )   | Enable (disable) input parity checking. This shall have the effect of setting (not                           |
| 108189 |                                  | setting) INPCK in the <b>termios</b> <i>c_iflag</i> field, as defined in XBD <a href="#">Chapter 11</a> (on  |
| 108190 |                                  | page 199).                                                                                                   |
| 108191 | <b>istrip</b> ( <b>-istrip</b> ) | Strip (do not strip) input characters to seven bits. This shall have the effect of                           |
| 108192 |                                  | setting (not setting) ISTRIP in the <b>termios</b> <i>c_iflag</i> field, as defined in XBD                   |
| 108193 |                                  | <a href="#">Chapter 11</a> (on page 199).                                                                    |

|        |                                      |                                                                                                                                                                                                                                                                                                                          |
|--------|--------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 108194 | <b>inlcr</b> ( <b>-inlcr</b> )       | Map (do not map) NL to CR on input. This shall have the effect of setting (not setting) INLCR in the <b>termios</b> <i>c_iflag</i> field, as defined in XBD <a href="#">Chapter 11</a> (on page 199).                                                                                                                    |
| 108195 |                                      |                                                                                                                                                                                                                                                                                                                          |
| 108196 |                                      |                                                                                                                                                                                                                                                                                                                          |
| 108197 | <b>igncr</b> ( <b>-igncr</b> )       | Ignore (do not ignore) CR on input. This shall have the effect of setting (not setting) IGNCR in the <b>termios</b> <i>c_iflag</i> field, as defined in XBD <a href="#">Chapter 11</a> (on page 199).                                                                                                                    |
| 108198 |                                      |                                                                                                                                                                                                                                                                                                                          |
| 108199 |                                      |                                                                                                                                                                                                                                                                                                                          |
| 108200 | <b>icrnl</b> ( <b>-icrnl</b> )       | Map (do not map) CR to NL on input. This shall have the effect of setting (not setting) ICRNL in the <b>termios</b> <i>c_iflag</i> field, as defined in XBD <a href="#">Chapter 11</a> (on page 199).                                                                                                                    |
| 108201 |                                      |                                                                                                                                                                                                                                                                                                                          |
| 108202 |                                      |                                                                                                                                                                                                                                                                                                                          |
| 108203 | <b>ixon</b> ( <b>-ixon</b> )         | Enable (disable) START/STOP output control. Output from the system is stopped when the system receives STOP and started when the system receives START. This shall have the effect of setting (not setting) IXON in the <b>termios</b> <i>c_iflag</i> field, as defined in XBD <a href="#">Chapter 11</a> (on page 199). |
| 108204 |                                      |                                                                                                                                                                                                                                                                                                                          |
| 108205 |                                      |                                                                                                                                                                                                                                                                                                                          |
| 108206 |                                      |                                                                                                                                                                                                                                                                                                                          |
| 108207 | <b>ixany</b> ( <b>-ixany</b> )       | Allow any character to restart output. This shall have the effect of setting (not setting) IXANY in the <b>termios</b> <i>c_iflag</i> field, as defined in XBD <a href="#">Chapter 11</a> (on page 199).                                                                                                                 |
| 108208 |                                      |                                                                                                                                                                                                                                                                                                                          |
| 108209 |                                      |                                                                                                                                                                                                                                                                                                                          |
| 108210 | <b>ixoff</b> ( <b>-ixoff</b> )       | Request that the system send (not send) STOP characters when the input queue is nearly full and START characters to resume data transmission. This shall have the effect of setting (not setting) IXOFF in the <b>termios</b> <i>c_iflag</i> field, as defined in XBD <a href="#">Chapter 11</a> (on page 199).          |
| 108211 |                                      |                                                                                                                                                                                                                                                                                                                          |
| 108212 |                                      |                                                                                                                                                                                                                                                                                                                          |
| 108213 |                                      |                                                                                                                                                                                                                                                                                                                          |
| 108214 | <b>Output Modes</b>                  |                                                                                                                                                                                                                                                                                                                          |
| 108215 | <b>opost</b> ( <b>-opost</b> )       | Post-process output (do not post-process output; ignore all other output modes). This shall have the effect of setting (not setting) OPOST in the <b>termios</b> <i>c_oflag</i> field, as defined in XBD <a href="#">Chapter 11</a> (on page 199).                                                                       |
| 108216 |                                      |                                                                                                                                                                                                                                                                                                                          |
| 108217 |                                      |                                                                                                                                                                                                                                                                                                                          |
| 108218 | XSI <b>ocrnl</b> ( <b>-ocrnl</b> )   | Map (do not map) CR to NL on output. This shall have the effect of setting (not setting) OCRNL in the <b>termios</b> <i>c_oflag</i> field, as defined in XBD <a href="#">Chapter 11</a> (on page 199).                                                                                                                   |
| 108219 |                                      |                                                                                                                                                                                                                                                                                                                          |
| 108220 |                                      |                                                                                                                                                                                                                                                                                                                          |
| 108221 | XSI <b>onocr</b> ( <b>-onocr</b> )   | Do not (do) output CR at column zero. This shall have the effect of setting (not setting) ONOCR in the <b>termios</b> <i>c_oflag</i> field, as defined in XBD <a href="#">Chapter 11</a> (on page 199).                                                                                                                  |
| 108222 |                                      |                                                                                                                                                                                                                                                                                                                          |
| 108223 |                                      |                                                                                                                                                                                                                                                                                                                          |
| 108224 | XSI <b>onlret</b> ( <b>-onlret</b> ) | The terminal newline key performs (does not perform) the CR function. This shall have the effect of setting (not setting) ONLRET in the <b>termios</b> <i>c_oflag</i> field, as defined in XBD <a href="#">Chapter 11</a> (on page 199).                                                                                 |
| 108225 |                                      |                                                                                                                                                                                                                                                                                                                          |
| 108226 |                                      |                                                                                                                                                                                                                                                                                                                          |
| 108227 | XSI <b>ofill</b> ( <b>-ofill</b> )   | Use fill characters (use timing) for delays. This shall have the effect of setting (not setting) OFILL in the <b>termios</b> <i>c_oflag</i> field, as defined in XBD <a href="#">Chapter 11</a> (on page 199).                                                                                                           |
| 108228 |                                      |                                                                                                                                                                                                                                                                                                                          |
| 108229 |                                      |                                                                                                                                                                                                                                                                                                                          |
| 108230 | XSI <b>ofdel</b> ( <b>-ofdel</b> )   | Fill characters are DELs (NULs). This shall have the effect of setting (not setting) OFDEL in the <b>termios</b> <i>c_oflag</i> field, as defined in XBD <a href="#">Chapter 11</a> (on page 199).                                                                                                                       |
| 108231 |                                      |                                                                                                                                                                                                                                                                                                                          |
| 108232 |                                      |                                                                                                                                                                                                                                                                                                                          |
| 108233 | XSI <b>cr0 cr1 cr2 cr3</b>           | Select the style of delay for CRs. This shall have the effect of setting CRDLY to CR0, CR1, CR2, or CR3, respectively, in the <b>termios</b> <i>c_oflag</i> field, as defined in XBD <a href="#">Chapter 11</a> (on page 199).                                                                                           |
| 108234 |                                      |                                                                                                                                                                                                                                                                                                                          |
| 108235 |                                      |                                                                                                                                                                                                                                                                                                                          |

|        |     |                            |                                                                                                                                                                                                                                                                                                                                     |
|--------|-----|----------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 108236 | XSI | <b>nl0 nl1</b>             | Select the style of delay for NL. This shall have the effect of setting NLDLY to NL0 or NL1, respectively, in the <b>termios</b> <i>c_oflag</i> field, as defined in XBD <a href="#">Chapter 11</a> (on page 199).                                                                                                                  |
| 108237 |     |                            |                                                                                                                                                                                                                                                                                                                                     |
| 108238 |     |                            |                                                                                                                                                                                                                                                                                                                                     |
| 108239 | XSI | <b>tab0 tab1 tab2 tab3</b> | Select the style of delay for horizontal tabs. This shall have the effect of setting TABDLY to TAB0, TAB1, TAB2, or TAB3, respectively, in the <b>termios</b> <i>c_oflag</i> field, as defined in XBD <a href="#">Chapter 11</a> (on page 199). Note that TAB3 has the effect of expanding <tab> characters to <space> characters.  |
| 108240 |     |                            |                                                                                                                                                                                                                                                                                                                                     |
| 108241 |     |                            |                                                                                                                                                                                                                                                                                                                                     |
| 108242 |     |                            |                                                                                                                                                                                                                                                                                                                                     |
| 108243 |     |                            |                                                                                                                                                                                                                                                                                                                                     |
| 108244 | XSI | <b>tabs (-tabs)</b>        | Synonym for <b>tab0 (tab3)</b> .                                                                                                                                                                                                                                                                                                    |
| 108245 | XSI | <b>bs0 bs1</b>             | Select the style of delay for <backspace> characters. This shall have the effect of setting BSDLY to BS0 or BS1, respectively, in the <b>termios</b> <i>c_oflag</i> field, as defined in XBD <a href="#">Chapter 11</a> (on page 199).                                                                                              |
| 108246 |     |                            |                                                                                                                                                                                                                                                                                                                                     |
| 108247 |     |                            |                                                                                                                                                                                                                                                                                                                                     |
| 108248 | XSI | <b>ff0 ff1</b>             | Select the style of delay for <form-feed> characters. This shall have the effect of setting FFDLY to FF0 or FF1, respectively, in the <b>termios</b> <i>c_oflag</i> field, as defined in XBD <a href="#">Chapter 11</a> (on page 199).                                                                                              |
| 108249 |     |                            |                                                                                                                                                                                                                                                                                                                                     |
| 108250 |     |                            |                                                                                                                                                                                                                                                                                                                                     |
| 108251 | XSI | <b>vt0 vt1</b>             | Select the style of delay for <vertical-tab> characters. This shall have the effect of setting VTDLY to VT0 or VT1, respectively, in the <b>termios</b> <i>c_oflag</i> field, as defined in XBD <a href="#">Chapter 11</a> (on page 199).                                                                                           |
| 108252 |     |                            |                                                                                                                                                                                                                                                                                                                                     |
| 108253 |     |                            |                                                                                                                                                                                                                                                                                                                                     |
| 108254 |     | <b>Local Modes</b>         |                                                                                                                                                                                                                                                                                                                                     |
| 108255 |     | <b>isig (-isig)</b>        | Enable (disable) the checking of characters against the special control characters INTR, QUIT, and SUSP. This shall have the effect of setting (not setting) ISIG in the <b>termios</b> <i>c_lflag</i> field, as defined in XBD <a href="#">Chapter 11</a> (on page 199).                                                           |
| 108256 |     |                            |                                                                                                                                                                                                                                                                                                                                     |
| 108257 |     |                            |                                                                                                                                                                                                                                                                                                                                     |
| 108258 |     |                            |                                                                                                                                                                                                                                                                                                                                     |
| 108259 |     | <b>icanon (-icanon)</b>    | Enable (disable) canonical input (ERASE and KILL processing). This shall have the effect of setting (not setting) ICANON in the <b>termios</b> <i>c_lflag</i> field, as defined in XBD <a href="#">Chapter 11</a> (on page 199).                                                                                                    |
| 108260 |     |                            |                                                                                                                                                                                                                                                                                                                                     |
| 108261 |     |                            |                                                                                                                                                                                                                                                                                                                                     |
| 108262 |     | <b>ixexten (-ixexten)</b>  | Enable (disable) any implementation-defined special control characters not currently controlled by <b>icanon</b> , <b>isig</b> , <b>ixon</b> , or <b>ixoff</b> . This shall have the effect of setting (not setting) IEXTEN in the <b>termios</b> <i>c_lflag</i> field, as defined in XBD <a href="#">Chapter 11</a> (on page 199). |
| 108263 |     |                            |                                                                                                                                                                                                                                                                                                                                     |
| 108264 |     |                            |                                                                                                                                                                                                                                                                                                                                     |
| 108265 |     |                            |                                                                                                                                                                                                                                                                                                                                     |
| 108266 |     | <b>echo (-echo)</b>        | Echo back (do not echo back) every character typed. This shall have the effect of setting (not setting) ECHO in the <b>termios</b> <i>c_lflag</i> field, as defined in XBD <a href="#">Chapter 11</a> (on page 199).                                                                                                                |
| 108267 |     |                            |                                                                                                                                                                                                                                                                                                                                     |
| 108268 |     |                            |                                                                                                                                                                                                                                                                                                                                     |
| 108269 |     | <b>echoe (-echoe)</b>      | The ERASE character visually erases (does not erase) the last character in the current line from the display, if possible. This shall have the effect of setting (not setting) ECHOE in the <b>termios</b> <i>c_lflag</i> field, as defined in XBD <a href="#">Chapter 11</a> (on page 199).                                        |
| 108270 |     |                            |                                                                                                                                                                                                                                                                                                                                     |
| 108271 |     |                            |                                                                                                                                                                                                                                                                                                                                     |
| 108272 |     |                            |                                                                                                                                                                                                                                                                                                                                     |
| 108273 |     | <b>echok (-echok)</b>      | Echo (do not echo) NL after KILL character. This shall have the effect of setting (not setting) ECHOK in the <b>termios</b> <i>c_lflag</i> field, as defined in XBD <a href="#">Chapter 11</a> (on page 199).                                                                                                                       |
| 108274 |     |                            |                                                                                                                                                                                                                                                                                                                                     |
| 108275 |     |                            |                                                                                                                                                                                                                                                                                                                                     |
| 108276 |     | <b>echonl (-echonl)</b>    | Echo (do not echo) NL, even if <b>echo</b> is disabled. This shall have the effect of setting (not setting) ECHONL in the <b>termios</b> <i>c_lflag</i> field, as defined in XBD <a href="#">Chapter 11</a> (on page 199).                                                                                                          |
| 108277 |     |                            |                                                                                                                                                                                                                                                                                                                                     |
| 108278 |     |                            |                                                                                                                                                                                                                                                                                                                                     |

**noflsh** (**-noflsh**) Disable (enable) flush after INTR, QUIT, SUSP. This shall have the effect of setting (not setting) NOFLSH in the **termios** *c\_lflag* field, as defined in XBD [Chapter 11](#) (on page 199).

**tostop** (**-tostop**) Send SIGTTOU for background output. This shall have the effect of setting (not setting) TOSTOP in the **termios** *c\_lflag* field, as defined in XBD [Chapter 11](#) (on page 199).

### Special Control Character Assignments

*<control>-character string*

Set *<control>-character* to *string*. If *<control>-character* is one of the character sequences in the first column of the following table, the corresponding XBD [Chapter 11](#) (on page 199) control character from the second column shall be recognized. This has the effect of setting the corresponding element of the **termios** *c\_cc* array (see XBD [Chapter 13](#) (on page 219), **<termios.h>**).

**Table 4-20** Control Character Names in *stty*

| Control Character | c_cc Subscript | Description     |
|-------------------|----------------|-----------------|
| <b>eof</b>        | VEOF           | EOF character   |
| <b>eol</b>        | VEOL           | EOL character   |
| <b>erase</b>      | VERASE         | ERASE character |
| <b>intr</b>       | VINTR          | INTR character  |
| <b>kill</b>       | VKILL          | KILL character  |
| <b>quit</b>       | VQUIT          | QUIT character  |
| <b>susp</b>       | VSUSP          | SUSP character  |
| <b>start</b>      | VSTART         | START character |
| <b>stop</b>       | VSTOP          | STOP character  |

If *string* is a single character, the control character shall be set to that character. If *string* is the two-character sequence "**^**-" or the string *undef*, the control character shall be set to `_POSIX_VDISABLE`, if it is in effect for the device; if `_POSIX_VDISABLE` is not in effect for the device, it shall be treated as an error. In the POSIX locale, if *string* is a two-character sequence beginning with **<circumflex>** ('**^**'), and the second character is one of those listed in the "**^c**" column of the following table, the control character shall be set to the corresponding character value in the Value column of the table.

**Table 4-21** Circumflex Control Characters in *stty*

| <b>^c</b> | Value              | <b>^c</b> | Value              | <b>^c</b> | Value              |
|-----------|--------------------|-----------|--------------------|-----------|--------------------|
| a, A      | <b>&lt;SOH&gt;</b> | l, L      | <b>&lt;FF&gt;</b>  | w, W      | <b>&lt;ETB&gt;</b> |
| b, B      | <b>&lt;STX&gt;</b> | m, M      | <b>&lt;CR&gt;</b>  | x, X      | <b>&lt;CAN&gt;</b> |
| c, C      | <b>&lt;ETX&gt;</b> | n, N      | <b>&lt;SO&gt;</b>  | y, Y      | <b>&lt;EM&gt;</b>  |
| d, D      | <b>&lt;EOT&gt;</b> | o, O      | <b>&lt;SI&gt;</b>  | z, Z      | <b>&lt;SUB&gt;</b> |
| e, E      | <b>&lt;ENQ&gt;</b> | p, P      | <b>&lt;DLE&gt;</b> | [         | <b>&lt;ESC&gt;</b> |
| f, F      | <b>&lt;ACK&gt;</b> | q, Q      | <b>&lt;DC1&gt;</b> | \         | <b>&lt;FS&gt;</b>  |
| g, G      | <b>&lt;BEL&gt;</b> | r, R      | <b>&lt;DC2&gt;</b> | ]         | <b>&lt;GS&gt;</b>  |
| h, H      | <b>&lt;BS&gt;</b>  | s, S      | <b>&lt;DC3&gt;</b> | ^         | <b>&lt;RS&gt;</b>  |
| i, I      | <b>&lt;HT&gt;</b>  | t, T      | <b>&lt;DC4&gt;</b> | _         | <b>&lt;US&gt;</b>  |
| j, J      | <b>&lt;LF&gt;</b>  | u, U      | <b>&lt;NAK&gt;</b> | ?         | <b>&lt;DEL&gt;</b> |
| k, K      | <b>&lt;VT&gt;</b>  | v, V      | <b>&lt;SYN&gt;</b> |           |                    |

108323 **min** *number*  
 108324 Set the value of MIN to *number*. MIN is used in non-canonical mode input processing  
 108325 (**icanon**).

108326 **time** *number*  
 108327 Set the value of TIME to *number*. TIME is used in non-canonical mode input processing  
 108328 (**icanon**).

108329 **Combination Modes**

108330 *saved settings*  
 108331 Set the current terminal characteristics to the saved settings produced by the **-g** option.

108332 **evenp** or **parity**  
 108333 Enable **parenb** and **cs7**; disable **parodd**.

108334 **oddp**  
 108335 Enable **parenb**, **cs7**, and **parodd**.

108336 **-parity**, **-evenp**, or **-oddp**  
 108337 Disable **parenb**, and set **cs8**.

108338 XSI **raw** (**-raw** or **cooked**)  
 108339 Enable (disable) raw input and output. Raw mode shall be equivalent to setting:  
 108340 `stty cs8 erase ^- kill ^- intr ^- \  
 108341 quit ^- eof ^- eol ^- -post -inpck`

108342 **nl** (**-nl**)  
 108343 Disable (enable) **icrnl**. In addition, **-nl** unsets **inlcr** and **igncr**.

108344 **ek** Reset ERASE and KILL characters back to system defaults.

108345 **sane**  
 108346 Reset all modes to some reasonable, unspecified, values.

108347 **STDIN**  
 108348 Although no input is read from standard input, standard input shall be used to get the current  
 108349 terminal I/O characteristics and to set new terminal I/O characteristics.

108350 **INPUT FILES**  
 108351 None.

108352 **ENVIRONMENT VARIABLES**  
 108353 The following environment variables shall affect the execution of *stty*:

108354 **LANG** Provide a default value for the internationalization variables that are unset or null.  
 108355 (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 108356 variables used to determine the values of locale categories.)

108357 **LC\_ALL** If set to a non-empty string value, override the values of all the other  
 108358 internationalization variables.

108359 **LC\_CTYPE** This variable determines the locale for the interpretation of sequences of bytes of  
 108360 text data as characters (for example, single-byte as opposed to multi-byte  
 108361 characters in arguments) and which characters are in the class **print**.

108362 **LC\_MESSAGES**  
 108363 Determine the locale that should be used to affect the format and contents of  
 108364 diagnostic messages written to standard error.

108365 XSI **NLSPATH** Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

108366 **ASYNCHRONOUS EVENTS**  
 108367 Default.

108368 **STDOUT**  
 108369 If operands are specified, no output shall be produced.

108370 If the **-g** option is specified, *stty* shall write to standard output the current settings in a form that  
 108371 can be used as arguments to another instance of *stty* on the same system.

108372 If the **-a** option is specified, all of the information as described in the OPERANDS section shall  
 108373 be written to standard output. Unless otherwise specified, this information shall be written as  
 108374 <space>-separated tokens in an unspecified format, on one or more lines, with an unspecified  
 108375 number of tokens per line. Additional information may be written.

108376 If no options or operands are specified, an unspecified subset of the information written for the  
 108377 **-a** option shall be written.

108378 If speed information is written as part of the default output, or if the **-a** option is specified and if  
 108379 the terminal input speed and output speed are the same, the speed information shall be written  
 108380 as follows:

108381 "speed %d baud;", <speed>  
 108382 Otherwise, speeds shall be written as:

108383 "ispeed %d baud; ospeed %d baud;", <ispeed>, <ospeed>

108384 In locales other than the POSIX locale, the word **baud** may be changed to something more  
 108385 appropriate in those locales.

108386 If control characters are written as part of the default output, or if the **-a** option is specified,  
 108387 control characters shall be written as:

108388 "%s = %s;", <control-character name>, <value>  
 108389 where <value> is either the character, or some visual representation of the character if it is non-  
 108390 printable, or the string *undef* if the character is disabled.

108391 **STDERR**  
 108392 The standard error shall be used only for diagnostic messages.

108393 **OUTPUT FILES**  
 108394 None.

108395 **EXTENDED DESCRIPTION**  
 108396 None.

108397 **EXIT STATUS**  
 108398 The following exit values shall be returned:

108399 0 The terminal options were read or set successfully.

108400 >0 An error occurred.

108401 **CONSEQUENCES OF ERRORS**

108402 Default.

108403 **APPLICATION USAGE**

108404 The **-g** flag is designed to facilitate the saving and restoring of terminal state from the shell level.  
 108405 For example, a program may:

```
108406 saveterm="$(stty -g)" # save terminal state
108407 stty (new settings) # set new state
108408 ... # ...
108409 stty $saveterm # restore terminal state
```

108410 Since the format is unspecified, the saved value is not portable across systems.

108411 Since the **-a** format is so loosely specified, scripts that save and restore terminal settings should  
 108412 use the **-g** option.

108413 **EXAMPLES**

108414 None.

108415 **RATIONALE**

108416 The original *stty* description was taken directly from System V and reflected the System V  
 108417 terminal driver **termio**. It has been modified to correspond to the terminal driver **termios**.

108418 Output modes are specified only for XSI-conformant systems. All implementations are expected  
 108419 to provide *stty* operands corresponding to all of the output modes they support.

108420 The *stty* utility is primarily used to tailor the user interface of the terminal, such as selecting the  
 108421 preferred ERASE and KILL characters. As an application programming utility, *stty* can be used  
 108422 within shell scripts to alter the terminal settings for the duration of the script.

108423 The **termios** section states that individual disabling of control characters is possible through the  
 108424 option `_POSIX_VDISABLE`. If enabled, two conventions currently exist for specifying this:  
 108425 System V uses `"^_"`, and BSD uses *undef*. Both are accepted by *stty* in this volume of  
 108426 POSIX.1-2008. The other BSD convention of using the letter `'u'` was rejected because it conflicts  
 108427 with the actual letter `'u'`, which is an acceptable value for a control character.

108428 Early proposals did not specify the mapping of `"^c"` to control characters because the control  
 108429 characters were not specified in the POSIX locale character set description file requirements. The  
 108430 control character set is now specified in XBD [Chapter 3](#) (on page 33), so the historical mapping is  
 108431 specified. Note that although the mapping corresponds to control-character key assignments on  
 108432 many terminals that use the ISO/IEC 646:1991 standard (or ASCII) character encodings, the  
 108433 mapping specified here is to the control characters, not their keyboard encodings.

108434 Since **termios** supports separate speeds for input and output, two new options were added to  
 108435 specify each distinctly.

108436 Some historical implementations use standard input to get and set terminal characteristics;  
 108437 others use standard output. Since input from a login TTY is usually restricted to the owner while  
 108438 output to a TTY is frequently open to anyone, using standard input provides fewer chances of  
 108439 accidentally (or maliciously) altering the terminal settings of other users. Using standard input  
 108440 also allows *stty -a* and *stty -g* output to be redirected for later use. Therefore, usage of standard  
 108441 input is required by this volume of POSIX.1-2008.

108442 **FUTURE DIRECTIONS**

108443 None.



108444 **SEE ALSO**108445 [Chapter 2](#) (on page 2319)108446 XBD [Chapter 8](#) (on page 173), [Chapter 11](#) (on page 199), [Section 12.2](#) (on page 215), [<termios.h>](#)108447 **CHANGE HISTORY**

108448 First released in Issue 2.

108449 **Issue 5**108450 The description of **tabs** is clarified.

108451 The FUTURE DIRECTIONS section is added.

108452 **Issue 6**108453 The LEGACY items **iuc lc(-iuc lc)**, **xcase**, **olc uc(-olc uc)**, **lcase(-lcase)**, and **LCASE(-LCASE)** are removed.108455 IEEE Std 1003.1-2001/Cor 1-2002, item XCU/TC1/D6/37 is applied, applying IEEE PASC Interpretation 1003.2 #133, fixing an error in the OPERANDS section for the Combination Modes **nl(-nl)**.108458 **Issue 7**

108459 Austin Group Interpretation 1003.1-2001 #144 is applied, moving functionality relating to the IXANY symbol from the XSI option to the Base.

108461 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.



108462 **NAME**108463        **tabs** — set terminal tabs108464 **SYNOPSIS**108465 XSI        **tabs** [*-n* | *-a* | *-a2* | *-c* | *-c2* | *-c3* | *-f* | *-p* | *-s* | *-u*] [*-T type*]108466        **tabs** [*-T type*] *n*[[*sep*[+]*n*]]...108467 **DESCRIPTION**

108468        The *tabs* utility shall display a series of characters that first clears the hardware terminal tab  
 108469 XSI       settings and then initializes the tab stops at the specified positions and optionally adjusts the  
 108470        margin.

108471        The phrase “tab-stop position *N*” shall be taken to mean that, from the start of a line of output,  
 108472        tabbing to position *N* shall cause the next character output to be in the (*N*+1)th column position  
 108473        on that line. The maximum number of tab stops allowed is terminal-dependent.

108474        It need not be possible to implement *tabs* on certain terminals. If the terminal type obtained from  
 108475        the *TERM* environment variable or *-T* option represents such a terminal, an appropriate  
 108476        diagnostic message shall be written to standard error and *tabs* shall exit with a status greater  
 108477        than zero.

108478 **OPTIONS**

108479 XSI        The *tabs* utility shall conform to XBD [Section 12.2](#) (on page 215), except for various extensions:  
 108480        the options *-a2*, *-c2*, and *-c3* are multi-character.

108481        The following options shall be supported:

108482        *-n*        Specify repetitive tab stops separated by a uniform number of column positions, *n*,  
 108483               where *n* is a single-digit decimal number. The default usage of *tabs* with no  
 108484               arguments shall be equivalent to *tabs -8*. When *-0* is used, the tab stops shall be  
 108485               cleared and no new ones set.

108486 XSI        *-a*        1,10,16,36,72  
 108487               Assembler, applicable to some mainframes.

108488 XSI        *-a2*       1,10,16,40,72  
 108489               Assembler, applicable to some mainframes.

108490 XSI        *-c*        1,8,12,16,20,55  
 108491               COBOL, normal format.

108492 XSI        *-c2*       1,6,10,14,49  
 108493               COBOL, compact format (columns 1 to 6 omitted).

108494 XSI        *-c3*       1,6,10,14,18,22,26,30,34,38,42,46,50,54,58,62,67  
 108495               COBOL compact format (columns 1 to 6 omitted), with more tabs than *-c2*.

108496 XSI        *-f*        1,7,11,15,19,23  
 108497               FORTRAN

108498 XSI        *-p*        1,5,9,13,17,21,25,29,33,37,41,45,49,53,57,61  
 108499               PL/1

108500 XSI        *-s*        1,10,55  
 108501               SNOBOL

108502 XSI        *-u*        1,12,20,44  
 108503               Assembler, applicable to some mainframes.

108504        -T *type*        Indicate the type of terminal. If this option is not supplied and the *TERM* variable  
108505                               is unset or null, an unspecified default terminal type shall be used. The setting of  
108506                               *type* shall take precedence over the value in *TERM*.

## 108507 OPERANDS

108508        The following operand shall be supported:

108509        *n*[[*sep*[+]*n*]...] A single command line argument that consists of one or more tab-stop values (*n*)  
108510                               separated by a separator character (*sep*) which is either a <comma> or a <blank>  
108511                               character. The application shall ensure that the tab-stop values are positive decimal  
108512                               integers in strictly ascending order. If any tab-stop value (except the first one) is  
108513                               preceded by a <plus-sign>, it is taken as an increment to be added to the previous  
108514                               value. For example, the tab lists 1,10,20,30 and "1 10 +10 +10" are considered  
108515                               to be identical.

## 108516 STDIN

108517        Not used.

## 108518 INPUT FILES

108519        None.

## 108520 ENVIRONMENT VARIABLES

108521        The following environment variables shall affect the execution of *tabs*:

108522        *LANG*            Provide a default value for the internationalization variables that are unset or null.  
108523                               (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
108524                               variables used to determine the values of locale categories.)

108525        *LC\_ALL*          If set to a non-empty string value, override the values of all the other  
108526                               internationalization variables.

108527        *LC\_CTYPE*        Determine the locale for the interpretation of sequences of bytes of text data as  
108528                               characters (for example, single-byte as opposed to multi-byte characters in  
108529                               arguments).

108530        *LC\_MESSAGES*

108531                               Determine the locale that should be used to affect the format and contents of  
108532                               diagnostic messages written to standard error.

108533 XSI        *NLSPATH*    Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

108534        *TERM*            Determine the terminal type. If this variable is unset or null, and if the -T option is  
108535                               not specified, an unspecified default terminal type shall be used.

## 108536 ASYNCHRONOUS EVENTS

108537        Default.

## 108538 STDOUT

108539        If standard output is a terminal, the appropriate sequence to clear and set the tab stops may be  
108540        written to standard output in an unspecified format. If standard output is not a terminal,  
108541        undefined results occur.

## 108542 STDERR

108543        The standard error shall be used only for diagnostic messages.

## 108544 OUTPUT FILES

108545        None.

108546 **EXTENDED DESCRIPTION**

108547 None.

108548 **EXIT STATUS**

108549 The following exit values shall be returned:

108550 0 Successful completion.

108551 &gt;0 An error occurred.

108552 **CONSEQUENCES OF ERRORS**

108553 Default.

108554 **APPLICATION USAGE**108555 This utility makes use of the terminal's hardware tabs and the *stty tabs* option.

108556 This utility is not recommended for application use.

108557 Some integrated display units might not have escape sequences to set tab stops, but may be set  
108558 by internal system calls. On these terminals, *tabs* works if standard output is directed to the  
108559 terminal; if output is directed to another file, however, *tabs* fails.

108560 **EXAMPLES**

108561 None.

108562 **RATIONALE**

108563 Consideration was given to having the *tput* utility handle all of the functions described in *tabs*.  
108564 However, the separate *tabs* utility was retained because it seems more intuitive to use a  
108565 command named *tabs* than *tput* with a new option. The *tput* utility does not support setting or  
108566 clearing tabs, and no known historical version of *tabs* supports the capability of setting arbitrary  
108567 tab stops.

108568 The System V *tabs* interface is very complex; the version in this volume of POSIX.1-2008 has a  
108569 reduced feature list, but many of the features omitted were restored as part of the XSI option  
108570 even though the supported languages and coding styles are primarily historical.

108571 There was considerable sentiment for specifying only a means of resetting the tabs back to a  
108572 known state—presumably the “standard” of tabs every eight positions. The following features  
108573 were omitted:

- 108574 • Setting tab stops via the first line in a file, using *--file*. Since even the SVID has no  
108575 complete explanation of this feature, it is doubtful that it is in widespread use.

108576 In an early proposal, a *-t tablist* option was added for consistency with *expand*; this was later  
108577 removed when inconsistencies with the historical list of tabs were identified.

108578 Consideration was given to adding a *-p* option that would output the current tab settings so  
108579 that they could be saved and then later restored. This was not accepted because querying the tab  
108580 stops of the terminal is not a capability in historical *terminfo* or *termcap* facilities and might not be  
108581 supported on a wide range of terminals.

108582 **FUTURE DIRECTIONS**

108583 None.

108584 **SEE ALSO**108585 *expand*, *stty*, *tput*, *unexpand*

108586 XBD Chapter 8 (on page 173), Section 12.2 (on page 215)

108587 **CHANGE HISTORY**

108588 First released in Issue 2.

108589 **Issue 6**

108590 This utility is marked as part of the User Portability Utilities option.

108591 The normative text is reworded to avoid use of the term “must” for application requirements.

108592 **Issue 7**

108593 The *tabs* utility is removed from the User Portability Utilities option. User Portability Utilities is  
108594 now an option for interactive utilities.

108595 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

108596 The SYNOPSIS and OPERANDS sections are updated.

108597 **NAME**

108598       tail — copy the last part of a file

108599 **SYNOPSIS**108600       tail [-f] [-c *number*|-n *number*] [*file*]108601 **DESCRIPTION**108602       The *tail* utility shall copy its input file to the standard output beginning at a designated place.

108603       Copying shall begin at the point in the file indicated by the *-c number* or *-n number* options. The  
 108604       option-argument *number* shall be counted in units of lines or bytes, according to the options *-n*  
 108605       and *-c*. Both line and byte counts start from 1.

108606       Tails relative to the end of the file may be saved in an internal buffer, and thus may be limited in  
 108607       length. Such a buffer, if any, shall be no smaller than {LINE\_MAX}\*10 bytes.

108608 **OPTIONS**

108609       The *tail* utility shall conform to XBD [Section 12.2](#) (on page 215), except that '+' may be  
 108610       recognized as an option delimiter as well as '-'.

108611       The following options shall be supported:

108612       *-c number*   The application shall ensure that the *number* option-argument is a decimal integer,  
 108613       optionally including a sign. The sign shall affect the location in the file, measured  
 108614       in bytes, to begin the copying:

| Sign        | Copying Starts                         |
|-------------|----------------------------------------|
| +           | Relative to the beginning of the file. |
| -           | Relative to the end of the file.       |
| <i>none</i> | Relative to the end of the file.       |

108619       The application shall ensure that if the sign of the *number* option-argument is '+',  
 108620       the *number* option-argument is a non-zero decimal integer.

108621       The origin for counting shall be 1; that is, *-c +1* represents the first byte of the file,  
 108622       *-c -1* the last.

108623       *-f*           If the input file is a regular file or if the *file* operand specifies a FIFO, do not  
 108624       terminate after the last line of the input file has been copied, but read and copy  
 108625       further bytes from the input file when they become available. If no *file* operand is  
 108626       specified and standard input is a pipe or FIFO, the *-f* option shall be ignored. If the  
 108627       input file is not a FIFO, pipe, or regular file, it is unspecified whether or not the *-f*  
 108628       option shall be ignored.

108629       *-n number*   This option shall be equivalent to *-c number*, except the starting location in the file  
 108630       shall be measured in lines instead of bytes. The origin for counting shall be 1; that  
 108631       is, *-n +1* represents the first line of the file, *-n -1* the last.

108632       If neither *-c* nor *-n* is specified, *-n 10* shall be assumed.108633 **OPERANDS**

108634       The following operand shall be supported:

108635       *file*        A pathname of an input file. If no *file* operand is specified, the standard input shall  
 108636       be used.

108637 **STDIN**

108638 The standard input shall be used if no *file* operand is specified, and shall be used if the *file*  
 108639 operand is '-' and the implementation treats the '-' as meaning standard input. Otherwise,  
 108640 the standard input shall not be used. See the INPUT FILES section.

108641 **INPUT FILES**

108642 If the -c option is specified, the input file can contain arbitrary data; otherwise, the input file  
 108643 shall be a text file.

108644 **ENVIRONMENT VARIABLES**

108645 The following environment variables shall affect the execution of *tail*:

108646 *LANG* Provide a default value for the internationalization variables that are unset or null.  
 108647 (See XBD Section 8.2 (on page 174) for the precedence of internationalization  
 108648 variables used to determine the values of locale categories.)

108649 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
 108650 internationalization variables.

108651 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
 108652 characters (for example, single-byte as opposed to multi-byte characters in  
 108653 arguments and input files).

108654 *LC\_MESSAGES*

108655 Determine the locale that should be used to affect the format and contents of  
 108656 diagnostic messages written to standard error.

108657 XSI *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

108658 **ASYNCHRONOUS EVENTS**

108659 Default.

108660 **STDOUT**

108661 The designated portion of the input file shall be written to standard output.

108662 **STDERR**

108663 The standard error shall be used only for diagnostic messages.

108664 **OUTPUT FILES**

108665 None.

108666 **EXTENDED DESCRIPTION**

108667 None.

108668 **EXIT STATUS**

108669 The following exit values shall be returned:

108670 0 Successful completion.

108671 >0 An error occurred.

108672 **CONSEQUENCES OF ERRORS**

108673 Default.

## 108674 APPLICATION USAGE

108675 The **-c** option should be used with caution when the input is a text file containing multi-byte  
 108676 characters; it may produce output that does not start on a character boundary.

108677 Although the input file to *tail* can be any type, the results might not be what would be expected  
 108678 on some character special device files or on file types not described by the System Interfaces  
 108679 volume of POSIX.1-2008. Since this volume of POSIX.1-2008 does not specify the block size used  
 108680 when doing input, *tail* need not read all of the data from devices that only perform block  
 108681 transfers.

## 108682 EXAMPLES

108683 The **-f** option can be used to monitor the growth of a file that is being written by some other  
 108684 process. For example, the command:

108685 `tail -f fred`

108686 prints the last ten lines of the file **fred**, followed by any lines that are appended to **fred** between  
 108687 the time *tail* is initiated and killed. As another example, the command:

108688 `tail -f -c 15 fred`

108689 prints the last 15 bytes of the file **fred**, followed by any bytes that are appended to **fred** between  
 108690 the time *tail* is initiated and killed.

## 108691 RATIONALE

108692 This version of *tail* was created to allow conformance to the Utility Syntax Guidelines. The  
 108693 historical **-b** option was omitted because of the general non-portability of block-sized units of  
 108694 text. The **-c** option historically meant “characters”, but this volume of POSIX.1-2008 indicates  
 108695 that it means “bytes”. This was selected to allow reasonable implementations when multi-byte  
 108696 characters are possible; it was not named **-b** to avoid confusion with the historical **-b**.

108697 The origin of counting both lines and bytes is 1, matching all widespread historical  
 108698 implementations. Hence *tail -n +0* is not conforming usage because it attempts to output line  
 108699 zero; but note that *tail -n 0* does conform, and outputs nothing.

108700 Earlier versions of this standard allowed the following forms in the SYNOPSIS:

108701 `tail -[number] [b|c|l] [f] [file]`

108702 `tail +[number] [b|c|l] [f] [file]`

108703 These forms are no longer specified by POSIX.1-2008, but may be present in some  
 108704 implementations.

108705 The restriction on the internal buffer is a compromise between the historical System V  
 108706 implementation of 4 096 bytes and the BSD 32 768 bytes.

108707 The **-f** option has been implemented as a loop that sleeps for 1 second and copies any bytes that  
 108708 are available. This is sufficient, but if more efficient methods of determining when new data are  
 108709 available are developed, implementations are encouraged to use them.

108710 Historical documentation indicates that *tail* ignores the **-f** option if the input file is a pipe (pipe  
 108711 and FIFO on systems that support FIFOs). On BSD-based systems, this has been true; on System  
 108712 V-based systems, this was true when input was taken from standard input, but it did not ignore  
 108713 the **-f** flag if a FIFO was named as the *file* operand. Since the **-f** option is not useful on pipes and  
 108714 all historical implementations ignore **-f** if no *file* operand is specified and standard input is a  
 108715 pipe, this volume of POSIX.1-2008 requires this behavior. However, since the **-f** option is useful  
 108716 on a FIFO, this volume of POSIX.1-2008 also requires that if a FIFO is named, the **-f** option shall  
 108717 not be ignored. Earlier versions of this standard did not state any requirement for the case where  
 108718 no *file* operand is specified and standard input is a FIFO. The standard has been updated to

108719 reflect current practice which is to treat this case the same as a pipe on standard input. Although  
108720 historical behavior does not ignore the `-f` option for other file types, this is unspecified so that  
108721 implementations are allowed to ignore the `-f` option if it is known that the file cannot be  
108722 extended.

#### 108723 **FUTURE DIRECTIONS**

108724 None.

#### 108725 **SEE ALSO**

108726 *head*

108727 XBD [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 215)

#### 108728 **CHANGE HISTORY**

108729 First released in Issue 2.

#### 108730 **Issue 6**

108731 The obsolescent SYNOPSIS lines and associated text are removed.

108732 The normative text is reworded to avoid use of the term “must” for application requirements.

#### 108733 **Issue 7**

108734 Austin Group Interpretation 1003.1-2001 #027 is applied, clarifying that ‘+’ may be recognized  
108735 as an option delimiter in the OPTIONS section.

108736 Austin Group Interpretation 1003.1-2001 #092 is applied.

108737 Austin Group Interpretation 1003.1-2001 #100 is applied, adding the requirement on applications  
108738 that if the sign of the option-argument *number* is ‘+’, the *number* option-argument is a non-zero  
108739 decimal integer.

108740 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

108741 SD5-XCU-ERN-114 is applied, updating the OPTIONS section (the `-f` option).

108742 SD5-XCU-ERN-149 is applied.



108743 **NAME**

108744       talk — talk to another user

108745 **SYNOPSIS**108746 UP       talk *address* [*terminal*]108747 **DESCRIPTION**108748       The *talk* utility is a two-way, screen-oriented communication program.108749       When first invoked, *talk* shall send a message similar to:

108750       Message from <unspecified string>  
 108751       talk: connection requested by *your\_address*  
 108752       talk: respond with: talk *your\_address*

108753       to the specified *address*. At this point, the recipient of the message can reply by typing:108754       talk *your\_address*

108755       Once communication is established, the two parties can type simultaneously, with their output  
 108756       displayed in separate regions of the screen. Characters shall be processed as follows:

- 108757       • Typing the <alert> character shall alert the recipient's terminal.
- 108758       • Typing <control>-L shall cause the sender's screen regions to be refreshed.
- 108759       • Typing the erase and kill characters shall affect the sender's terminal in the manner  
 108760       described by the **termios** interface in XBD [Chapter 11](#) (on page 199).
- 108761       • Typing the interrupt or end-of-file characters shall terminate the local *talk* utility. Once the  
 108762       *talk* session has been terminated on one side, the other side of the *talk* session shall be  
 108763       notified that the *talk* session has been terminated and shall be able to do nothing except  
 108764       exit.
- 108765       • Typing characters from *LC\_CTYPE* classifications **print** or **space** shall cause those  
 108766       characters to be sent to the recipient's terminal.
- 108767       • When and only when the *stty iexten* local mode is enabled, the existence and processing of  
 108768       additional special control characters and multi-byte or single-byte functions shall be  
 108769       implementation-defined.
- 108770       • Typing other non-printable characters shall cause implementation-defined sequences of  
 108771       printable characters to be sent to the recipient's terminal.

108772       Permission to be a recipient of a *talk* message can be denied or granted by use of the *mesg* utility.  
 108773       However, a user's privilege may further constrain the domain of accessibility of other users'  
 108774       terminals. The *talk* utility shall fail when the user lacks appropriate privileges to perform the  
 108775       requested action.

108776       Certain block-mode terminals do not have all the capabilities necessary to support the  
 108777       simultaneous exchange of messages required for *talk*. When this type of exchange cannot be  
 108778       supported on such terminals, the implementation may support an exchange with reduced levels  
 108779       of simultaneous interaction or it may report an error describing the terminal-related deficiency.

108780 **OPTIONS**

108781       None.

108782 **OPERANDS**

108783 The following operands shall be supported:

108784 *address* The recipient of the *talk* session. One form of *address* is the *<user name>*, as returned  
 108785 by the *who* utility. Other address formats and how they are handled are  
 108786 unspecified.

108787 *terminal* If the recipient is logged in more than once, the *terminal* argument can be used to  
 108788 indicate the appropriate terminal name. If *terminal* is not specified, the *talk* message  
 108789 shall be displayed on one or more accessible terminals in use by the recipient. The  
 108790 format of *terminal* shall be the same as that returned by the *who* utility.

108791 **STDIN**

108792 Characters read from standard input shall be copied to the recipient's terminal in an unspecified  
 108793 manner. If standard input is not a terminal, *talk* shall write a diagnostic message and exit with a  
 108794 non-zero status.

108795 **INPUT FILES**

108796 None.

108797 **ENVIRONMENT VARIABLES**108798 The following environment variables shall affect the execution of *talk*:

108799 *LANG* Provide a default value for the internationalization variables that are unset or null.  
 108800 (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 108801 variables used to determine the values of locale categories.)

108802 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
 108803 internationalization variables.

108804 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
 108805 characters (for example, single-byte as opposed to multi-byte characters in  
 108806 arguments and input files). If the recipient's locale does not use an *LC\_CTYPE*  
 108807 equivalent to the sender's, the results are undefined.

108808 *LC\_MESSAGES*

108809 Determine the locale that should be used to affect the format and contents of  
 108810 diagnostic messages written to standard error and informative messages written to  
 108811 standard output.

108812 XSI *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

108813 *TERM* Determine the name of the invoker's terminal type. If this variable is unset or null,  
 108814 an unspecified default terminal type shall be used.

108815 **ASYNCHRONOUS EVENTS**

108816 When the *talk* utility receives a SIGINT signal, the utility shall terminate and exit with a zero  
 108817 status. It shall take the standard action for all other signals.

108818 **STDOUT**

108819 If standard output is a terminal, characters copied from the recipient's standard input may be  
 108820 written to standard output. Standard output also may be used for diagnostic messages. If  
 108821 standard output is not a terminal, *talk* shall exit with a non-zero status.

108822 **STDERR**

108823 None.

108824 **OUTPUT FILES**

108825 None.

108826 **EXTENDED DESCRIPTION**

108827 None.

108828 **EXIT STATUS**

108829 The following exit values shall be returned:

108830 0 Successful completion.

108831 >0 An error occurred or *talk* was invoked on a terminal incapable of supporting it.108832 **CONSEQUENCES OF ERRORS**

108833 Default.

108834 **APPLICATION USAGE**

108835 Because the handling of non-printable, non-`<space>` characters is tied to the *stty* description of  
108836 *ixten*, implementation extensions within the terminal driver can be accessed. For example,  
108837 some implementations provide line editing functions with certain control character sequences.

108838 **EXAMPLES**

108839 None.

108840 **RATIONALE**

108841 The *write* utility was included in this volume of POSIX.1-2008 since it can be implemented on all  
108842 terminal types. The *talk* utility, which cannot be implemented on certain terminals, was  
108843 considered to be a “better” communications interface. Both of these programs are in widespread  
108844 use on historical implementations. Therefore, both utilities have been specified.

108845 All references to networking abilities (*talking* to a user on another system) were removed as  
108846 being outside the scope of this volume of POSIX.1-2008.

108847 Historical BSD and System V versions of *talk* terminate both of the conversations when either  
108848 user breaks out of the session. This can lead to adverse consequences if a user unwittingly  
108849 continues to enter text that is interpreted by the shell when the other terminates the session.  
108850 Therefore, the version of *talk* specified by this volume of POSIX.1-2008 requires both users to  
108851 terminate their end of the session explicitly.

108852 Only messages sent to the terminal of the invoking user can be internationalized in any way:

- 108853 • The original “Message from *<unspecified string>* ...” message sent to the terminal of the  
108854 recipient cannot be internationalized because the environment of the recipient is as yet  
108855 inaccessible to the *talk* utility. The environment of the invoking party is irrelevant.
- 108856 • Subsequent communication between the two parties cannot be internationalized because  
108857 the two parties may specify different languages in their environment (and non-portable  
108858 characters cannot be mapped from one language to another).
- 108859 • Neither party can be required to communicate in a language other than C and/or the one  
108860 specified by their environment because unavailable terminal hardware support (for  
108861 example, fonts) may be required.

108862 The text in the STDOUT section reflects the usage of the verb “display” in this section; some *talk*  
108863 implementations actually use standard output to write to the terminal, but this volume of  
108864 POSIX.1-2008 does not require that to be the case.

108865 The format of the terminal name is unspecified, but the descriptions of *ps*, *talk*, *who*, and *write*  
108866 require that they all use or accept the same format.

108867 The handling of non-printable characters is partially implementation-defined because the details  
108868 of mapping them to printable sequences is not needed by the user. Historical implementations,  
108869 for security reasons, disallow the transmission of non-printable characters that may send  
108870 commands to the other terminal.

108871 **FUTURE DIRECTIONS**

108872 None.

108873 **SEE ALSO**

108874 *mesg, stty, who, write*

108875 XBD [Chapter 8](#) (on page 173), [Chapter 11](#) (on page 199)

108876 **CHANGE HISTORY**

108877 First released in Issue 4.

108878 **Issue 6**

108879 This utility is marked as part of the User Portability Utilities option.

108880 **NAME**

108881       tee — duplicate standard input

108882 **SYNOPSIS**108883       tee [-ai] [*file...*]108884 **DESCRIPTION**108885       The *tee* utility shall copy standard input to standard output, making a copy in zero or more files.108886       The *tee* utility shall not buffer output.108887       If the **-a** option is not specified, output files shall be written (see [Section 1.1.1.4](#) (on page 2302)).108888 **OPTIONS**108889       The *tee* utility shall conform to XBD [Section 12.2](#) (on page 215).

108890       The following options shall be supported:

108891       **-a**           Append the output to the files.108892       **-i**           Ignore the SIGINT signal.108893 **OPERANDS**

108894       The following operands shall be supported:

108895       *file*           A pathname of an output file. If a *file* operand is '-', it shall refer to a file named  
 108896       -; implementations shall not treat it as meaning standard output. Processing of at  
 108897       least 13 *file* operands shall be supported.

108898 **STDIN**

108899       The standard input can be of any type.

108900 **INPUT FILES**

108901       None.

108902 **ENVIRONMENT VARIABLES**108903       The following environment variables shall affect the execution of *tee*:

108904       **LANG**           Provide a default value for the internationalization variables that are unset or null.  
 108905                       (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 108906                       variables used to determine the values of locale categories.)

108907       **LC\_ALL**       If set to a non-empty string value, override the values of all the other  
 108908                       internationalization variables.

108909       **LC\_CTYPE**   Determine the locale for the interpretation of sequences of bytes of text data as  
 108910                       characters (for example, single-byte as opposed to multi-byte characters in  
 108911                       arguments).

108912       **LC\_MESSAGES**

108913                       Determine the locale that should be used to affect the format and contents of  
 108914                       diagnostic messages written to standard error.

108915 **XSI**       **NLSPATH**   Determine the location of message catalogs for the processing of *LC\_MESSAGES*.108916 **ASYNCHRONOUS EVENTS**108917       Default, except that if the **-i** option was specified, SIGINT shall be ignored.108918 **STDOUT**

108919       The standard output shall be a copy of the standard input.

**108920 STDERR**

108921 The standard error shall be used only for diagnostic messages.

**108922 OUTPUT FILES**

108923 If any *file* operands are specified, the standard input shall be copied to each named file.

**108924 EXTENDED DESCRIPTION**

108925 None.

**108926 EXIT STATUS**

108927 The following exit values shall be returned:

108928 0 The standard input was successfully copied to all output files.

108929 >0 An error occurred.

**108930 CONSEQUENCES OF ERRORS**

108931 If a write to any successfully opened *file* operand fails, writes to other successfully opened *file*  
108932 operands and standard output shall continue, but the exit status shall be non-zero. Otherwise,  
108933 the default actions specified in [Section 1.4](#) (on page 2310) apply.

**108934 APPLICATION USAGE**

108935 The *tee* utility is usually used in a pipeline, to make a copy of the output of some utility.

108936 The *file* operand is technically optional, but *tee* is no more useful than *cat* when none is specified.

**108937 EXAMPLES**

108938 Save an unsorted intermediate form of the data in a pipeline:

108939 `... | tee unsorted | sort > sorted`

**108940 RATIONALE**

108941 The buffering requirement means that *tee* is not allowed to use ISO C standard fully buffered or  
108942 line-buffered writes. It does not mean that *tee* has to do 1-byte reads followed by 1-byte writes.

108943 It should be noted that early versions of BSD ignore any invalid options and accept a single `'-'`  
108944 as an alternative to `-i`. They also print a message if unable to open a file:

108945 `"tee: cannot access %s\n", <pathname>`

108946 Historical implementations ignore write errors. This is explicitly not permitted by this volume of  
108947 POSIX.1-2008.

108948 Some historical implementations use `O_APPEND` when providing append mode; others use the  
108949 `lseek()` function to seek to the end-of-file after opening the file without `O_APPEND`. This volume  
108950 of POSIX.1-2008 requires functionality equivalent to using `O_APPEND`; see [Section 1.1.1.4](#) (on  
108951 page 2302).

**108952 FUTURE DIRECTIONS**

108953 None.

**108954 SEE ALSO**

108955 [Chapter 1](#) (on page 2301), *cat*

108956 XBD [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 215)

108957 XSH *lseek()*

108958 **CHANGE HISTORY**

108959 First released in Issue 2.

108960 **Issue 6**

108961 IEEE PASC Interpretation 1003.2 #168 is applied.

108962 **Issue 7**

108963 Austin Group Interpretation 1003.1-2001 #092 is applied.

108964 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

108965 **NAME**

108966       test — evaluate expression

108967 **SYNOPSIS**108968       test [*expression*]108969       [ [*expression*] ]108970 **DESCRIPTION**

108971       The *test* utility shall evaluate the *expression* and indicate the result of the evaluation by its exit  
 108972       status. An exit status of zero indicates that the expression evaluated as true and an exit status of  
 108973       1 indicates that the expression evaluated as false.

108974       In the second form of the utility, which uses "[ ]" rather than *test*, the application shall ensure  
 108975       that the square brackets are separate arguments.

108976 **OPTIONS**

108977       The *test* utility shall not recognize the "--" argument in the manner specified by Guideline 10 in  
 108978       XBD [Section 12.2](#) (on page 215).

108979       No options shall be supported.

108980 **OPERANDS**

108981       The application shall ensure that all operators and elements of primaries are presented as  
 108982       separate arguments to the *test* utility.

108983       The following primaries can be used to construct *expression*:

108984       **-b** *pathname* True if *pathname* resolves to an existing directory entry for a block special file. False  
 108985       if *pathname* cannot be resolved, or if *pathname* resolves to an existing directory entry  
 108986       for a file that is not a block special file.

108987       **-c** *pathname* True if *pathname* resolves to an existing directory entry for a character special file.  
 108988       False if *pathname* cannot be resolved, or if *pathname* resolves to an existing directory  
 108989       entry for a file that is not a character special file.

108990       **-d** *pathname* True if *pathname* resolves to an existing directory entry for a directory. False if  
 108991       *pathname* cannot be resolved, or if *pathname* resolves to an existing directory entry  
 108992       for a file that is not a directory.

108993       **-e** *pathname* True if *pathname* resolves to an existing directory entry. False if *pathname* cannot be  
 108994       resolved.

108995       **-f** *pathname* True if *pathname* resolves to an existing directory entry for a regular file. False if  
 108996       *pathname* cannot be resolved, or if *pathname* resolves to an existing directory entry  
 108997       for a file that is not a regular file.

108998       **-g** *pathname* True if *pathname* resolves to an existing directory entry for a file that has its set-  
 108999       group-ID flag set. False if *pathname* cannot be resolved, or if *pathname* resolves to an  
 109000       existing directory entry for a file that does not have its set-group-ID flag set.

109001       **-h** *pathname* True if *pathname* resolves to an existing directory entry for a symbolic link. False if  
 109002       *pathname* cannot be resolved, or if *pathname* resolves to an existing directory entry  
 109003       for a file that is not a symbolic link. If the final component of *pathname* is a  
 109004       symbolic link, that symbolic link is not followed.

109005       **-L** *pathname* True if *pathname* resolves to an existing directory entry for a symbolic link. False if  
 109006       *pathname* cannot be resolved, or if *pathname* resolves to an existing directory entry  
 109007       for a file that is not a symbolic link. If the final component of *pathname* is a  
 109008       symbolic link, that symbolic link is not followed.



|        |                                  |                                                                                                               |
|--------|----------------------------------|---------------------------------------------------------------------------------------------------------------|
| 109009 | <b>-n</b> <i>string</i>          | True if the length of <i>string</i> is non-zero; otherwise, false.                                            |
| 109010 | <b>-p</b> <i>pathname</i>        | True if <i>pathname</i> resolves to an existing directory entry for a FIFO. False if <i>pathname</i>          |
| 109011 |                                  | cannot be resolved, or if <i>pathname</i> resolves to an existing directory entry for a file                  |
| 109012 |                                  | that is not a FIFO.                                                                                           |
| 109013 | <b>-r</b> <i>pathname</i>        | True if <i>pathname</i> resolves to an existing directory entry for a file for which                          |
| 109014 |                                  | permission to read from the file will be granted, as defined in <a href="#">Section 1.1.1.4</a> (on           |
| 109015 |                                  | page 2302). False if <i>pathname</i> cannot be resolved, or if <i>pathname</i> resolves to an                 |
| 109016 |                                  | existing directory entry for a file for which permission to read from the file will not                       |
| 109017 |                                  | be granted.                                                                                                   |
| 109018 | <b>-S</b> <i>pathname</i>        | True if <i>pathname</i> resolves to an existing directory entry for a socket. False if                        |
| 109019 |                                  | <i>pathname</i> cannot be resolved, or if <i>pathname</i> resolves to an existing directory entry             |
| 109020 |                                  | for a file that is not a socket.                                                                              |
| 109021 | <b>-s</b> <i>pathname</i>        | True if <i>pathname</i> resolves to an existing directory entry for a file that has a size                    |
| 109022 |                                  | greater than zero. False if <i>pathname</i> cannot be resolved, or if <i>pathname</i> resolves to an          |
| 109023 |                                  | existing directory entry for a file that does not have a size greater than zero.                              |
| 109024 | <b>-t</b> <i>file_descriptor</i> |                                                                                                               |
| 109025 |                                  | True if file descriptor number <i>file_descriptor</i> is open and is associated with a                        |
| 109026 |                                  | terminal. False if <i>file_descriptor</i> is not a valid file descriptor number, or if file                   |
| 109027 |                                  | descriptor number <i>file_descriptor</i> is not open, or if it is open but is not associated                  |
| 109028 |                                  | with a terminal.                                                                                              |
| 109029 | <b>-u</b> <i>pathname</i>        | True if <i>pathname</i> resolves to an existing directory entry for a file that has its set-                  |
| 109030 |                                  | user-ID flag set. False if <i>pathname</i> cannot be resolved, or if <i>pathname</i> resolves to an           |
| 109031 |                                  | existing directory entry for a file that does not have its set-user-ID flag set.                              |
| 109032 | <b>-w</b> <i>pathname</i>        | True if <i>pathname</i> resolves to an existing directory entry for a file for which                          |
| 109033 |                                  | permission to write to the file will be granted, as defined in <a href="#">Section 1.1.1.4</a> (on page       |
| 109034 |                                  | 2302). False if <i>pathname</i> cannot be resolved, or if <i>pathname</i> resolves to an existing             |
| 109035 |                                  | directory entry for a file for which permission to write to the file will not be                              |
| 109036 |                                  | granted.                                                                                                      |
| 109037 | <b>-x</b> <i>pathname</i>        | True if <i>pathname</i> resolves to an existing directory entry for a file for which                          |
| 109038 |                                  | permission to execute the file (or search it, if it is a directory) will be granted, as                       |
| 109039 |                                  | defined in <a href="#">Section 1.1.1.4</a> (on page 2302). False if <i>pathname</i> cannot be resolved, or if |
| 109040 |                                  | <i>pathname</i> resolves to an existing directory entry for a file for which permission to                    |
| 109041 |                                  | execute (or search) the file will not be granted.                                                             |
| 109042 | <b>-z</b> <i>string</i>          | True if the length of string <i>string</i> is zero; otherwise, false.                                         |
| 109043 | <i>string</i>                    | True if the string <i>string</i> is not the null string; otherwise, false.                                    |
| 109044 | <i>s1</i> = <i>s2</i>            | True if the strings <i>s1</i> and <i>s2</i> are identical; otherwise, false.                                  |
| 109045 | <i>s1</i> != <i>s2</i>           | True if the strings <i>s1</i> and <i>s2</i> are not identical; otherwise, false.                              |
| 109046 | <i>n1</i> -eq <i>n2</i>          | True if the integers <i>n1</i> and <i>n2</i> are algebraically equal; otherwise, false.                       |
| 109047 | <i>n1</i> -ne <i>n2</i>          | True if the integers <i>n1</i> and <i>n2</i> are not algebraically equal; otherwise, false.                   |
| 109048 | <i>n1</i> -gt <i>n2</i>          | True if the integer <i>n1</i> is algebraically greater than the integer <i>n2</i> ; otherwise, false.         |
| 109049 | <i>n1</i> -ge <i>n2</i>          | True if the integer <i>n1</i> is algebraically greater than or equal to the integer <i>n2</i> ;               |
| 109050 |                                  | otherwise, false.                                                                                             |

109051 *n1 -lt n2* True if the integer *n1* is algebraically less than the integer *n2*; otherwise, false.

109052 *n1 -le n2* True if the integer *n1* is algebraically less than or equal to the integer *n2*; otherwise,  
109053 false.

109054 OB XSI *expression1 -a expression2*  
109055 True if both *expression1* and *expression2* are true; otherwise, false. The *-a* binary  
109056 primary is left associative. It has a higher precedence than *-o*.

109057 OB XSI *expression1 -o expression2*  
109058 True if either *expression1* or *expression2* is true; otherwise, false. The *-o* binary  
109059 primary is left associative.

109060 With the exception of the *-h pathname* and *-L pathname* primaries, if a *pathname* argument is a  
109061 symbolic link, *test* shall evaluate the expression by resolving the symbolic link and using the file  
109062 referenced by the link.

109063 These primaries can be combined with the following operators:

109064 *! expression* True if *expression* is false. False if *expression* is true.

109065 OB XSI *( expression )* True if *expression* is true. False if *expression* is false. The parentheses can be used to  
109066 alter the normal precedence and associativity.

109067 The primaries with two elements of the form:

109068 *-primary\_operator primary\_operand*

109069 are known as *unary primaries*. The primaries with three elements in either of the two forms:

109070 *primary\_operand -primary\_operator primary\_operand*  
109071 *primary\_operand primary\_operator primary\_operand*

109072 are known as *binary primaries*. Additional implementation-defined operators and  
109073 *primary\_operators* may be provided by implementations. They shall be of the form *-operator*  
109074 where the first character of *operator* is not a digit.

109075 The algorithm for determining the precedence of the operators and the return value that shall be  
109076 generated is based on the number of arguments presented to *test*. (However, when using the  
109077 "[...]" form, the <right-square-bracket> final argument shall not be counted in this  
109078 algorithm.)

109079 In the following list, \$1, \$2, \$3, and \$4 represent the arguments presented to *test*:

109080 0 arguments: Exit false (1).

109081 1 argument: Exit true (0) if \$1 is not null; otherwise, exit false.

109082 2 arguments:

- If \$1 is '!', exit true if \$2 is null, false if \$2 is not null.
- If \$1 is a unary primary, exit true if the unary test is true, false if the unary test is false.
- Otherwise, produce unspecified results.

109083

109084

109085

109086 3 arguments:

- If \$2 is a binary primary, perform the binary test of \$1 and \$3.
- If \$1 is '!', negate the two-argument test of \$2 and \$3.
- If \$1 is '(' and \$3 is ')', perform the unary test of \$2. On systems that  
109088 OB XSI do not support the XSI option, the results are unspecified if \$1 is '('  
109089 and \$3 is ')'.  
109090

|        |                              |                                                                                                                                      |
|--------|------------------------------|--------------------------------------------------------------------------------------------------------------------------------------|
| 109091 |                              | <ul style="list-style-type: none"> <li>• Otherwise, produce unspecified results.</li> </ul>                                          |
| 109092 | 4 arguments:                 | <ul style="list-style-type: none"> <li>• If \$1 is ' ! ' , negate the three-argument test of \$2, \$3, and \$4.</li> </ul>           |
| 109093 | OB XSI                       | <ul style="list-style-type: none"> <li>• If \$1 is ' ( ' and \$4 is ' ) ' , perform the two-argument test of \$2 and \$3.</li> </ul> |
| 109094 |                              | On systems that do not support the XSI option, the results are                                                                       |
| 109095 |                              | unspecified if \$1 is ' ( ' and \$4 is ' ) ' .                                                                                       |
| 109096 |                              | <ul style="list-style-type: none"> <li>• Otherwise, the results are unspecified.</li> </ul>                                          |
| 109097 | >4 arguments:                | The results are unspecified.                                                                                                         |
| 109098 | OB XSI                       | On XSI-conformant systems, combinations of primaries and operators shall be                                                          |
| 109099 |                              | evaluated using the precedence and associativity rules described previously.                                                         |
| 109100 |                              | In addition, the string comparison binary primaries ' = ' and " ! = " shall have                                                     |
| 109101 |                              | a higher precedence than any unary primary.                                                                                          |
| 109102 | <b>STDIN</b>                 |                                                                                                                                      |
| 109103 |                              | Not used.                                                                                                                            |
| 109104 | <b>INPUT FILES</b>           |                                                                                                                                      |
| 109105 |                              | None.                                                                                                                                |
| 109106 | <b>ENVIRONMENT VARIABLES</b> |                                                                                                                                      |
| 109107 |                              | The following environment variables shall affect the execution of <i>test</i> :                                                      |
| 109108 | <i>LANG</i>                  | Provide a default value for the internationalization variables that are unset or null.                                               |
| 109109 |                              | (See XBD <a href="#">Section 8.2</a> (on page 174) for the precedence of internationalization                                        |
| 109110 |                              | variables used to determine the values of locale categories.)                                                                        |
| 109111 | <i>LC_ALL</i>                | If set to a non-empty string value, override the values of all the other                                                             |
| 109112 |                              | internationalization variables.                                                                                                      |
| 109113 | <i>LC_CTYPE</i>              | Determine the locale for the interpretation of sequences of bytes of text data as                                                    |
| 109114 |                              | characters (for example, single-byte as opposed to multi-byte characters in                                                          |
| 109115 |                              | arguments).                                                                                                                          |
| 109116 | <i>LC_MESSAGES</i>           |                                                                                                                                      |
| 109117 |                              | Determine the locale that should be used to affect the format and contents of                                                        |
| 109118 |                              | diagnostic messages written to standard error.                                                                                       |
| 109119 | XSI <i>NLSPATH</i>           | Determine the location of message catalogs for the processing of <i>LC_MESSAGES</i> .                                                |
| 109120 | <b>ASYNCHRONOUS EVENTS</b>   |                                                                                                                                      |
| 109121 |                              | Default.                                                                                                                             |
| 109122 | <b>STDOUT</b>                |                                                                                                                                      |
| 109123 |                              | Not used.                                                                                                                            |
| 109124 | <b>STDERR</b>                |                                                                                                                                      |
| 109125 |                              | The standard error shall be used only for diagnostic messages.                                                                       |
| 109126 | <b>OUTPUT FILES</b>          |                                                                                                                                      |
| 109127 |                              | None.                                                                                                                                |
| 109128 | <b>EXTENDED DESCRIPTION</b>  |                                                                                                                                      |
| 109129 |                              | None.                                                                                                                                |

109130 **EXIT STATUS**

109131 The following exit values shall be returned:

109132 0 *expression* evaluated to true.109133 1 *expression* evaluated to false or *expression* was missing.

109134 &gt;1 An error occurred.

109135 **CONSEQUENCES OF ERRORS**

109136 Default.

109137 **APPLICATION USAGE**

109138 The XSI extensions specifying the `-a` and `-o` binary primaries and the `' ('` and `') '` operators  
 109139 have been marked obsolescent. (Many expressions using them are ambiguously defined by the  
 109140 grammar depending on the specific expressions being evaluated.) Scripts using these  
 109141 expressions should be converted to the forms given below. Even though many implementations  
 109142 will continue to support these obsolescent forms, scripts should be extremely careful when  
 109143 dealing with user-supplied input that could be confused with these and other primaries and  
 109144 operators. Unless the application developer knows all the cases that produce input to the script,  
 109145 invocations like:

109146 `test "$1" -a "$2"`

109147 should be written as:

109148 `test "$1" && test "$2"`

109149 to avoid problems if a user supplied values such as \$1 set to `' ! '` and \$2 set to the null string.  
 109150 That is, in cases where maximal portability is of concern, replace:

109151 `test expr1 -a expr2`

109152 with:

109153 `test expr1 && test expr2`

109154 and replace:

109155 `test expr1 -o expr2`

109156 with:

109157 `test expr1 || test expr2`

109158 but note that, in *test*, `-a` has higher precedence than `-o` while `"&&"` and `"||"` have equal  
 109159 precedence in the shell.

109160 Parentheses or braces can be used in the shell command language to effect grouping.

109161 Parentheses must be escaped when using *sh*; for example:109162 `test \( expr1 -a expr2 \) -o expr3`

109163 This command is not always portable even on XSI-conformant systems depending on the  
 109164 expressions specified by *expr1*, *expr2*, and *expr3*. The following form can be used instead:

109165 `( test expr1 && test expr2 ) || test expr3`

109166 The two commands:

109167 `test "$1"`109168 `test ! "$1"`

could not be used reliably on some historical systems. Unexpected results would occur if such a *string* expression were used and \$1 expanded to '!', '(', or a known unary primary. Better constructs are:

```
test -n "$1"
test -z "$1"
```

respectively.

Historical systems have also been unreliable given the common construct:

```
test "$response" = "expected string"
```

One of the following is a more reliable form:

```
test "X$response" = "Xexpected string"
test "expected string" = "$response"
```

Note that the second form assumes that *expected string* could not be confused with any unary primary. If *expected string* starts with '-', '(', '!', or even '=', the first form should be used instead. Using the preceding rules without the XSI marked extensions, any of the three comparison forms is reliable, given any input. (However, note that the strings are quoted in all cases.)

Because the string comparison binary primaries, '=' and '!=', have a higher precedence than any unary primary in the greater than 4 argument case, unexpected results can occur if arguments are not properly prepared. For example, in:

```
test -d $1 -o -d $2
```

If \$1 evaluates to a possible directory name of '=', the first three arguments are considered a string comparison, which shall cause a syntax error when the second -d is encountered. One of the following forms prevents this; the second is preferred:

```
test \(-d "$1" \) -o \(-d "$2" \)
test -d "$1" || test -d "$2"
```

Also in the greater than 4 argument case:

```
test "$1" = "bat" -a "$2" = "ball"
```

syntax errors occur if \$1 evaluates to ' (' or ' ! '. One of the following forms prevents this; the third is preferred:

```
test "X$1" = "Xbat" -a "X$2" = "Xball"
test "$1" = "bat" && test "$2" = "ball"
test "X$1" = "Xbat" && test "X$2" = "Xball"
```

## EXAMPLES

1. Exit if there are not two or three arguments (two variations):

```
if [$# -ne 2] && [$# -ne 3]; then exit 1; fi
if [$# -lt 2] || [$# -gt 3]; then exit 1; fi
```

2. Perform a *mkdir* if a directory does not exist:

```
test ! -d tempdir && mkdir tempdir
```

3. Wait for a file to become non-readable:

```
while test -r thefile
do
```

```

109210 sleep 30
109211 done
109212 echo ' "thefile" is no longer readable'

```

4. Perform a command if the argument is one of three strings (two variations):

```

109214 if ["$1" = "pear"] || ["$1" = "grape"] || ["$1" = "apple"]
109215 then
109216 command
109217 fi
109218
109218 case "$1" in
109219 pear|grape|apple) command ;;
109220 esac

```

## RATIONALE

The KornShell-derived conditional command (double bracket `[[ ]]`) was removed from the shell command language description in an early proposal. Objections were raised that the real problem is misuse of the *test* command (`I`), and putting it into the shell is the wrong way to fix the problem. Instead, proper documentation and a new shell reserved word (`!`) are sufficient.

Tests that require multiple *test* operations can be done at the shell level using individual invocations of the *test* command and shell logicals, rather than using the error-prone `-o` flag of *test*.

XSI-conformant systems support more than four arguments.

XSI-conformant systems support the combining of primaries with the following constructs:

*expression1* **-a** *expression2*

True if both *expression1* and *expression2* are true.

*expression1* **-o** *expression2*

True if at least one of *expression1* and *expression2* are true.

**(** *expression* **)**

True if *expression* is true.

In evaluating these more complex combined expressions, the following precedence rules are used:

- The unary primaries have higher precedence than the algebraic binary primaries.
- The unary primaries have lower precedence than the string binary primaries.
- The unary and binary primaries have higher precedence than the unary *string* primary.
- The `!` operator has higher precedence than the `-a` operator, and the `-a` operator has higher precedence than the `-o` operator.
- The `-a` and `-o` operators are left associative.
- The parentheses can be used to alter the normal precedence and associativity.

The BSD and System V versions of `-f` are not the same. The BSD definition was:

**-f** *file*      True if *file* exists and is not a directory.

The SVID version (true if the file exists and is a regular file) was chosen for this volume of POSIX.1-2008 because its use is consistent with the `-b`, `-c`, `-d`, and `-p` operands (*file* exists and is a specific file type).

The **-e** primary, possessing similar functionality to that provided by the C shell, was added because it provides the only way for a shell script to find out if a file exists without trying to open the file. Since implementations are allowed to add additional file types, a portable script cannot use:

```
test -b foo -o -c foo -o -d foo -o -f foo -o -p foo
```

to find out if **foo** is an existing file. On historical BSD systems, the existence of a file could be determined by:

```
test -f foo -o -d foo
```

but there was no easy way to determine that an existing file was a regular file. An early proposal used the KornShell **-a** primary (with the same meaning), but this was changed to **-e** because there were concerns about the high probability of humans confusing the **-a** primary with the **-a** binary operator.

The following options were not included in this volume of POSIX.1-2008, although they are provided by some implementations. These operands should not be used by new implementations for other purposes:

**-k file** True if *file* exists and its sticky bit is set.

**-C file** True if *file* is a contiguous file.

**-V file** True if *file* is a version file.

The following option was not included because it was undocumented in most implementations, has been removed from some implementations (including System V), and the functionality is provided by the shell (see [Section 2.6.2](#) (on page 2328)).

**-l string** The length of the string *string*.

The **-b**, **-c**, **-g**, **-p**, **-u**, and **-x** operands are derived from the SVID; historical BSD does not provide them. The **-k** operand is derived from System V; historical BSD does not provide it.

On historical BSD systems, **test -w directory** always returned false because **test** tried to open the directory for writing, which always fails.

Some additional primaries newly invented or from the KornShell appeared in an early proposal as part of the conditional command ([I II]): *s1 > s2*, *s1 < s2*, *str = pattern*, *str != pattern*, *f1 -nt f2*, *f1 -ot f2*, and *f1 -ef f2*. They were not carried forward into the **test** utility when the conditional command was removed from the shell because they have not been included in the **test** utility built into historical implementations of the *sh* utility.

The **-t file\_descriptor** primary is shown with a mandatory argument because the grammar is ambiguous if it can be omitted. Historical implementations have allowed it to be omitted, providing a default of 1.

It is noted that ' [ ' is not part of the portable filename character set; however, since it is required to be encoded by a single byte, and is part of the portable character set, the name of this utility forms a character string across all supported locales.

## FUTURE DIRECTIONS

None.

## SEE ALSO

[Section 1.1.1.4](#) (on page 2302), *find*

[XBD Chapter 8](#) (on page 173), [Section 12.2](#) (on page 215)

**CHANGE HISTORY**

109293  
109294 First released in Issue 2.

**Issue 5**

109295  
109296 The FUTURE DIRECTIONS section is added.

**Issue 6**

109297  
109298 The **-h** operand is added for symbolic links, and access permission requirements are clarified for  
109299 the **-r**, **-w**, and **-x** operands to align with the IEEE P1003.2b draft standard.

109300 The normative text is reworded to avoid use of the term “must” for application requirements.

109301 The **-L** and **-S** operands are added for symbolic links and sockets.

109302 IEEE Std 1003.1-2001/Cor 1-2002, item XCU/TC1/D6/38 is applied, adding XSI margin  
109303 marking and shading to a line in the OPERANDS section referring to the use of parentheses as  
109304 arguments to the *test* utility.

109305 IEEE Std 1003.1-2001/Cor 2-2004, item XCU/TC2/D6/30 is applied, rewording the existence  
109306 primaries for the *test* utility.

**Issue 7**

109307  
109308 Austin Group Interpretation 1003.1-2001 #107 is applied.

109309 POSIX.1-2008, Technical Corrigendum 1, XCU/TC1-2008/0143 [291] is applied.

+



109310 **NAME**

109311       time — time a simple command

109312 **SYNOPSIS**109313       time [-p] *utility* [*argument...*]109314 **DESCRIPTION**

109315       The *time* utility shall invoke the utility named by the *utility* operand with arguments supplied as  
 109316       the *argument* operands and write a message to standard error that lists timing statistics for the  
 109317       utility. The message shall include the following information:

- 109318       • The elapsed (real) time between invocation of *utility* and its termination.
- 109319       • The User CPU time, equivalent to the sum of the *tms\_utime* and *tms\_cutime* fields returned  
 109320       by the *times()* function defined in the System Interfaces volume of POSIX.1-2008 for the  
 109321       process in which *utility* is executed.
- 109322       • The System CPU time, equivalent to the sum of the *tms\_stime* and *tms\_cstime* fields  
 109323       returned by the *times()* function for the process in which *utility* is executed.

109324       The precision of the timing shall be no less than the granularity defined for the size of the clock  
 109325       tick unit on the system, but the results shall be reported in terms of standard time units (for  
 109326       example, 0.02 seconds, 00:00:00.02, 1m33.75s, 365.21 seconds), not numbers of clock ticks.

109327       When *time* is used as part of a pipeline, the times reported are unspecified, except when it is the  
 109328       sole command within a grouping command (see [Section 2.9.4.1](#), on page 2344) in that pipeline.  
 109329       For example, the commands on the left are unspecified; those on the right report on utilities **a**  
 109330       and **c**, respectively:

```
109331 time a | b | c { time a; } | b | c
109332 a | b | time c a | b | (time c)
```

109333 **OPTIONS**109334       The *time* utility shall conform to XBD [Section 12.2](#) (on page 215).

109335       The following option shall be supported:

109336       **-p**           Write the timing output to standard error in the format shown in the STDERR  
 109337       section.

109338 **OPERANDS**

109339       The following operands shall be supported:

109340       *utility*       The name of a utility that is to be invoked. If the *utility* operand names any of the  
 109341       special built-in utilities in [Section 2.14](#) (on page 2356), the results are undefined.

109342       *argument*     Any string to be supplied as an argument when invoking the utility named by the  
 109343       *utility* operand.

109344 **STDIN**

109345       Not used.

109346 **INPUT FILES**

109347       None.

109348 **ENVIRONMENT VARIABLES**109349       The following environment variables shall affect the execution of *time*:

109350       **LANG**       Provide a default value for the internationalization variables that are unset or null.  
 109351       (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 109352       variables used to determine the values of locale categories.)

|        |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|--------|-------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 109353 | <i>LC_ALL</i>                 | If set to a non-empty string value, override the values of all the other internationalization variables.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 109354 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 109355 | <i>LC_CTYPE</i>               | Determine the locale for the interpretation of sequences of bytes of text data as characters (for example, single-byte as opposed to multi-byte characters in arguments).                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| 109356 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 109357 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 109358 | <i>LC_MESSAGES</i>            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 109359 |                               | Determine the locale that should be used to affect the format and contents of diagnostic and informative messages written to standard error.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 109360 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 109361 | <i>LC_NUMERIC</i>             |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 109362 |                               | Determine the locale for numeric formatting.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 109363 | XSI <i>NLSPATH</i>            | Determine the location of message catalogs for the processing of <i>LC_MESSAGES</i> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| 109364 | <i>PATH</i>                   | Determine the search path that shall be used to locate the utility to be invoked; see XBD Chapter 8 (on page 173).                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 109365 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 109366 | <b>ASYNCHRONOUS EVENTS</b>    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 109367 |                               | Default.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 109368 | <b>STDOUT</b>                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 109369 |                               | Not used.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| 109370 | <b>STDERR</b>                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 109371 |                               | The standard error shall be used to write the timing statistics. If <b>-p</b> is specified, the following format shall be used in the POSIX locale:                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| 109372 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 109373 |                               | "real %f\nuser %f\nsys %f\n", <real seconds>, <user seconds>,<br><system seconds>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 109374 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 109375 |                               | where each floating-point number shall be expressed in seconds. The precision used may be less than the default six digits of %f, but shall be sufficiently precise to accommodate the size of the clock tick on the system (for example, if there were 60 clock ticks per second, at least two digits shall follow the radix character). The number of digits following the radix character shall be no less than one, even if this always results in a trailing zero. The implementation may append white space and additional information following the format shown here. The implementation may also prepend a single empty line before the format shown here. |
| 109376 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 109377 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 109378 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 109379 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 109380 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 109381 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 109382 | <b>OUTPUT FILES</b>           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 109383 |                               | None.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| 109384 | <b>EXTENDED DESCRIPTION</b>   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 109385 |                               | None.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| 109386 | <b>EXIT STATUS</b>            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 109387 |                               | If the <i>utility</i> utility is invoked, the exit status of <i>time</i> shall be the exit status of <i>utility</i> ; otherwise, the <i>time</i> utility shall exit with one of the following values:                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| 109388 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 109389 | 1-125                         | An error occurred in the <i>time</i> utility.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 109390 | 126                           | The utility specified by <i>utility</i> was found but could not be invoked.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 109391 | 127                           | The utility specified by <i>utility</i> could not be found.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 109392 | <b>CONSEQUENCES OF ERRORS</b> |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 109393 |                               | Default.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |

**APPLICATION USAGE**

The *command*, *env*, *nice*, *nohup*, *time*, and *xargs* utilities have been specified to use exit code 127 if an error occurs so that applications can distinguish “failure to find a utility” from “invoked utility exited with an error indication”. The value 127 was chosen because it is not commonly used for other meanings; most utilities use small values for “normal error conditions” and the values above 128 can be confused with termination due to receipt of a signal. The value 126 was chosen in a similar manner to indicate that the utility could be found, but not invoked. Some scripts produce meaningful error messages differentiating the 126 and 127 cases. The distinction between exit codes 126 and 127 is based on KornShell practice that uses 127 when all attempts to *exec* the utility fail with [ENOENT], and uses 126 when any attempt to *exec* the utility fails for any other reason.

**EXAMPLES**

It is frequently desirable to apply *time* to pipelines or lists of commands. This can be done by placing pipelines and command lists in a single file; this file can then be invoked as a utility, and the *time* applies to everything in the file.

Alternatively, the following command can be used to apply *time* to a complex command:

```
time sh -c 'complex-command-line'
```

**RATIONALE**

When the *time* utility was originally proposed to be included in the ISO POSIX-2: 1993 standard, questions were raised about its suitability for inclusion on the grounds that it was not useful for conforming applications, specifically:

- The underlying CPU definitions from the System Interfaces volume of POSIX.1-2008 are vague, so the numeric output could not be compared accurately between systems or even between invocations.
- The creation of portable benchmark programs was outside the scope this volume of POSIX.1-2008.

However, *time* does fit in the scope of user portability. Human judgement can be applied to the analysis of the output, and it could be very useful in hands-on debugging of applications or in providing subjective measures of system performance. Hence it has been included in this volume of POSIX.1-2008.

The default output format has been left unspecified because historical implementations differ greatly in their style of depicting this numeric output. The *-p* option was invented to provide scripts with a common means of obtaining this information.

In the KornShell, *time* is a shell reserved word that can be used to time an entire pipeline, rather than just a simple command. The POSIX definition has been worded to allow this implementation. Consideration was given to invalidating this approach because of the historical model from the C shell and System V shell. However, since the System V *time* utility historically has not produced accurate results in pipeline timing (because the constituent processes are not all owned by the same parent process, as allowed by POSIX), it did not seem worthwhile to break historical KornShell usage.

The term *utility* is used, rather than *command*, to highlight the fact that shell compound commands, pipelines, special built-ins, and so on, cannot be used directly. However, *utility* includes user application programs and shell scripts, not just the standard utilities.

109437 **FUTURE DIRECTIONS**

109438 None.

109439 **SEE ALSO**109440 [Chapter 2](#) (on page 2319), *sh*109441 XBD [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 215)109442 XSH *times()*109443 **CHANGE HISTORY**

109444 First released in Issue 2.

109445 **Issue 6**

109446 This utility is marked as part of the User Portability Utilities option.

109447 **Issue 7**109448 The *time* utility is moved from the User Portability Utilities option to the Base. User Portability  
109449 Utilities is now an option for interactive utilities.

109450 SD5-XCU-ERN-115 is applied, updating the example in the DESCRIPTION.

109451 POSIX.1-2008, Technical Corrigendum 1, XCU/TC1-2008/0144 [266] is applied.

+

109452 **NAME**

109453 touch — change file access and modification times

109454 **SYNOPSIS**109455 touch [-acm] [-r *ref\_file*|-t *time*|-d *date\_time*] *file*...109456 **DESCRIPTION**109457 The *touch* utility shall change the last data modification timestamps, the last data access  
109458 timestamps, or both.109459 The time used can be specified by the -t *time* option-argument, the corresponding *time* fields of  
109460 the file referenced by the -r *ref\_file* option-argument, or the -d *date\_time* option-argument, as  
109461 specified in the following sections. If none of these are specified, *touch* shall use the current time.109462 For each *file* operand, *touch* shall perform actions equivalent to the following functions defined  
109463 in the System Interfaces volume of POSIX.1-2008:

- 109464 1. If *file* does not exist:
  - 109465 a. The *creat*() function is called with the following arguments:
    - 109466 — The *file* operand is used as the *path* argument.
    - 109467 — The value of the bitwise-inclusive OR of S\_IRUSR, S\_IWUSR, S\_IRGRP,  
109468 S\_IWGRP, S\_IROTH, and S\_IWOTH is used as the *mode* argument.
  - 109469 b. The *futimens*() function is called with the following arguments:
    - 109470 — The file descriptor opened in step 1a.
    - 109471 — The access time and the modification time, set as described in the OPTIONS  
109472 section, are used as the first and second elements of the *times* array argument,  
109473 respectively.
- 109474 2. If *file* exists, the *utimensat*() function is called with the following arguments:
  - 109475 a. The AT\_FDCWD special value is used as the *fd* argument.
  - 109476 b. The *file* operand is used as the *path* argument.
  - 109477 c. The access time and the modification time, set as described in the OPTIONS  
109478 section, are used as the first and second elements of the *times* array argument,  
109479 respectively.
  - 109480 d. The *flag* argument is set to zero.

109481 **OPTIONS**109482 The *touch* utility shall conform to XBD [Section 12.2](#) (on page 215).

109483 The following options shall be supported:

- 109484 -a Change the access time of *file*. Do not change the modification time unless -m is  
109485 also specified.
- 109486 -c Do not create a specified *file* if it does not exist. Do not write any diagnostic  
109487 messages concerning this condition.
- 109488 -d *date\_time* Use the specified *date\_time* instead of the current time. The option-argument shall  
109489 be a string of the form:  
109490 YYYY-MM-DDThh:mm:ss[.frac][tz]  
109491 or:

109492 `YYYY-MM-DDThh:mm:ss[,frac][tz]`

109493 where:

- 109494 • YYYY are at least four decimal digits giving the year.
- 109495 • MM, DD, hh, mm, and SS are as with `-t time`.
- 109496 • T is the time designator, and can be replaced by a single <space>.
- 109497 • [,frac] and [,frac] are either empty, or a <period> ('.') or a <comma>
- 109498 (' , ') respectively, followed by one or more decimal digits, specifying a
- 109499 fractional second.
- 109500 • [tz] is either empty, signifying local time, or the letter 'Z', signifying UTC.
- 109501 If [tz] is empty, the resulting time shall be affected by the value of the TZ
- 109502 environment variable.

109503 If the resulting time precedes the Epoch, the behavior is implementation-defined. If

109504 the time cannot be represented as the file's timestamp, *touch* shall exit immediately

109505 with an error status.

109506 **-m** Change the modification time of *file*. Do not change the access time unless **-a** is

109507 also specified.

109508 **-r ref\_file** Use the corresponding time of the file named by the pathname *ref\_file* instead of

109509 the current time.

109510 **-t time** Use the specified *time* instead of the current time. The option-argument shall be a

109511 decimal number of the form:

109512 `[[CC]YY]MMDDhhmm[.SS]`

109513 where each two digits represents the following:

|        |    |                                                 |
|--------|----|-------------------------------------------------|
| 109514 | MM | The month of the year [01,12].                  |
| 109515 | DD | The day of the month [01,31].                   |
| 109516 | hh | The hour of the day [00,23].                    |
| 109517 | mm | The minute of the hour [00,59].                 |
| 109518 | CC | The first two digits of the year (the century). |
| 109519 | YY | The second two digits of the year.              |
| 109520 | SS | The second of the minute [00,60].               |

109521 Both CC and YY shall be optional. If neither is given, the current year shall be

109522 assumed. If YY is specified, but CC is not, CC shall be derived as follows:

| If YY is: | CC becomes: |
|-----------|-------------|
| [69,99]   | 19          |
| [00,68]   | 20          |

109526 **Note:** It is expected that in a future version of this standard the default century inferred

109527 from a 2-digit year will change. (This would apply to all commands accepting a

109528 2-digit year as input.)

109529 The resulting time shall be affected by the value of the TZ environment variable. If

109530 the resulting time value precedes the Epoch, the behavior is implementation-

109531 defined. If the time is out of range for the file's timestamp, *touch* shall exit

109532 immediately with an error status. The range of valid times past the Epoch is  
 109533 implementation-defined, but it shall extend to at least the time 0 hours, 0 minutes,  
 109534 0 seconds, January 1, 2038, Coordinated Universal Time. Some implementations  
 109535 may not be able to represent dates beyond January 18, 2038, because they use  
 109536 **signed int** as a time holder.

109537 The range for *SS* is [00,60] rather than [00,59] because of leap seconds. If *SS* is 60,  
 109538 and the resulting time, as affected by the *TZ* environment variable, does not refer  
 109539 to a leap second, the resulting time shall be one second after a time where *SS* is 59.  
 109540 If *SS* is not given a value, it is assumed to be zero.

109541 If neither the **-a** nor **-m** options were specified, *touch* shall behave as if both the **-a** and **-m**  
 109542 options were specified.

#### 109543 OPERANDS

109544 The following operands shall be supported:

109545 *file* A pathname of a file whose times shall be modified.

#### 109546 STDIN

109547 Not used.

#### 109548 INPUT FILES

109549 None.

#### 109550 ENVIRONMENT VARIABLES

109551 The following environment variables shall affect the execution of *touch*:

109552 *LANG* Provide a default value for the internationalization variables that are unset or null.  
 109553 (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 109554 variables used to determine the values of locale categories.)

109555 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
 109556 internationalization variables.

109557 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
 109558 characters (for example, single-byte as opposed to multi-byte characters in  
 109559 arguments).

109560 *LC\_MESSAGES*

109561 Determine the locale that should be used to affect the format and contents of  
 109562 diagnostic messages written to standard error.

109563 XSI *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

109564 *TZ* Determine the timezone to be used for interpreting the *time* option-argument. If *TZ*  
 109565 is unset or null, an unspecified default timezone shall be used.

#### 109566 ASYNCHRONOUS EVENTS

109567 Default.

#### 109568 STDOUT

109569 Not used.

#### 109570 STDERR

109571 The standard error shall be used only for diagnostic messages.

109572 **OUTPUT FILES**

109573       None.

109574 **EXTENDED DESCRIPTION**

109575       None.

109576 **EXIT STATUS**

109577       The following exit values shall be returned:

109578       0   The utility executed successfully and all requested changes were made.

109579       &gt;0   An error occurred.

109580 **CONSEQUENCES OF ERRORS**

109581       Default.

109582 **APPLICATION USAGE**

109583       The interpretation of time is taken to be *seconds since the Epoch* (see XBD [Section 4.15](#), on page 113). It should be noted that implementations conforming to the System Interfaces volume of POSIX.1-2008 do not take leap seconds into account when computing seconds since the Epoch. When *SS=60* is used, the resulting time always refers to 1 plus *seconds since the Epoch* for a time when *SS=59*.

109588       Although the `-t time` option-argument specifies values in 1969, the access time and modification time fields are defined in terms of seconds since the Epoch (00:00:00 on 1 January 1970 UTC). Therefore, depending on the value of *TZ* when *touch* is run, there is never more than a few valid hours in 1969 and there need not be any valid times in 1969.

109592       One ambiguous situation occurs if `-t time` is not specified, `-r ref_file` is not specified, and the first operand is an eight or ten-digit decimal number. A portable script can avoid this problem by using:

109595       touch -- file

109596       or:

109597       touch ./file

109598       in this case.

109599       If the *T* time designator is replaced by a <space> for the `-d date_time` option-argument, the <space> must be quoted to prevent the shell from splitting the argument.

109601 **EXAMPLES**

109602       Create or update a file called **dwc**; the resulting file has both the last data modification and last data access timestamps set to November 12, 2007 at 10:15:30 local time:

109604       touch -d 2007-11-12T10:15:30 dwc

109605       Create or update a file called **nick**; the resulting file has both the last data modification and last data access timestamps set to November 12, 2007 at 10:15:30 UTC:

109607       touch -d 2007-11-12T10:15:30Z nick

109608       Create or update a file called **gwc**; the resulting file has both the last data modification and last data access timestamps set to November 12, 2007 at 10:15:30 local time with a fractional second timestamp of .002 seconds:

109611       touch -d 2007-11-12T10:15:30.002 gwc

109612       Create or update a file called **ajosey**; the resulting file has both the last data modification and last data access timestamps set to November 12, 2007 at 10:15:30 UTC with a fractional second



109614 timestamp of .002 seconds:

109615 `touch -d "2007-11-12 10:15:30.002Z" ajosey`

109616 Create or update a file called **cathy**; the resulting file has both the last data modification and last  
109617 data access timestamps set to November 12, 2007 at 10:15:00 local time:

109618 `touch -t 200711121015 cathy`

109619 Create or update a file called **drepper**; the resulting file has both the last data modification and  
109620 last data access timestamps set to November 12, 2007 at 10:15:30 local time:

109621 `touch -t 200711121015.30 drepper`

109622 Create or update a file called **ebb9**; the resulting file has both the last data modification and last  
109623 data access timestamps set to November 12, 2007 at 10:15:30 local time:

109624 `touch -t 0711121015.30 ebb9`

109625 Create or update a file called **eggert**; the resulting file has the last data access timestamp set to  
109626 the corresponding time of the file named **mark** instead of the current time. If the file exists, the  
109627 last data modification time is not changed:

109628 `touch -a -r mark eggert`

## 109629 RATIONALE

109630 The functionality of *touch* is described almost entirely through references to functions in the  
109631 System Interfaces volume of POSIX.1-2008. In this way, there is no duplication of effort required  
109632 for describing such side-effects as the relationship of user IDs to the user database, permissions,  
109633 and so on.

109634 There are some significant differences between the *touch* utility in this volume of POSIX.1-2008  
109635 and those in System V and BSD systems. They are upwards-compatible for historical  
109636 applications from both implementations:

- 109637 1. In System V, an ambiguity exists when a pathname that is a decimal number leads the  
109638 operands; it is treated as a time value. In BSD, no *time* value is allowed; files may only be  
109639 *touched* to the current time. The `-t time` construct solves these problems for future  
109640 conforming applications (note that the `-t` option is not historical practice).
- 109641 2. The inclusion of the century digits, *CC*, is also new. Note that a ten-digit *time* value is  
109642 treated as if *YY*, and not *CC*, were specified. The caveat about the range of dates  
109643 following the Epoch was included as recognition that some implementations are not able  
109644 to represent dates beyond 18 January 2038 because they use **signed int** as a time holder.

109645 The `-r` option was added because several comments requested this capability. This option was  
109646 named `-f` in an early proposal, but was changed because the `-f` option is used in the BSD  
109647 version of *touch* with a different meaning.

109648 At least one historical implementation of *touch* incremented the exit code if `-c` was specified and  
109649 the file did not exist. This volume of POSIX.1-2008 requires exit status zero if no errors occur.

109650 In previous version of the standard, if at least two operands are specified, and the first operand  
109651 is an eight or ten-digit decimal integer, the first operand was assumed to be a *date\_time* operand.  
109652 This usage was removed in this version of the standard since it had been marked obsolescent  
109653 previously.

109654 The `-d date_time` format is an ISO 8601:2004 standard complete representation of date and time  
109655 extended format with an optional decimal point or <comma> followed by a string of digits  
109656 following the seconds portion to specify fractions of a second. It is not necessary to recognize

109657 " [+/-] hh:mm" and " [+/-] hh" to specify timezones other than local time and UTC. The *T*  
109658 time designator in the ISO 8601:2004 standard extended format may be replaced by <space>.

109659 **FUTURE DIRECTIONS**

109660 None.

109661 **SEE ALSO**

109662 *date*

109663 XBD [Section 4.15](#) (on page 113), [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 215), [<sys/stat.h>](#)

109664 XSH [creat\(\)](#), [futimens\(\)](#), [time\(\)](#), [utime\(\)](#)

109665 **CHANGE HISTORY**

109666 First released in Issue 2.

109667 **Issue 6**

109668 The obsolescent *date\_time* operand is removed.

109669 The Open Group Corrigendum U027/1 is applied. This extends the range of valid time past the  
109670 Epoch to at least the time 0 hours, 0 minutes, 0 seconds, January 1, 2038, Coordinated Universal  
109671 Time. This is a new requirement on POSIX implementations.

109672 The range for seconds is changed from [00,61] to [00,60] to align with the ISO/IEC 9899:1999  
109673 standard, and to allow for positive leap seconds.

109674 **Issue 7**

109675 Austin Group Interpretation 1003.1-2001 #118 is applied.

109676 Austin Group Interpretation 1003.1-2001 #193 is applied, adding support for subsecond  
109677 timestamps.

109678 SD5-XCU-ERN-45 is applied, adding a new paragraph to the RATIONALE.

109679 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

109680 SD5-XCU-ERN-110 is applied, updating the OPTIONS section.

109681 Changes are made related to support for finegrained timestamps.

109682 **NAME**

109683       tput — change terminal characteristics

109684 **SYNOPSIS**109685       tput [-T *type*] *operand...*109686 **DESCRIPTION**

109687       The *tput* utility shall display terminal-dependent information. The manner in which this  
 109688       information is retrieved is unspecified. The information displayed shall clear the terminal  
 109689       screen, initialize the user's terminal, or reset the user's terminal, depending on the operand  
 109690       given. The exact consequences of displaying this information are unspecified.

109691 **OPTIONS**109692       The *tput* utility shall conform to XBD [Section 12.2](#) (on page 215).

109693       The following option shall be supported:

109694       -T *type*       Indicate the type of terminal. If this option is not supplied and the *TERM* variable  
 109695       is unset or null, an unspecified default terminal type shall be used. The setting of  
 109696       *type* shall take precedence over the value in *TERM*.

109697 **OPERANDS**

109698       The following strings shall be supported as operands by the implementation in the POSIX locale:

109699       **clear**       Display the clear-screen sequence.

109700       **init**        Display the sequence that initializes the user's terminal in an implementation-  
 109701       defined manner.

109702       **reset**       Display the sequence that resets the user's terminal in an implementation-defined  
 109703       manner.

109704       If a terminal does not support any of the operations described by these operands, this shall not  
 109705       be considered an error condition.

109706 **STDIN**

109707       Not used.

109708 **INPUT FILES**

109709       None.

109710 **ENVIRONMENT VARIABLES**109711       The following environment variables shall affect the execution of *tput*:

109712       LANG        Provide a default value for the internationalization variables that are unset or null.  
 109713       (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 109714       variables used to determine the values of locale categories.)

109715       LC\_ALL       If set to a non-empty string value, override the values of all the other  
 109716       internationalization variables.

109717       LC\_CTYPE     Determine the locale for the interpretation of sequences of bytes of text data as  
 109718       characters (for example, single-byte as opposed to multi-byte characters in  
 109719       arguments).

109720       LC\_MESSAGES

109721       Determine the locale that should be used to affect the format and contents of  
 109722       diagnostic messages written to standard error.

109723 XSI **NLSPATH** Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

109724 **TERM** Determine the terminal type. If this variable is unset or null, and if the **-T** option is  
109725 not specified, an unspecified default terminal type shall be used.

109726 **ASYNCHRONOUS EVENTS**

109727 Default.

109728 **STDOUT**

109729 If standard output is a terminal device, it may be used for writing the appropriate sequence to  
109730 clear the screen or reset or initialize the terminal. If standard output is not a terminal device,  
109731 undefined results occur.

109732 **STDERR**

109733 The standard error shall be used only for diagnostic messages.

109734 **OUTPUT FILES**

109735 None.

109736 **EXTENDED DESCRIPTION**

109737 None.

109738 **EXIT STATUS**

109739 The following exit values shall be returned:

109740 0 The requested string was written successfully.

109741 1 Unspecified.

109742 2 Usage error.

109743 3 No information is available about the specified terminal type.

109744 4 The specified operand is invalid.

109745 >4 An error occurred.

109746 **CONSEQUENCES OF ERRORS**

109747 If one of the operands is not available for the terminal, *tput* continues processing the remaining  
109748 operands.

109749 **APPLICATION USAGE**

109750 The difference between resetting and initializing a terminal is left unspecified, as they vary  
109751 greatly based on hardware types. In general, resetting is a more severe action.

109752 Some terminals use control characters to perform the stated functions, and on such terminals it  
109753 might make sense to use *tput* to store the initialization strings in a file or environment variable  
109754 for later use. However, because other terminals might rely on system calls to do this work, the  
109755 standard output cannot be used in a portable manner, such as the following non-portable  
109756 constructs:

109757 ClearVar='tput clear'  
109758 tput reset | mailx -s "Wake Up" ddg

109759 **EXAMPLES**

109760 1. Initialize the terminal according to the type of terminal in the environmental variable  
109761 *TERM*. This command can be included in a **.profile** file.

109762 tput init

109763           2.   Reset a 450 terminal.  
109764           tput -T 450 reset

109765 **RATIONALE**

109766           The list of operands was reduced to a minimum for the following reasons:

- 109767           • The only features chosen were those that were likely to be used by human users interacting  
109768           with a terminal.
- 109769           • Specifying the full *terminfo* set was not considered desirable, but the standard developers  
109770           did not want to select among operands.
- 109771           • This volume of POSIX.1-2008 does not attempt to provide applications with sophisticated  
109772           terminal handling capabilities, as that falls outside of its assigned scope and intersects with  
109773           the responsibilities of other standards bodies.

109774           The difference between resetting and initializing a terminal is left unspecified as this varies  
109775           greatly based on hardware types. In general, resetting is a more severe action.

109776           The exit status of 1 is historically reserved for finding out if a Boolean operand is not set.  
109777           Although the operands were reduced to a minimum, the exit status of 1 should still be reserved  
109778           for the Boolean operands, for those sites that wish to support them.

109779 **FUTURE DIRECTIONS**

109780           None.

109781 **SEE ALSO**

109782           [\*stty\*](#), [\*tabs\*](#)

109783           XBD [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 215)

109784 **CHANGE HISTORY**

109785           First released in Issue 4.

109786 **Issue 6**

109787           This utility is marked as part of the User Portability Utilities option.

109788 **Issue 7**

109789           The *tput* utility is moved from the User Portability Utilities option to the Base. User Portability  
109790           Utilities is now an option for interactive utilities.

109791 **NAME**

109792 tr — translate characters

109793 **SYNOPSIS**109794 tr [-c|-C] [-s] *string1 string2*109795 tr -s [-c|-C] *string1*109796 tr -d [-c|-C] *string1*109797 tr -ds [-c|-C] *string1 string2*109798 **DESCRIPTION**

109799 The *tr* utility shall copy the standard input to the standard output with substitution or deletion  
 109800 of selected characters. The options specified and the *string1* and *string2* operands shall control  
 109801 translations that occur while copying characters and single-character collating elements.

109802 **OPTIONS**109803 The *tr* utility shall conform to XBD [Section 12.2](#) (on page 215).

109804 The following options shall be supported:

109805 **-c** Complement the set of values specified by *string1*. See the EXTENDED  
 109806 DESCRIPTION section.

109807 **-C** Complement the set of characters specified by *string1*. See the EXTENDED  
 109808 DESCRIPTION section.

109809 **-d** Delete all occurrences of input characters that are specified by *string1*.

109810 **-s** Replace instances of repeated characters with a single character, as described in the  
 109811 EXTENDED DESCRIPTION section.

109812 **OPERANDS**

109813 The following operands shall be supported:

109814 *string1, string2*

109815 Translation control strings. Each string shall represent a set of characters to be  
 109816 converted into an array of characters used for the translation. For a detailed  
 109817 description of how the strings are interpreted, see the EXTENDED DESCRIPTION  
 109818 section.

109819 **STDIN**

109820 The standard input can be any type of file.

109821 **INPUT FILES**

109822 None.

109823 **ENVIRONMENT VARIABLES**109824 The following environment variables shall affect the execution of *tr*:

109825 **LANG** Provide a default value for the internationalization variables that are unset or null.  
 109826 (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 109827 variables used to determine the values of locale categories.)

109828 **LC\_ALL** If set to a non-empty string value, override the values of all the other  
 109829 internationalization variables.

109830 **LC\_COLLATE**

109831 Determine the locale for the behavior of range expressions and equivalence classes.

|        |                                                                                                                                                                                                                                                                                                                                                                                              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|--------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 109832 | <i>LC_CTYPE</i>                                                                                                                                                                                                                                                                                                                                                                              | Determine the locale for the interpretation of sequences of bytes of text data as characters (for example, single-byte as opposed to multi-byte characters in arguments) and the behavior of character classes.                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 109833 |                                                                                                                                                                                                                                                                                                                                                                                              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 109834 |                                                                                                                                                                                                                                                                                                                                                                                              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 109835 | <i>LC_MESSAGES</i>                                                                                                                                                                                                                                                                                                                                                                           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 109836 |                                                                                                                                                                                                                                                                                                                                                                                              | Determine the locale that should be used to affect the format and contents of diagnostic messages written to standard error.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 109837 |                                                                                                                                                                                                                                                                                                                                                                                              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 109838 | XSI <i>NLSPATH</i>                                                                                                                                                                                                                                                                                                                                                                           | Determine the location of message catalogs for the processing of <i>LC_MESSAGES</i> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 109839 | <b>ASYNCHRONOUS EVENTS</b>                                                                                                                                                                                                                                                                                                                                                                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 109840 | Default.                                                                                                                                                                                                                                                                                                                                                                                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 109841 | <b>STDOUT</b>                                                                                                                                                                                                                                                                                                                                                                                |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 109842 | The <i>tr</i> output shall be identical to the input, with the exception of the specified transformations.                                                                                                                                                                                                                                                                                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 109843 | <b>STDERR</b>                                                                                                                                                                                                                                                                                                                                                                                |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 109844 | The standard error shall be used only for diagnostic messages.                                                                                                                                                                                                                                                                                                                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 109845 | <b>OUTPUT FILES</b>                                                                                                                                                                                                                                                                                                                                                                          |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 109846 | None.                                                                                                                                                                                                                                                                                                                                                                                        |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 109847 | <b>EXTENDED DESCRIPTION</b>                                                                                                                                                                                                                                                                                                                                                                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 109848 | The operands <i>string1</i> and <i>string2</i> (if specified) define two arrays of characters. The constructs in the following list can be used to specify characters or single-character collating elements. If any of the constructs result in multi-character collating elements, <i>tr</i> shall exclude, without a diagnostic, those multi-character elements from the resulting array. |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 109849 |                                                                                                                                                                                                                                                                                                                                                                                              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 109850 |                                                                                                                                                                                                                                                                                                                                                                                              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 109851 |                                                                                                                                                                                                                                                                                                                                                                                              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 109852 | <i>character</i>                                                                                                                                                                                                                                                                                                                                                                             | Any character not described by one of the conventions below shall represent itself.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 109853 | <i>\octal</i>                                                                                                                                                                                                                                                                                                                                                                                | Octal sequences can be used to represent characters with specific coded values. An octal sequence shall consist of a <backslash> followed by the longest sequence of one, two, or three-octal-digit characters (01234567). The sequence shall cause the value whose encoding is represented by the one, two, or three-digit octal integer to be placed into the array. Multi-byte characters require multiple, concatenated escape sequences of this type, including the leading <backslash> for each byte.                                                                                                                                    |
| 109854 |                                                                                                                                                                                                                                                                                                                                                                                              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 109855 |                                                                                                                                                                                                                                                                                                                                                                                              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 109856 |                                                                                                                                                                                                                                                                                                                                                                                              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 109857 |                                                                                                                                                                                                                                                                                                                                                                                              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 109858 |                                                                                                                                                                                                                                                                                                                                                                                              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 109859 | <i>\character</i>                                                                                                                                                                                                                                                                                                                                                                            | The <backslash>-escape sequences in XBD <a href="#">Table 5-1</a> (on page 121) ('\\', '\a', '\b', '\f', '\n', '\r', '\t', '\v') shall be supported. The results of using any other character, other than an octal digit, following the <backslash> are unspecified. Also, if there is no character following the <backslash>, the results are unspecified.                                                                                                                                                                                                                                                                                    |
| 109860 |                                                                                                                                                                                                                                                                                                                                                                                              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 109861 |                                                                                                                                                                                                                                                                                                                                                                                              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 109862 |                                                                                                                                                                                                                                                                                                                                                                                              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 109863 |                                                                                                                                                                                                                                                                                                                                                                                              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 109864 | <i>c-c</i>                                                                                                                                                                                                                                                                                                                                                                                   | In the POSIX locale, this construct shall represent the range of collating elements between the range endpoints (as long as neither endpoint is an octal sequence of the form <i>\octal</i> ), inclusive, as defined by the collation sequence. The characters or collating elements in the range shall be placed in the array in ascending collation sequence. If the second endpoint precedes the starting endpoint in the collation sequence, it is unspecified whether the range of collating elements is empty, or this construct is treated as invalid. In locales other than the POSIX locale, this construct has unspecified behavior. |
| 109865 |                                                                                                                                                                                                                                                                                                                                                                                              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 109866 |                                                                                                                                                                                                                                                                                                                                                                                              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 109867 |                                                                                                                                                                                                                                                                                                                                                                                              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 109868 |                                                                                                                                                                                                                                                                                                                                                                                              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 109869 |                                                                                                                                                                                                                                                                                                                                                                                              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 109870 |                                                                                                                                                                                                                                                                                                                                                                                              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 109871 |                                                                                                                                                                                                                                                                                                                                                                                              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 109872 |                                                                                                                                                                                                                                                                                                                                                                                              | If either or both of the range endpoints are octal sequences of the form <i>\octal</i> , this shall represent the range of specific coded values between the two range endpoints, inclusive.                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 109873 |                                                                                                                                                                                                                                                                                                                                                                                              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 109874 |                                                                                                                                                                                                                                                                                                                                                                                              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |

|        |                                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|--------|--------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 109875 | [:class:]                            | Represents all characters belonging to the defined character class, as defined by the current setting of the LC_CTYPE locale category. The following character class names shall be accepted when specified in string1:                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| 109876 |                                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 109877 |                                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 109878 |                                      | alnum   blank   digit   lower   punct   upper                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 109879 |                                      | alpha   cntrl   graph   print   space   xdigit                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 109880 | XSI                                  | In addition, character class expressions of the form [:name:] shall be recognized in those locales where the name keyword has been given a charclass definition in the LC_CTYPE category.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| 109881 |                                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 109882 |                                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 109883 |                                      | When both the -d and -s options are specified, any of the character class names shall be accepted in string2. Otherwise, only character class names lower or upper are valid in string2 and then only if the corresponding character class (upper and lower, respectively) is specified in the same relative position in string1. Such a specification shall be interpreted as a request for case conversion. When [:lower:] appears in string1 and [:upper:] appears in string2, the arrays shall contain the characters from the toupper mapping in the LC_CTYPE category of the current locale. When [:upper:] appears in string1 and [:lower:] appears in string2, the arrays shall contain the characters from the tolower mapping in the LC_CTYPE category of the current locale. The first character from each mapping pair shall be in the array for string1 and the second character from each mapping pair shall be in the array for string2 in the same relative position. |
| 109884 |                                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 109885 |                                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 109886 |                                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 109887 |                                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 109888 |                                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 109889 |                                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 109890 |                                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 109891 |                                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 109892 |                                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 109893 |                                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 109894 |                                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 109895 |                                      | Except for case conversion, the characters specified by a character class expression shall be placed in the array in an unspecified order.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 109896 |                                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 109897 |                                      | If the name specified for class does not define a valid character class in the current locale, the behavior is undefined.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| 109898 |                                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 109899 | [=equiv=]                            | Represents all characters or collating elements belonging to the same equivalence class as equiv, as defined by the current setting of the LC_COLLATE locale category. An equivalence class expression shall be allowed only in string1, or in string2 when it is being used by the combined -d and -s options. The characters belonging to the equivalence class shall be placed in the array in an unspecified order.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| 109900 |                                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 109901 |                                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 109902 |                                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 109903 |                                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 109904 | [x*n]                                | Represents n repeated occurrences of the character x. Because this expression is used to map multiple characters to one, it is only valid when it occurs in string2. If n is omitted or is zero, it shall be interpreted as large enough to extend the string2-based sequence to the length of the string1-based sequence. If n has a leading zero, it shall be interpreted as an octal value. Otherwise, it shall be interpreted as a decimal value.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| 109905 |                                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 109906 |                                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 109907 |                                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 109908 |                                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 109909 |                                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 109910 | When the -d option is not specified: |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 109911 |                                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 109912 |                                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 109913 |                                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 109914 |                                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 109915 |                                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 109916 |                                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 109917 |                                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 109918 |                                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |

- If string2 is present, each input character found in the array specified by string1 shall be replaced by the character in the same relative position in the array specified by string2. If the array specified by string2 is shorter than the one specified by string1, or if a character occurs more than once in string1, the results are unspecified.
- If the -C option is specified, the complements of the characters specified by string1 (the set of all characters in the current character set, as defined by the current setting of LC\_CTYPE, except for those actually specified in the string1 operand) shall be placed in the array in ascending collation sequence, as defined by the current setting of LC\_COLLATE.



- If the **-c** option is specified, the complement of the values specified by *string1* shall be placed in the array in ascending order by binary value.
- Because the order in which characters specified by character class expressions or equivalence class expressions is undefined, such expressions should only be used if the intent is to map several characters into one. An exception is case conversion, as described previously.

When the **-d** option is specified:

- Input characters found in the array specified by *string1* shall be deleted.
- When the **-C** option is specified with **-d**, all characters except those specified by *string1* shall be deleted. The contents of *string2* are ignored, unless the **-s** option is also specified.
- When the **-c** option is specified with **-d**, all values except those specified by *string1* shall be deleted. The contents of *string2* shall be ignored, unless the **-s** option is also specified.
- The same string cannot be used for both the **-d** and the **-s** option; when both options are specified, both *string1* (used for deletion) and *string2* (used for squeezing) shall be required.

When the **-s** option is specified, after any deletions or translations have taken place, repeated sequences of the same character shall be replaced by one occurrence of the same character, if the character is found in the array specified by the last operand. If the last operand contains a character class, such as the following example:

```
tr -s '[:space:]'
```

the last operand's array shall contain all of the characters in that character class. However, in a case conversion, as described previously, such as:

```
tr -s '[:upper:]' '[:lower:]'
```

the last operand's array shall contain only those characters defined as the second characters in each of the **toupper** or **tolower** character pairs, as appropriate.

An empty string used for *string1* or *string2* produces undefined results.

## EXIT STATUS

The following exit values shall be returned:

- 0 All input was processed successfully.
- >0 An error occurred.

## CONSEQUENCES OF ERRORS

Default.

## APPLICATION USAGE

If necessary, *string1* and *string2* can be quoted to avoid pattern matching by the shell.

If an ordinary digit (representing itself) is to follow an octal sequence, the octal sequence must use the full three digits to avoid ambiguity.

When *string2* is shorter than *string1*, a difference results between historical System V and BSD systems. A BSD system pads *string2* with the last character found in *string2*. Thus, it is possible to do the following:

```
tr 0123456789 d
```

which would translate all digits to the letter 'd'. Since this area is specifically unspecified in

this volume of POSIX.1-2008, both the BSD and System V behaviors are allowed, but a conforming application cannot rely on the BSD behavior. It would have to code the example in the following way:

```
tr 0123456789 '[d*]'
```

It should be noted that, despite similarities in appearance, the string operands used by *tr* are not regular expressions.

Unlike some historical implementations, this definition of the *tr* utility correctly processes NUL characters in its input stream. NUL characters can be stripped by using:

```
tr -d '\000'
```

## EXAMPLES

1. The following example creates a list of all words in **file1** one per line in **file2**, where a word is taken to be a maximal string of letters.

```
tr -cs "[:alpha:]" "[\n*]" <file1 >file2
```

2. The next example translates all lowercase characters in **file1** to uppercase and writes the results to standard output.

```
tr "[:lower:]" "[:upper:]" <file1
```

3. This example uses an equivalence class to identify accented variants of the base character 'e' in **file1**, which are stripped of diacritical marks and written to **file2**.

```
tr "[=e=]" "[e*]" <file1 >file2
```

## RATIONALE

In some early proposals, an explicit option **-n** was added to disable the historical behavior of stripping NUL characters from the input. It was considered that automatically stripping NUL characters from the input was not correct functionality. However, the removal of **-n** in a later proposal does not remove the requirement that *tr* correctly process NUL characters in its input stream. NUL characters can be stripped by using *tr -d '\000'*.

Historical implementations of *tr* differ widely in syntax and behavior. For example, the BSD version has not needed the bracket characters for the repetition sequence. The *tr* utility syntax is based more closely on the System V and XPG3 model while attempting to accommodate historical BSD implementations. In the case of the short *string2* padding, the decision was to unspecify the behavior and preserve System V and XPG3 scripts, which might find difficulty with the BSD method. The assumption was made that BSD users of *tr* have to make accommodations to meet the syntax defined here. Since it is possible to use the repetition sequence to duplicate the desired behavior, whereas there is no simple way to achieve the System V method, this was the correct, if not desirable, approach.

The use of octal values to specify control characters, while having historical precedents, is not portable. The introduction of escape sequences for control characters should provide the necessary portability. It is recognized that this may cause some historical scripts to break.

An early proposal included support for multi-character collating elements. It was pointed out that, while *tr* does employ some syntactical elements from REs, the aim of *tr* is quite different; ranges, for example, do not have a similar meaning ("any of the chars in the range matches", *versus* "translate each character in the range to the output counterpart"). As a result, the previously included support for multi-character collating elements has been removed. What remains are ranges in current collation order (to support, for example, accented characters), character classes, and equivalence classes.

110004 In XPG3 the `[class:]` and `[equiv=]` conventions are shown with double brackets, as in RE syntax.  
 110005 However, *tr* does not implement RE principles; it just borrows part of the syntax. Consequently,  
 110006 `[class:]` and `[equiv=]` should be regarded as syntactical elements on a par with `[x*n]`, which is  
 110007 not an RE bracket expression.

110008 The standard developers will consider changes to *tr* that allow it to translate characters between  
 110009 different character encodings, or they will consider providing a new utility to accomplish this.

110010 On historical System V systems, a range expression requires enclosing square-brackets, such as:

110011 `tr '[a-z]' '[A-Z]'`

110012 However, BSD-based systems did not require the brackets, and this convention is used here to  
 110013 avoid breaking large numbers of BSD scripts:

110014 `tr a-z A-Z`

110015 The preceding System V script will continue to work because the brackets, treated as regular  
 110016 characters, are translated to themselves. However, any System V script that relied on "a-z"  
 110017 representing the three characters 'a', '-', and 'z' have to be rewritten as "az-".

110018 The ISO POSIX-2: 1993 standard had a `-c` option that behaved similarly to the `-C` option, but did  
 110019 not supply functionality equivalent to the `-c` option specified in POSIX.1-2008. This meant that  
 110020 historical practice of being able to specify `tr -cd\000-\177` (which would delete all bytes with  
 110021 the top bit set) would have no effect because, in the C locale, bytes with the values octal 200 to  
 110022 octal 377 are not characters.

110023 The earlier version also said that octal sequences referred to collating elements and could be  
 110024 placed adjacent to each other to specify multi-byte characters. However, it was noted that this  
 110025 caused ambiguities because *tr* would not be able to tell whether adjacent octal sequences were  
 110026 intending to specify multi-byte characters or multiple single byte characters. POSIX.1-2008  
 110027 specifies that octal sequences always refer to single byte binary values when used to specify an  
 110028 endpoint of a range of collating elements.

110029 Earlier versions of this standard allowed for implementations with bytes other than eight bits,  
 110030 but this has been modified in this version.

#### 110031 FUTURE DIRECTIONS

110032 None.

#### 110033 SEE ALSO

110034 *sed*

110035 XBD [Table 5-1](#) (on page 121), [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 215)

#### 110036 CHANGE HISTORY

110037 First released in Issue 2.

#### 110038 Issue 6

110039 The `-C` operand is added, and the description of the `-c` operand is changed to align with the  
 110040 IEEE P1003.2b draft standard.

110041 The normative text is reworded to avoid use of the term "must" for application requirements.

110042 IEEE Std 1003.1-2001/Cor 2-2004, item XCU/TC2/D6/31 is applied, removing text describing  
 110043 behavior on systems with bytes consisting of more than eight bits.

110044 IEEE Std 1003.1-2001/Cor 2-2004, item XCU/TC2/D6/32 is applied, updating an example in the  
 110045 EXAMPLES section to avoid using unspecified behavior.

110046 IEEE Std 1003.1-2001/Cor 2-2004, item XCU/TC2/D6/33 is applied, making a correction to the

|        |                                                                                                    |   |
|--------|----------------------------------------------------------------------------------------------------|---|
| 110047 | RATIONALE.                                                                                         |   |
| 110048 | <b>Issue 7</b>                                                                                     |   |
| 110049 | SD5-XCU-ERN-30 is applied.                                                                         |   |
| 110050 | SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.                                                  |   |
| 110051 | Austin Group Interpretation 1003.1-2001 #132 is applied, adding rationale to the <i>\character</i> |   |
| 110052 | construct.                                                                                         |   |
| 110053 | POSIX.1-2008, Technical Corrigendum 1, XCU/TC1-2008/0145 [325] is applied.                         | + |

110054 **NAME**

110055           true — return true value

110056 **SYNOPSIS**

110057           true

110058 **DESCRIPTION**110059           The *true* utility shall return with exit code zero.110060 **OPTIONS**

110061           None.

110062 **OPERANDS**

110063           None.

110064 **STDIN**

110065           Not used.

110066 **INPUT FILES**

110067           None.

110068 **ENVIRONMENT VARIABLES**

110069           None.

110070 **ASYNCHRONOUS EVENTS**

110071           Default.

110072 **STDOUT**

110073           Not used.

110074 **STDERR**

110075           Not used.

110076 **OUTPUT FILES**

110077           None.

110078 **EXTENDED DESCRIPTION**

110079           None.

110080 **EXIT STATUS**

110081           Zero.

110082 **CONSEQUENCES OF ERRORS**

110083           None.

110084 **APPLICATION USAGE**

110085           This utility is typically used in shell scripts, as shown in the EXAMPLES section. The special  
110086           built-in utility `:` is sometimes more efficient than *true*.

110087 **EXAMPLES**

110088           This command is executed forever:

```
110089 while true
110090 do
110091 command
110092 done
```

110093 **RATIONALE**

110094 The *true* utility has been retained in this volume of POSIX.1-2008, even though the shell special  
110095 built-in : provides similar functionality, because *true* is widely used in historical scripts and is  
110096 less cryptic to novice script readers.

110097 **FUTURE DIRECTIONS**

110098 None.

110099 **SEE ALSO**

110100 [Section 2.9](#) (on page 2338), *false*

110101 **CHANGE HISTORY**

110102 First released in Issue 2.

110103 **Issue 6**

110104 IEEE Std 1003.1-2001/Cor 1-2002, item XCU/TC1/D6/39 is applied, replacing the terms “None”  
110105 and “Default” from the STDERR and EXIT STATUS sections, respectively, with terms as defined  
110106 in [Section 1.4](#) (on page 2310).

110107 **NAME**

110108           tsort — topological sort

110109 **SYNOPSIS**110110           tsort [*file*]110111 **DESCRIPTION**110112           The *tsort* utility shall write to standard output a totally ordered list of items consistent with a  
110113           partial ordering of items contained in the input.110114           The application shall ensure that the input consists of pairs of items (non-empty strings)  
110115           separated by <blank> characters. Pairs of different items indicate ordering. Pairs of identical  
110116           items indicate presence, but not ordering.110117 **OPTIONS**

110118           None.

110119 **OPERANDS**

110120           The following operand shall be supported:

110121           *file*           A pathname of a text file to order. If no *file* operand is given, the standard input  
110122           shall be used.110123 **STDIN**110124           The standard input shall be used if no *file* operand is specified, and shall be used if the *file*  
110125           operand is '-' and the implementation treats the '-' as meaning standard input. Otherwise,  
110126           the standard input shall not be used. See the INPUT FILES section.110127 **INPUT FILES**

110128           The input file shall be a text file.

110129 **ENVIRONMENT VARIABLES**110130           The following environment variables shall affect the execution of *tsort*:110131           *LANG*           Provide a default value for the internationalization variables that are unset or null.  
110132                           (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
110133                           variables used to determine the values of locale categories.)110134           *LC\_ALL*          If set to a non-empty string value, override the values of all the other  
110135           internationalization variables.110136           *LC\_CTYPE*       Determine the locale for the interpretation of sequences of bytes of text data as  
110137           characters (for example, single-byte as opposed to multi-byte characters in  
110138           arguments and input files).110139           *LC\_MESSAGES*110140                           Determine the locale that should be used to affect the format and contents of  
110141           diagnostic messages written to standard error.110142 XSI           *NLSPATH*   Determine the location of message catalogs for the processing of *LC\_MESSAGES*.110143 **ASYNCHRONOUS EVENTS**

110144           Default.

110145 **STDOUT**110146           The standard output shall be a text file consisting of the order list produced from the partially  
110147           ordered input.

**110148 STDERR**

110149       The standard error shall be used only for diagnostic messages.

**110150 OUTPUT FILES**

110151       None.

**110152 EXTENDED DESCRIPTION**

110153       None.

**110154 EXIT STATUS**

110155       The following exit values shall be returned:

110156       0   Successful completion.

110157       >0  An error occurred.

**110158 CONSEQUENCES OF ERRORS**

110159       Default.

**110160 APPLICATION USAGE**

110161       The *LC\_COLLATE* variable need not affect the actions of *tsort*. The output ordering is not  
110162       lexicographic, but depends on the pairs of items given as input.

**110163 EXAMPLES**

110164       The command:

```
110165 tsort <<EOF
110166 a b c c d e
110167 g g
110168 f g e f
110169 h h
110170 EOF
```

110171       produces the output:

```
110172 a
110173 b
110174 c
110175 d
110176 e
110177 f
110178 g
110179 h
```

**110180 RATIONALE**

110181       None.

**110182 FUTURE DIRECTIONS**

110183       None.

**110184 SEE ALSO**

110185       XBD [Chapter 8](#) (on page 173)

**110186 CHANGE HISTORY**

110187       First released in Issue 2.



110188 **Issue 6**

110189 The normative text is reworded to avoid use of the term “must” for application requirements.

110190 **Issue 7**

110191 Austin Group Interpretation 1003.1-2001 #092 is applied.

110192 The *tsort* utility is moved from the XSI option to the Base.

110193 POSIX.1-2008, Technical Corrigendum 1, XCU/TC1-2008/0146 [241] is applied.

+

110194 **NAME**110195        `tty` — return user's terminal name110196 **SYNOPSIS**110197        `tty`110198 **DESCRIPTION**

110199        The `tty` utility shall write to the standard output the name of the terminal that is open as  
 110200        standard input. The name that is used shall be equivalent to the string that would be returned by  
 110201        the `ttyname()` function defined in the System Interfaces volume of POSIX.1-2008.

110202 **OPTIONS**110203        The `tty` utility shall conform to XBD [Section 12.2](#) (on page 215).110204 **OPERANDS**

110205        None.

110206 **STDIN**

110207        While no input is read from standard input, standard input shall be examined to determine  
 110208        whether or not it is a terminal, and, if so, to determine the name of the terminal.

110209 **INPUT FILES**

110210        None.

110211 **ENVIRONMENT VARIABLES**110212        The following environment variables shall affect the execution of `tty`:

110213        `LANG`        Provide a default value for the internationalization variables that are unset or null.  
 110214                      (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 110215                      variables used to determine the values of locale categories.)

110216        `LC_ALL`      If set to a non-empty string value, override the values of all the other  
 110217                      internationalization variables.

110218        `LC_CTYPE`   Determine the locale for the interpretation of sequences of bytes of text data as  
 110219                      characters (for example, single-byte as opposed to multi-byte characters in  
 110220                      arguments).

110221        `LC_MESSAGES`  
 110222                      Determine the locale that should be used to affect the format and contents of  
 110223                      diagnostic messages written to standard error and informative messages written to  
 110224                      standard output.

110225 `XSI`        `NLSPATH`   Determine the location of message catalogs for the processing of `LC_MESSAGES`.

110226 **ASYNCHRONOUS EVENTS**

110227        Default.

110228 **STDOUT**

110229        If standard input is a terminal device, a pathname of the terminal as specified by the `ttyname()`  
 110230        function defined in the System Interfaces volume of POSIX.1-2008 shall be written in the  
 110231        following format:

110232        `"%s\n", <terminal name>`

110233        Otherwise, a message shall be written indicating that standard input is not connected to a  
 110234        terminal. In the POSIX locale, the `tty` utility shall use the format:

110235        `"not a tty\n"`

**110236 STDERR**

110237 The standard error shall be used only for diagnostic messages.

**110238 OUTPUT FILES**

110239 None.

**110240 EXTENDED DESCRIPTION**

110241 None.

**110242 EXIT STATUS**

110243 The following exit values shall be returned:

110244 0 Standard input is a terminal.

110245 1 Standard input is not a terminal.

110246 >1 An error occurred.

**110247 CONSEQUENCES OF ERRORS**

110248 Default.

**110249 APPLICATION USAGE**

110250 This utility checks the status of the file open as standard input against that of an  
110251 implementation-defined set of files. It is possible that no match can be found, or that the match  
110252 found need not be the same file as that which was opened for standard input (although they are  
110253 the same device).

**110254 EXAMPLES**

110255 None.

**110256 RATIONALE**

110257 None.

**110258 FUTURE DIRECTIONS**

110259 None.

**110260 SEE ALSO**

110261 XBD [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 215)

110262 XSH [isatty\(\)](#), [ttyname\(\)](#)

**110263 CHANGE HISTORY**

110264 First released in Issue 2.

**110265 Issue 5**

110266 The SYNOPSIS is changed to indicate two forms of the command, with the second form marked  
110267 as obsolete. This is a clarification and does not change the functionality published in previous  
110268 issues.

**110269 Issue 6**

110270 The obsolescent `-s` option is removed.

110271 **NAME**

110272           type — write a description of command type

110273 **SYNOPSIS**110274 XSI       type *name*...110275 **DESCRIPTION**110276           The *type* utility shall indicate how each argument would be interpreted if used as a command  
110277           name.110278 **OPTIONS**

110279           None.

110280 **OPERANDS**

110281           The following operand shall be supported:

110282           *name*           A name to be interpreted.110283 **STDIN**

110284           Not used.

110285 **INPUT FILES**

110286           None.

110287 **ENVIRONMENT VARIABLES**110288           The following environment variables shall affect the execution of *type*:110289           *LANG*           Provide a default value for the internationalization variables that are unset or null.  
110290                           (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
110291                           variables used to determine the values of locale categories.)110292           *LC\_ALL*       If set to a non-empty string value, override the values of all the other  
110293                           internationalization variables.110294           *LC\_CTYPE*   Determine the locale for the interpretation of sequences of bytes of text data as  
110295                           characters (for example, single-byte as opposed to multi-byte characters in  
110296                           arguments).110297           *LC\_MESSAGES*  
110298                           Determine the locale that should be used to affect the format and contents of  
110299                           diagnostic messages written to standard error.110300           *NLSPATH*   Determine the location of message catalogs for the processing of *LC\_MESSAGES*.110301           *PATH*       Determine the location of *name*, as described in XBD [Chapter 8](#) (on page 173).110302 **ASYNCHRONOUS EVENTS**

110303           Default.

110304 **STDOUT**110305           The standard output of *type* contains information about each operand in an unspecified format.110306           The information provided typically identifies the operand as a shell built-in, function, alias, or  
110307           keyword, and where applicable, may display the operand's pathname.110308 **STDERR**

110309           The standard error shall be used only for diagnostic messages.

**110310 OUTPUT FILES**

110311 None.

**110312 EXTENDED DESCRIPTION**

110313 None.

**110314 EXIT STATUS**

110315 The following exit values shall be returned:

110316 0 Successful completion.

110317 >0 An error occurred.

**110318 CONSEQUENCES OF ERRORS**

110319 Default.

**110320 APPLICATION USAGE**

110321 Since *type* must be aware of the contents of the current shell execution environment (such as the  
110322 lists of commands, functions, and built-ins processed by *hash*), it is always provided as a shell  
110323 regular built-in. If it is called in a separate utility execution environment, such as one of the  
110324 following:

110325 `nohup type writer`

110326 `find . -type f | xargs type`

110327 it might not produce accurate results.

**110328 EXAMPLES**

110329 None.

**110330 RATIONALE**

110331 None.

**110332 FUTURE DIRECTIONS**

110333 None.

**110334 SEE ALSO**

110335 *command*, *hash*

110336 XBD [Chapter 8](#) (on page 173)

**110337 CHANGE HISTORY**

110338 First released in Issue 2.

110339 **NAME**

110340           ulimit — set or report file size limit

110341 **SYNOPSIS**110342 XSI        ulimit [-f] [*blocks*]110343 **DESCRIPTION**

110344       The *ulimit* utility shall set or report the file-size writing limit imposed on files written by the  
 110345       shell and its child processes (files of any size may be read). Only a process with appropriate  
 110346       privileges can increase the limit.

110347 **OPTIONS**110348       The *ulimit* utility shall conform to XBD [Section 12.2](#) (on page 215).

110349       The following option shall be supported:

110350       -f           Set (or report, if no *blocks* operand is present), the file size limit in blocks. The -f  
 110351       option shall also be the default case.

110352 **OPERANDS**

110353       The following operand shall be supported:

110354       *blocks*       The number of 512-byte blocks to use as the new file size limit.110355 **STDIN**

110356       Not used.

110357 **INPUT FILES**

110358       None.

110359 **ENVIRONMENT VARIABLES**110360       The following environment variables shall affect the execution of *ulimit*:

110361       LANG        Provide a default value for the internationalization variables that are unset or null.  
 110362                   (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 110363                   variables used to determine the values of locale categories.)

110364       LC\_ALL      If set to a non-empty string value, override the values of all the other  
 110365                   internationalization variables.

110366       LC\_CTYPE    Determine the locale for the interpretation of sequences of bytes of text data as  
 110367                   characters (for example, single-byte as opposed to multi-byte characters in  
 110368                   arguments).

110369       LC\_MESSAGES

110370                   Determine the locale that should be used to affect the format and contents of  
 110371                   diagnostic messages written to standard error.

110372       NLSPATH     Determine the location of message catalogs for the processing of *LC\_MESSAGES*.110373 **ASYNCHRONOUS EVENTS**

110374       Default.

110375 **STDOUT**

110376       The standard output shall be used when no *blocks* operand is present. If the current number of  
 110377       blocks is limited, the number of blocks in the current limit shall be written in the following  
 110378       format:

110379       "%d\n", &lt;number of 512-byte blocks&gt;

110380       If there is no current limit on the number of blocks, in the POSIX locale the following format

110381 shall be used:

110382 "unlimited\n"

### 110383 **STDERR**

110384 The standard error shall be used only for diagnostic messages.

### 110385 **OUTPUT FILES**

110386 None.

### 110387 **EXTENDED DESCRIPTION**

110388 None.

### 110389 **EXIT STATUS**

110390 The following exit values shall be returned:

110391 0 Successful completion.

110392 >0 A request for a higher limit was rejected or an error occurred.

### 110393 **CONSEQUENCES OF ERRORS**

110394 Default.

### 110395 **APPLICATION USAGE**

110396 Since *ulimit* affects the current shell execution environment, it is always provided as a shell  
110397 regular built-in. If it is called in a separate utility execution environment, such as one of the  
110398 following:

110399 `nohup ulimit -f 10000`

110400 `env ulimit 10000`

110401 it does not affect the file size limit of the caller's environment.

110402 Once a limit has been decreased by a process, it cannot be increased (unless appropriate  
110403 privileges are involved), even back to the original system limit.

### 110404 **EXAMPLES**

110405 Set the file size limit to 51 200 bytes:

110406 `ulimit -f 100`

### 110407 **RATIONALE**

110408 None.

### 110409 **FUTURE DIRECTIONS**

110410 None.

### 110411 **SEE ALSO**

110412 XBD [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 215)

110413 XSH [ulimit\(\)](#)

### 110414 **CHANGE HISTORY**

110415 First released in Issue 2.

### 110416 **Issue 7**

110417 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

110418 **NAME**

110419       umask — get or set the file mode creation mask

110420 **SYNOPSIS**110421       umask [-S] [*mask*]110422 **DESCRIPTION**

110423       The *umask* utility shall set the file mode creation mask of the current shell execution environment (see [Section 2.12](#), on page 2353) to the value specified by the *mask* operand. This mask shall affect the initial value of the file permission bits of subsequently created files. If *umask* is called in a subshell or separate utility execution environment, such as one of the following:

```
110427 (umask 002)
110428 nohup umask ...
110429 find . -exec umask ... \;
```

110430       it shall not affect the file mode creation mask of the caller's environment.

110431       If the *mask* operand is not specified, the *umask* utility shall write to standard output the value of the file mode creation mask of the invoking process.

110433 **OPTIONS**110434       The *umask* utility shall conform to XBD [Section 12.2](#) (on page 215).

110435       The following option shall be supported:

110436       -S       Produce symbolic output.

110437       The default output style is unspecified, but shall be recognized on a subsequent invocation of *umask* on the same system as a *mask* operand to restore the previous file mode creation mask.

110439 **OPERANDS**

110440       The following operand shall be supported:

110441       *mask*       A string specifying the new file mode creation mask. The string is treated in the same way as the *mode* operand described in the EXTENDED DESCRIPTION section for *chmod*.

110444       For a *symbolic\_mode* value, the new value of the file mode creation mask shall be the logical complement of the file permission bits portion of the file mode specified by the *symbolic\_mode* string.

110447       In a *symbolic\_mode* value, the permissions *op* characters '+' and '-' shall be interpreted relative to the current file mode creation mask; '+' shall cause the bits for the indicated permissions to be cleared in the mask; '-' shall cause the bits for the indicated permissions to be set in the mask.

110451       The interpretation of *mode* values that specify file mode bits other than the file permission bits is unspecified.

110453       In the octal integer form of *mode*, the specified bits are set in the file mode creation mask.

110455       The file mode creation mask shall be set to the resulting numeric value.

110456       The default output of a prior invocation of *umask* on the same system with no operand also shall be recognized as a *mask* operand.



110458 **STDIN**

110459 Not used.

110460 **INPUT FILES**

110461 None.

110462 **ENVIRONMENT VARIABLES**110463 The following environment variables shall affect the execution of *umask*:

110464 *LANG* Provide a default value for the internationalization variables that are unset or null.  
 110465 (See XBD Section 8.2 (on page 174) for the precedence of internationalization  
 110466 variables used to determine the values of locale categories.)

110467 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
 110468 internationalization variables.

110469 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
 110470 characters (for example, single-byte as opposed to multi-byte characters in  
 110471 arguments).

110472 *LC\_MESSAGES*

110473 Determine the locale that should be used to affect the format and contents of  
 110474 diagnostic messages written to standard error.

110475 *XSI NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

110476 **ASYNCHRONOUS EVENTS**

110477 Default.

110478 **STDOUT**

110479 When the *mask* operand is not specified, the *umask* utility shall write a message to standard  
 110480 output that can later be used as a *umask mask* operand.

110481 If *-S* is specified, the message shall be in the following format:

110482 "u=%s,g=%s,o=%s\n", <owner permissions>, <group permissions>,  
 110483 <other permissions>

110484 where the three values shall be combinations of letters from the set {*r*, *w*, *x*}; the presence of a  
 110485 letter shall indicate that the corresponding bit is clear in the file mode creation mask.

110486 If a *mask* operand is specified, there shall be no output written to standard output.110487 **STDERR**

110488 The standard error shall be used only for diagnostic messages.

110489 **OUTPUT FILES**

110490 None.

110491 **EXTENDED DESCRIPTION**

110492 None.

110493 **EXIT STATUS**

110494 The following exit values shall be returned:

110495 0 The file mode creation mask was successfully changed, or no *mask* operand was supplied.

110496 &gt;0 An error occurred.

## 110497 CONSEQUENCES OF ERRORS

110498 Default.

## 110499 APPLICATION USAGE

110500 Since *umask* affects the current shell execution environment, it is generally provided as a shell  
 110501 regular built-in.

110502 In contrast to the negative permission logic provided by the file mode creation mask and the  
 110503 octal number form of the *mask* argument, the symbolic form of the *mask* argument specifies those  
 110504 permissions that are left alone.

## 110505 EXAMPLES

110506 Either of the commands:

110507 `umask a=rx,ug+w`110508 `umask 002`

110509 sets the mode mask so that subsequently created files have their S\_IWOTH bit cleared.

110510 After setting the mode mask with either of the above commands, the *umask* command can be  
 110511 used to write out the current value of the mode mask:

110512 `$ umask`110513 `0002`

110514 (The output format is unspecified, but historical implementations use the octal integer mode  
 110515 format.)

110516 `$ umask -S`110517 `u=rwx,g=rwx,o=rx`

110518 Either of these outputs can be used as the mask operand to a subsequent invocation of the *umask*  
 110519 utility.

110520 Assuming the mode mask is set as above, the command:

110521 `umask g-w`

110522 sets the mode mask so that subsequently created files have their S\_IWGRP and S\_IWOTH bits  
 110523 cleared.

110524 The command:

110525 `umask -- -w`

110526 sets the mode mask so that subsequently created files have all their write bits cleared. Note that  
 110527 *mask* operands `-r`, `-w`, `-x` or anything beginning with a <hyphen>, must be preceded by `--` to  
 110528 keep it from being interpreted as an option.

## 110529 RATIONALE

110530 Since *umask* affects the current shell execution environment, it is generally provided as a shell  
 110531 regular built-in. If it is called in a subshell or separate utility execution environment, such as one  
 110532 of the following:

110533 `(umask 002)`110534 `nohup umask ...`110535 `find . -exec umask ... \;`

110536 it does not affect the file mode creation mask of the environment of the caller.

110537 The description of the historical utility was modified to allow it to use the symbolic modes of

110538 *chmod*. The **-s** option used in early proposals was changed to **-S** because **-s** could be confused  
110539 with a *symbolic\_mode* form of mask referring to the **S\_ISUID** and **S\_ISGID** bits.

110540 The default output style is unspecified to permit implementors to provide migration to the new  
110541 symbolic style at the time most appropriate to their users. A **-o** flag to force octal mode output  
110542 was omitted because the octal mode may not be sufficient to specify all of the information that  
110543 may be present in the file mode creation mask when more secure file access permission checks  
110544 are implemented.

110545 It has been suggested that trusted systems developers might appreciate ameliorating the  
110546 requirement that the mode mask “affects” the file access permissions, since it seems access  
110547 control lists might replace the mode mask to some degree. The wording has been changed to say  
110548 that it affects the file permission bits, and it leaves the details of the behavior of how they affect  
110549 the file access permissions to the description in the System Interfaces volume of POSIX.1-2008.

#### 110550 FUTURE DIRECTIONS

110551 None.

#### 110552 SEE ALSO

110553 [Chapter 2](#) (on page 2319), *chmod*

110554 XBD [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 215)

110555 XSH *umask()*

#### 110556 CHANGE HISTORY

110557 First released in Issue 2.

#### 110558 Issue 6

110559 The following new requirements on POSIX implementations derive from alignment with the  
110560 Single UNIX Specification:

- 110561 • The octal mode is supported.

110562 IEEE Std 1003.1-2001/Cor 2-2004, item XCU/TC2/D6/34 is applied, making a correction to the  
110563 RATIONALE.

#### 110564 Issue 7

110565 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

110566 **NAME**

110567 unalias — remove alias definitions

110568 **SYNOPSIS**

110569 unalias *alias-name*...

110570 unalias -a

110571 **DESCRIPTION**

110572 The *unalias* utility shall remove the definition for each alias name specified. See [Section 2.3.1](#) (on  
110573 page 2322). The aliases shall be removed from the current shell execution environment; see  
110574 [Section 2.12](#) (on page 2353).

110575 **OPTIONS**

110576 The *unalias* utility shall conform to XBD [Section 12.2](#) (on page 215).

110577 The following option shall be supported:

110578 -a Remove all alias definitions from the current shell execution environment.

110579 **OPERANDS**

110580 The following operand shall be supported:

110581 *alias-name* The name of an alias to be removed.

110582 **STDIN**

110583 Not used.

110584 **INPUT FILES**

110585 None.

110586 **ENVIRONMENT VARIABLES**

110587 The following environment variables shall affect the execution of *unalias*:

110588 *LANG* Provide a default value for the internationalization variables that are unset or null.  
110589 (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
110590 variables used to determine the values of locale categories.)

110591 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
110592 internationalization variables.

110593 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
110594 characters (for example, single-byte as opposed to multi-byte characters in  
110595 arguments).

110596 *LC\_MESSAGES*

110597 Determine the locale that should be used to affect the format and contents of  
110598 diagnostic messages written to standard error.

110599 *XSI NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

110600 **ASYNCHRONOUS EVENTS**

110601 Default.

110602 **STDOUT**

110603 Not used.

110604 **STDERR**

110605 The standard error shall be used only for diagnostic messages.

**110606 OUTPUT FILES**

110607 None.

**110608 EXTENDED DESCRIPTION**

110609 None.

**110610 EXIT STATUS**

110611 The following exit values shall be returned:

110612 0 Successful completion.

110613 >0 One of the *alias-name* operands specified did not represent a valid alias definition, or an  
110614 error occurred.

**110615 CONSEQUENCES OF ERRORS**

110616 Default.

**110617 APPLICATION USAGE**

110618 Since *unalias* affects the current shell execution environment, it is generally provided as a shell  
110619 regular built-in.

**110620 EXAMPLES**

110621 None.

**110622 RATIONALE**

110623 The *unalias* description is based on that from historical KornShell implementations. Known  
110624 differences exist between that and the C shell. The KornShell version was adopted to be  
110625 consistent with all the other KornShell features in this volume of POSIX.1-2008, such as  
110626 command line editing.

110627 The *-a* option is the equivalent of the *unalias \** form of the C shell and is provided to address  
110628 security concerns about unknown aliases entering the environment of a user (or application)  
110629 through the allowable implementation-defined predefined alias route or as a result of an *ENV*  
110630 file. (Although *unalias* could be used to simplify the “secure” shell script shown in the *command*  
110631 rationale, it does not obviate the need to quote all command names. An initial call to *unalias -a*  
110632 would have to be quoted in case there was an alias for *unalias*.)

**110633 FUTURE DIRECTIONS**

110634 None.

**110635 SEE ALSO**

110636 [Chapter 2](#) (on page 2319), *alias*

110637 XBD [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 215)

**110638 CHANGE HISTORY**

110639 First released in Issue 4.

**110640 Issue 6**

110641 This utility is marked as part of the User Portability Utilities option.

**110642 Issue 7**

110643 The *unalias* utility is moved from the User Portability Utilities option to the Base. User  
110644 Portability Utilities is now an option for interactive utilities.

110645 **NAME**110646        `uname` — return system name110647 **SYNOPSIS**110648        `uname [-amnrsv]`110649 **DESCRIPTION**

110650        By default, the *uname* utility shall write the operating system name to standard output. When  
 110651        options are specified, symbols representing one or more system characteristics shall be written to  
 110652        the standard output. The format and contents of the symbols are implementation-defined. On  
 110653        systems conforming to the System Interfaces volume of POSIX.1-2008, the symbols written shall  
 110654        be those supported by the *uname()* function as defined in the System Interfaces volume of  
 110655        POSIX.1-2008.

110656 **OPTIONS**110657        The *uname* utility shall conform to XBD [Section 12.2](#) (on page 215).

110658        The following options shall be supported:

- 110659        **-a**            Behave as though all of the options **-mnrsv** were specified.
- 110660        **-m**            Write the name of the hardware type on which the system is running to standard  
 110661                        output.
- 110662        **-n**            Write the name of this node within an implementation-defined communications  
 110663                        network.
- 110664        **-r**            Write the current release level of the operating system implementation.
- 110665        **-s**            Write the name of the implementation of the operating system.
- 110666        **-v**            Write the current version level of this release of the operating system  
 110667                        implementation.

110668        If no options are specified, the *uname* utility shall write the operating system name, as if the **-s**  
 110669        option had been specified.

110670 **OPERANDS**

110671        None.

110672 **STDIN**

110673        Not used.

110674 **INPUT FILES**

110675        None.

110676 **ENVIRONMENT VARIABLES**110677        The following environment variables shall affect the execution of *uname*:

- 110678        **LANG**        Provide a default value for the internationalization variables that are unset or null.  
 110679                        (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 110680                        variables used to determine the values of locale categories.)
- 110681        **LC\_ALL**        If set to a non-empty string value, override the values of all the other  
 110682                        internationalization variables.
- 110683        **LC\_CTYPE**     Determine the locale for the interpretation of sequences of bytes of text data as  
 110684                        characters (for example, single-byte as opposed to multi-byte characters in  
 110685                        arguments).

110686 **LC\_MESSAGES**  
 110687 Determine the locale that should be used to affect the format and contents of  
 110688 diagnostic messages written to standard error.

110689 XSI **NLSPATH** Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

110690 **ASYNCHRONOUS EVENTS**  
 110691 Default.

110692 **STDOUT**  
 110693 By default, the output shall be a single line of the following form:  
 110694 "%s\n", <sysname>  
 110695 If the **-a** option is specified, the output shall be a single line of the following form:  
 110696 "%s %s %s %s %s\n", <sysname>, <nodename>, <release>,  
 110697 <version>, <machine>  
 110698 Additional implementation-defined symbols may be written; all such symbols shall be written at  
 110699 the end of the line of output before the <newline>.  
 110700 If options are specified to select different combinations of the symbols, only those symbols shall  
 110701 be written, in the order shown above for the **-a** option. If a symbol is not selected for writing, its  
 110702 corresponding trailing <blank> characters also shall not be written.

110703 **STDERR**  
 110704 The standard error shall be used only for diagnostic messages.

110705 **OUTPUT FILES**  
 110706 None.

110707 **EXTENDED DESCRIPTION**  
 110708 None.

110709 **EXIT STATUS**  
 110710 The following exit values shall be returned:  
 110711 0 The requested information was successfully written.  
 110712 >0 An error occurred.

110713 **CONSEQUENCES OF ERRORS**  
 110714 Default.

110715 **APPLICATION USAGE**  
 110716 Note that any of the symbols could include embedded <space> characters, which may affect  
 110717 parsing algorithms if multiple options are selected for output.  
 110718 The node name is typically a name that the system uses to identify itself for inter-system  
 110719 communication addressing.

110720 **EXAMPLES**  
 110721 The following command:  
 110722 `uname -sr`  
 110723 writes the operating system name and release level, separated by one or more <blank>  
 110724 characters.

110725 **RATIONALE**

110726 It was suggested that this utility cannot be used portably since the format of the symbols is  
110727 implementation-defined. The POSIX.1 working group could not achieve consensus on defining  
110728 these formats in the underlying *uname()* function, and there was no expectation that this volume  
110729 of POSIX.1-2008 would be any more successful. Some applications may still find this historical  
110730 utility of value. For example, the symbols could be used for system log entries or for comparison  
110731 with operator or user input.

110732 **FUTURE DIRECTIONS**

110733 None.

110734 **SEE ALSO**

110735 XBD [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 215)

110736 XSH [uname\(\)](#)

110737 **CHANGE HISTORY**

110738 First released in Issue 2.



110739 **NAME**

110740 uncompress — expand compressed data

110741 **SYNOPSIS**110742 XSI `uncompress [-cfv] [file...]`110743 **DESCRIPTION**

110744 The *uncompress* utility shall restore files to their original state after they have been compressed  
 110745 using the *compress* utility. If no files are specified, the standard input shall be uncompressed to  
 110746 the standard output. If the invoking process has appropriate privileges, the ownership, modes,  
 110747 access time, and modification time of the original file shall be preserved.

110748 This utility shall support the uncompressing of any files produced by the *compress* utility on the  
 110749 same implementation. For files produced by *compress* on other systems, *uncompress* supports 9 to  
 110750 14-bit compression (see *compress*, **-b**); it is implementation-defined whether values of **-b** greater  
 110751 than 14 are supported.

110752 **OPTIONS**

110753 The *uncompress* utility shall conform to XBD [Section 12.2](#) (on page 215), except that Guideline 1  
 110754 does apply since the utility name has ten letters.

110755 The following options shall be supported:

- 110756 **-c** Write to standard output; no files are changed.
- 110757 **-f** Do not prompt for overwriting files. Except when run in the background, if **-f** is  
 110758 not given the user shall be prompted as to whether an existing file should be  
 110759 overwritten. If the standard input is not a terminal and **-f** is not given, *uncompress*  
 110760 shall write a diagnostic message to standard error and exit with a status greater  
 110761 than zero.
- 110762 **-v** Write messages to standard error concerning the expansion of each file.

110763 **OPERANDS**

110764 The following operand shall be supported:

- 110765 *file* A pathname of a file. If *file* already has the **.Z** suffix specified, it shall be used as the  
 110766 input file and the output file shall be named **file** with the **.Z** suffix removed.  
 110767 Otherwise, *file* shall be used as the name of the output file and **file** with the **.Z**  
 110768 suffix appended shall be used as the input file.

110769 **STDIN**

110770 The standard input shall be used only if no *file* operands are specified, or if a *file* operand is **'-'**.

110771 **INPUT FILES**

110772 Input files shall be in the format produced by the *compress* utility.

110773 **ENVIRONMENT VARIABLES**

110774 The following environment variables shall affect the execution of *uncompress*:

- 110775 **LANG** Provide a default value for the internationalization variables that are unset or null.  
 110776 (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 110777 variables used to determine the values of locale categories.)
- 110778 **LC\_ALL** If set to a non-empty string value, override the values of all the other  
 110779 internationalization variables.

110780 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
110781 characters (for example, single-byte as opposed to multi-byte characters in  
110782 arguments).

110783 *LC\_MESSAGES*  
110784 Determine the locale that should be used to affect the format and contents of  
110785 diagnostic messages written to standard error.

110786 *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

110787 **ASYNCHRONOUS EVENTS**  
110788 Default.

110789 **STDOUT**  
110790 When there are no *file* operands or the *-c* option is specified, the uncompressed output is written  
110791 to standard output.

110792 **STDERR**  
110793 Prompts shall be written to the standard error output under the conditions specified in the  
110794 DESCRIPTION and OPTIONS sections. The prompts shall contain the *file* pathname, but their  
110795 format is otherwise unspecified. Otherwise, the standard error output shall be used only for  
110796 diagnostic messages.

110797 **OUTPUT FILES**  
110798 Output files are the same as the respective input files to *compress*.

110799 **EXTENDED DESCRIPTION**  
110800 None.

110801 **EXIT STATUS**  
110802 The following exit values shall be returned:  
110803 0 Successful completion.  
110804 >0 An error occurred.

110805 **CONSEQUENCES OF ERRORS**  
110806 The input file remains unmodified.

110807 **APPLICATION USAGE**  
110808 The limit of 14 on the *compress -b bits* argument is to achieve portability to all systems (within  
110809 the restrictions imposed by the lack of an explicit published file format). Some implementations  
110810 based on 16-bit architectures cannot support 15 or 16-bit uncompression.

110811 **EXAMPLES**  
110812 None.

110813 **RATIONALE**  
110814 None.

110815 **FUTURE DIRECTIONS**  
110816 None.

110817 **SEE ALSO**  
110818 *compress*, *zcat*  
110819 XBD Chapter 8 (on page 173), Section 12.2 (on page 215)

**110820 CHANGE HISTORY**

110821 First released in Issue 4.

**110822 Issue 6**

110823 The normative text is reworded to avoid use of the term “must” for application requirements.

**110824 Issue 7**

110825 SD5-XCU-ERN-26 is applied, clarifying that this utility is allowed to break the Utility Syntax Guidelines by having ten letters in its name.

110826 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

110828 **NAME**

110829 unexpand — convert spaces to tabs

110830 **SYNOPSIS**110831 unexpand [-a|-t *tablist*] [*file...*]110832 **DESCRIPTION**

110833 The *unexpand* utility shall copy files or standard input to standard output, converting <blank>  
 110834 characters at the beginning of each line into the maximum number of <tab> characters followed  
 110835 by the minimum number of <space> characters needed to fill the same column positions  
 110836 originally filled by the translated <blank> characters. By default, tabstops shall be set at every  
 110837 eighth column position. Each <backspace> shall be copied to the output, and shall cause the  
 110838 column position count for tab calculations to be decremented; the count shall never be  
 110839 decremented to a value less than one.

110840 **OPTIONS**110841 The *unexpand* utility shall conform to XBD [Section 12.2](#) (on page 215).

110842 The following options shall be supported:

110843 **-a** In addition to translating <blank> characters at the beginning of each line,  
 110844 translate all sequences of two or more <blank> characters immediately preceding a  
 110845 tab stop to the maximum number of <tab> characters followed by the minimum  
 110846 number of <space> characters needed to fill the same column positions originally  
 110847 filled by the translated <blank> characters.

110848 **-t *tablist*** Specify the tab stops. The application shall ensure that the *tablist* option-argument  
 110849 is a single argument consisting of a single positive decimal integer or multiple  
 110850 positive decimal integers, separated by <blank> or <comma> characters, in  
 110851 ascending order. If a single number is given, tabs shall be set *tablist* column  
 110852 positions apart instead of the default 8. If multiple numbers are given, the tabs  
 110853 shall be set at those specific column positions.

110854 The application shall ensure that each tab-stop position *N* is an integer value  
 110855 greater than zero, and the list shall be in strictly ascending order. This is taken to  
 110856 mean that, from the start of a line of output, tabbing to position *N* shall cause the  
 110857 next character output to be in the (*N*+1)th column position on that line. When the  
 110858 **-t** option is not specified, the default shall be the equivalent of specifying **-t 8**  
 110859 (except for the interaction with **-a**, described below).

110860 No <space>-to-<tab> conversions shall occur for characters at positions beyond  
 110861 the last of those specified in a multiple tab-stop list.

110862 When **-t** is specified, the presence or absence of the **-a** option shall be ignored;  
 110863 conversion shall not be limited to the processing of leading <blank> characters.

110864 **OPERANDS**

110865 The following operand shall be supported:

110866 *file* A pathname of a text file to be used as input.

110867 **STDIN**

110868 See the INPUT FILES section.

110869 **INPUT FILES**

110870 The input files shall be text files.

110871 **ENVIRONMENT VARIABLES**

110872 The following environment variables shall affect the execution of *unexpand*:

110873 *LANG* Provide a default value for the internationalization variables that are unset or null.  
 110874 (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 110875 variables used to determine the values of locale categories.)

110876 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
 110877 internationalization variables.

110878 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
 110879 characters (for example, single-byte as opposed to multi-byte characters in  
 110880 arguments and input files), the processing of <tab> and <space> characters, and  
 110881 for the determination of the width in column positions each character would  
 110882 occupy on an output device.

110883 *LC\_MESSAGES*

110884 Determine the locale that should be used to affect the format and contents of  
 110885 diagnostic messages written to standard error.

110886 XSI *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

110887 **ASYNCHRONOUS EVENTS**

110888 Default.

110889 **STDOUT**

110890 The standard output shall be equivalent to the input files with the specified <space>-to-<tab>  
 110891 conversions.

110892 **STDERR**

110893 The standard error shall be used only for diagnostic messages.

110894 **OUTPUT FILES**

110895 None.

110896 **EXTENDED DESCRIPTION**

110897 None.

110898 **EXIT STATUS**

110899 The following exit values shall be returned:

110900 0 Successful completion.

110901 >0 An error occurred.

110902 **CONSEQUENCES OF ERRORS**

110903 Default.

110904 **APPLICATION USAGE**

110905 One non-intuitive aspect of *unexpand* is its restriction to leading <space> characters when neither  
 110906 *-a* nor *-t* is specified. Users who always want to convert all <space> characters in a file can  
 110907 easily alias *unexpand* to use the *-a* or *-t 8* option.

110908 **EXAMPLES**

110909 None.

110910 **RATIONALE**

110911 On several occasions, consideration was given to adding a *-t* option to the *unexpand* utility to  
 110912 complement the *-t* in *expand* (see [expand](#)). The historical intent of *unexpand* was to translate  
 110913 multiple <blank> characters into tab stops, where tab stops were a multiple of eight column

110914 positions on most UNIX systems. An early proposal omitted `-t` because it seemed outside the  
110915 scope of the User Portability Utilities option; it was not described in any of the base documents.  
110916 However, hard-coding tab stops every eight columns was not suitable for the international  
110917 community and broke historical precedents for some vendors in the FORTRAN community, so  
110918 `-t` was restored in conjunction with the list of valid extension categories considered by the  
110919 standard developers. Thus, *unexpand* is now the logical converse of *expand*.

#### 110920 FUTURE DIRECTIONS

110921 None.

#### 110922 SEE ALSO

110923 *expand*, *tabs*

110924 XBD [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 215)

#### 110925 CHANGE HISTORY

110926 First released in Issue 4.

#### 110927 Issue 6

110928 This utility is marked as part of the User Portability Utilities option.

110929 The definition of the `LC_CTYPE` environment variable is changed to align with the  
110930 IEEE P1003.2b draft standard.

110931 The normative text is reworded to avoid use of the term “must” for application requirements.

#### 110932 Issue 7

110933 The *unexpand* utility is moved from the User Portability Utilities option to the Base. User  
110934 Portability Utilities is now an option for interactive utilities.

110935 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

110936 **NAME**110937 unget — undo a previous get of an SCCS file (**DEVELOPMENT**)110938 **SYNOPSIS**110939 XSI unget [-ns] [-r *SID*] *file...*110940 **DESCRIPTION**110941 The *unget* utility shall reverse the effect of a *get -e* done prior to creating the intended new delta.110942 **OPTIONS**110943 The *unget* utility shall conform to XBD [Section 12.2](#) (on page 215).

110944 The following options shall be supported:

110945 **-r** *SID* Uniquely identify which delta is no longer intended. (This would have been  
 110946 specified by *get* as the new delta.) The use of this option is necessary only if two or  
 110947 more outstanding *get* commands for editing on the same SCCS file were done by  
 110948 the same person (login name).

110949 **-s** Suppress the writing to standard output of the intended delta's *SID*.

110950 **-n** Retain the file that was obtained by *get*, which would normally be removed from  
 110951 the current directory.

110952 **OPERANDS**

110953 The following operands shall be supported:

110954 *file* A pathname of an existing SCCS file or a directory. If *file* is a directory, the *unget*  
 110955 utility shall behave as though each file in the directory were specified as a named  
 110956 file, except that non-SCCS files (last component of the pathname does not begin  
 110957 with **s**.) and unreadable files shall be silently ignored.

110958 If exactly one *file* operand appears, and it is **'-'**, the standard input shall be read;  
 110959 each line of the standard input shall be taken to be the name of an SCCS file to be  
 110960 processed. Non-SCCS files and unreadable files shall be silently ignored.

110961 **STDIN**

110962 The standard input shall be a text file used only when the *file* operand is specified as **'-'**. Each  
 110963 line of the text file shall be interpreted as an SCCS pathname.

110964 **INPUT FILES**

110965 Any SCCS files processed shall be files of an unspecified format.

110966 **ENVIRONMENT VARIABLES**110967 The following environment variables shall affect the execution of *unget*:

110968 **LANG** Provide a default value for the internationalization variables that are unset or null.  
 110969 (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 110970 variables used to determine the values of locale categories.)

110971 **LC\_ALL** If set to a non-empty string value, override the values of all the other  
 110972 internationalization variables.

110973 **LC\_CTYPE** Determine the locale for the interpretation of sequences of bytes of text data as  
 110974 characters (for example, single-byte as opposed to multi-byte characters in  
 110975 arguments and input files).

110976 **LC\_MESSAGES**

110977 Determine the locale that should be used to affect the format and contents of  
 110978 diagnostic messages written to standard error.

110979 *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

110980 **ASYNCHRONOUS EVENTS**

110981 Default.

110982 **STDOUT**

110983 The standard output shall consist of a line for each file, in the following format:

110984 "%s\n", <*SID removed from file*>

110985 If there is more than one named file or if a directory or standard input is named, each pathname

110986 shall be written before each of the preceding lines:

110987 "\n%s:\n", <*pathname*>

110988 **STDERR**

110989 The standard error shall be used only for diagnostic messages.

110990 **OUTPUT FILES**

110991 Any SCCS files updated shall be files of an unspecified format. During processing of a *file*, a

110992 locking *z-file*, as described in *get*, and a *q-file* (a working copy of the *p-file*), may be created and

110993 deleted. The *p-file* and *g-file*, as described in *get*, shall be deleted.

110994 **EXTENDED DESCRIPTION**

110995 None.

110996 **EXIT STATUS**

110997 The following exit values shall be returned:

110998 0 Successful completion.

110999 >0 An error occurred.

111000 **CONSEQUENCES OF ERRORS**

111001 Default.

111002 **APPLICATION USAGE**

111003 None.

111004 **EXAMPLES**

111005 None.

111006 **RATIONALE**

111007 None.

111008 **FUTURE DIRECTIONS**

111009 None.

111010 **SEE ALSO**

111011 *delta*, *get*, *sact*

111012 XBD [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 215)

111013 **CHANGE HISTORY**

111014 First released in Issue 2.

111015 **Issue 6**

111016 The normative text is reworded to avoid use of the term “must” for application requirements.



111017 **Issue 7**  
111018

SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

111019 **NAME**

111020        **uniq** — report or filter out repeated lines in a file

111021 **SYNOPSIS**

111022        **uniq** [**-c|-d|-u**] [**-f** *fields*] [**-s** *char*] [*input\_file* [*output\_file*]]

111023 **DESCRIPTION**

111024        The *uniq* utility shall read an input file comparing adjacent lines, and write one copy of each  
111025        input line on the output. The second and succeeding copies of repeated adjacent input lines shall  
111026        not be written. The trailing <newline> of each line in the input shall be ignored when doing  
111027        comparisons.

111028        Repeated lines in the input shall not be detected if they are not adjacent.

111029 **OPTIONS**

111030        The *uniq* utility shall conform to XBD [Section 12.2](#) (on page 215), except that '+' may be  
111031        recognized as an option delimiter as well as '-'.  
111032        The following options shall be supported:

111033        **-c**            Precede each output line with a count of the number of times the line occurred in  
111034                      the input.

111035        **-d**            Suppress the writing of lines that are not repeated in the input.

111036        **-f** *fields*    Ignore the first *fields* fields on each input line when doing comparisons, where  
111037                      *fields* is a positive decimal integer. A field is the maximal string matched by the  
111038                      basic regular expression:

111039                      `[[:blank:]]*([[:blank:]]*)`

111040                      If the *fields* option-argument specifies more fields than appear on an input line, a  
111041                      null string shall be used for comparison.

111042        **-s** *chars*    Ignore the first *chars* characters when doing comparisons, where *chars* shall be a  
111043                      positive decimal integer. If specified in conjunction with the **-f** option, the first  
111044                      *chars* characters after the first *fields* fields shall be ignored. If the *chars* option-  
111045                      argument specifies more characters than remain on an input line, a null string shall  
111046                      be used for comparison.

111047        **-u**            Suppress the writing of lines that are repeated in the input.

111048 **OPERANDS**

111049        The following operands shall be supported:

111050        *input\_file*    A pathname of the input file. If the *input\_file* operand is not specified, or if the  
111051                      *input\_file* is '-', the standard input shall be used.

111052        *output\_file*   A pathname of the output file. If the *output\_file* operand is not specified, the  
111053                      standard output shall be used. The results are unspecified if the file named by  
111054                      *output\_file* is the file named by *input\_file*.

111055 **STDIN**

111056        The standard input shall be used only if no *input\_file* operand is specified or if *input\_file* is '-'.  
111057        See the INPUT FILES section.

111058 **INPUT FILES**

111059        The input file shall be a text file.

111060 **ENVIRONMENT VARIABLES**

111061 The following environment variables shall affect the execution of *uniq*:

111062 *LANG* Provide a default value for the internationalization variables that are unset or null.  
 111063 (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 111064 variables used to determine the values of locale categories.)

111065 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
 111066 internationalization variables.

111067 *LC\_COLLATE*  
 111068 Determine the locale for ordering rules.

111069 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
 111070 characters (for example, single-byte as opposed to multi-byte characters in  
 111071 arguments and input files) and which characters constitute a <blank> in the  
 111072 current locale.

111073 *LC\_MESSAGES*  
 111074 Determine the locale that should be used to affect the format and contents of  
 111075 diagnostic messages written to standard error.

111076 XSI *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

111077 **ASYNCHRONOUS EVENTS**

111078 Default.

111079 **STDOUT**

111080 The standard output shall be used if no *output\_file* operand is specified, and shall be used if the  
 111081 *output\_file* operand is '-' and the implementation treats the '-' as meaning standard output.  
 111082 Otherwise, the standard output shall not be used. See the OUTPUT FILES section.

111083 **STDERR**

111084 The standard error shall be used only for diagnostic messages.

111085 **OUTPUT FILES**

111086 If the *-c* option is specified, the output file shall be empty or each line shall be of the form:

111087 "%d %s", <number of duplicates>, <line>

111088 otherwise, the output file shall be empty or each line shall be of the form:

111089 "%s", <line>

111090 **EXTENDED DESCRIPTION**

111091 None.

111092 **EXIT STATUS**

111093 The following exit values shall be returned:

111094 0 The utility executed successfully.

111095 >0 An error occurred.

111096 **CONSEQUENCES OF ERRORS**

111097 Default.

## 111098 APPLICATION USAGE

111099 The *sort* utility can be used to cause repeated lines to be adjacent in the input file.

## 111100 EXAMPLES

111101 The following input file data (but flushed left) was used for a test series on *uniq*:

```
111102 #01 foo0 bar0 fool bar1
111103 #02 bar0 fool bar1 fool
111104 #03 foo0 bar0 fool bar1
111105 #04
111106 #05 foo0 bar0 fool bar1
111107 #06 foo0 bar0 fool bar1
111108 #07 bar0 fool bar1 foo0
```

111109 What follows is a series of test invocations of the *uniq* utility that use a mixture of *uniq* options  
 111110 against the input file data. These tests verify the meaning of *adjacent*. The *uniq* utility views the  
 111111 input data as a sequence of strings delimited by '*\n*'. Accordingly, for the *fields*th member of  
 111112 the sequence, *uniq* interprets unique or repeated adjacent lines strictly relative to the *fields*+1th  
 111113 member.

- 111114 1. This first example tests the line counting option, comparing each line of the input file data  
 111115 starting from the second field:

```
111116 uniq -c -f 1 uniq_0I.t
111117 1 #01 foo0 bar0 fool bar1
111118 1 #02 bar0 fool bar1 fool
111119 1 #03 foo0 bar0 fool bar1
111120 1 #04
111121 2 #05 foo0 bar0 fool bar1
111122 1 #07 bar0 fool bar1 foo0
```

111123 The number '*2*', prefixing the fifth line of output, signifies that the *uniq* utility detected a  
 111124 pair of repeated lines. Given the input data, this can only be true when *uniq* is run using  
 111125 the *-f 1* option (which shall cause *uniq* to ignore the first field on each input line).

- 111126 2. The second example tests the option to suppress unique lines, comparing each line of the  
 111127 input file data starting from the second field:

```
111128 uniq -d -f 1 uniq_0I.t
111129 #05 foo0 bar0 fool bar1
```

- 111130 3. This test suppresses repeated lines, comparing each line of the input file data starting  
 111131 from the second field:

```
111132 uniq -u -f 1 uniq_0I.t
111133 #01 foo0 bar0 fool bar1
111134 #02 bar0 fool bar1 fool
111135 #03 foo0 bar0 fool bar1
111136 #04
111137 #07 bar0 fool bar1 foo0
```

- 111138 4. This suppresses unique lines, comparing each line of the input file data starting from the  
 111139 third character:

```
111140 uniq -d -s 2 uniq_0I.t
```

111141 In the last example, the *uniq* utility found no input matching the above criteria.

**111142 RATIONALE**

111143 Some historical implementations have limited lines to be 1 080 bytes in length, which does not  
111144 meet the implied {LINE\_MAX} limit.

111145 Earlier versions of this standard allowed the *-number* and *+number* options. These options are no  
111146 longer specified by POSIX.1-2008 but may be present in some implementations.

**111147 FUTURE DIRECTIONS**

111148 None.

**111149 SEE ALSO**

111150 *comm*, *sort*

111151 XBD [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 215)

**111152 CHANGE HISTORY**

111153 First released in Issue 2.

**111154 Issue 6**

111155 The obsolescent SYNOPSIS and associated text are removed.

111156 The normative text is reworded to avoid use of the term “must” for application requirements.

111157 IEEE Std 1003.1-2001/Cor 1-2002, item XCU/TC1/D6/40 is applied, adding *LC\_COLLATE* to the  
111158 ENVIRONMENT VARIABLES section, and changing “the application shall ensure that” in the  
111159 OUTPUT FILES section.

**111160 Issue 7**

111161 Austin Group Interpretation 1003.1-2001 #027 is applied, clarifying that ‘+’ may be recognized  
111162 as an option delimiter in the OPTIONS section.

111163 Austin Group Interpretation 1003.1-2001 #092 is applied.

111164 Austin Group Interpretation 1003.1-2001 #133 is applied, clarifying the behavior of the trailing  
111165 <newline>.

111166 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

111167 SD5-XCU-ERN-141 is applied, updating the EXAMPLES section.

111168 **NAME**

111169 unlink — call the *unlink()* function

111170 **SYNOPSIS**

111171 XSI `unlink file`

111172 **DESCRIPTION**

111173 The *unlink* utility shall perform the function call:

111174 `unlink (file);`

111175 A user may need appropriate privileges to invoke the *unlink* utility.

111176 **OPTIONS**

111177 None.

111178 **OPERANDS**

111179 The following operands shall be supported:

111180 *file* The pathname of an existing file.

111181 **STDIN**

111182 Not used.

111183 **INPUT FILES**

111184 Not used.

111185 **ENVIRONMENT VARIABLES**

111186 The following environment variables shall affect the execution of *unlink*:

111187 *LANG* Provide a default value for the internationalization variables that are unset or null.  
111188 (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
111189 variables used to determine the values of locale categories.)

111190 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
111191 internationalization variables.

111192 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
111193 characters (for example, single-byte as opposed to multi-byte characters in  
111194 arguments).

111195 *LC\_MESSAGES*  
111196 Determine the locale that should be used to affect the format and contents of  
111197 diagnostic messages written to standard error.

111198 *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

111199 **ASYNCHRONOUS EVENTS**

111200 Default.

111201 **STDOUT**

111202 None.

111203 **STDERR**

111204 The standard error shall be used only for diagnostic messages.

111205 **OUTPUT FILES**

111206 None.

111207 **EXTENDED DESCRIPTION**

111208 None.

111209 **EXIT STATUS**

111210 The following exit values shall be returned:

111211 0 Successful completion.

111212 &gt;0 An error occurred.

111213 **CONSEQUENCES OF ERRORS**

111214 Default.

111215 **APPLICATION USAGE**

111216 None.

111217 **EXAMPLES**

111218 None.

111219 **RATIONALE**

111220 None.

111221 **FUTURE DIRECTIONS**

111222 None.

111223 **SEE ALSO**111224 *link, rm*111225 XBD [Chapter 8](#) (on page 173)111226 XSH *[unlink\(\)](#)*111227 **CHANGE HISTORY**

111228 First released in Issue 5.

111229 **NAME**

111230 uucp — system-to-system copy

111231 **SYNOPSIS**111232 UU `uucp [-cCdfjmr] [-n user] source-file... destination-file`111233 **DESCRIPTION**

111234 The *uucp* utility shall copy files named by the *source-file* argument to the *destination-file* argument.  
 111235 The files named can be on local or remote systems.

111236 The *uucp* utility cannot guarantee support for all character encodings in all circumstances. For  
 111237 example, transmission data may be restricted to 7 bits by the underlying network, 8-bit data and  
 111238 filenames need not be portable to non-internationalized systems, and so on. Under these  
 111239 circumstances, it is recommended that only characters defined in the ISO/IEC 646:1991  
 111240 standard International Reference Version (equivalent to ASCII) 7-bit range of characters be used,  
 111241 and that only characters defined in the portable filename character set be used for naming files.  
 111242 The protocol for transfer of files is unspecified by POSIX.1-2008.

111243 Typical implementations of this utility require a communications line configured to use XBD  
 111244 [Chapter 11](#) (on page 199), but other communications means may be used. On systems where  
 111245 there are no available communications means (either temporarily or permanently), this utility  
 111246 shall write an error message describing the problem and exit with a non-zero exit status.

111247 **OPTIONS**111248 The *uucp* utility shall conform to XBD [Section 12.2](#) (on page 215).

111249 The following options shall be supported:

- 111250 **-c** Do not copy local file to the spool directory for transfer to the remote machine  
 111251 (default).
- 111252 **-C** Force the copy of local files to the spool directory for transfer.
- 111253 **-d** Make all necessary directories for the file copy (default).
- 111254 **-f** Do not make intermediate directories for the file copy.
- 111255 **-j** Write the job identification string to standard output. This job identification can be  
 111256 used by *uustat* to obtain the status or terminate a job.
- 111257 **-m** Send mail to the requester when the copy is completed.
- 111258 **-n user** Notify *user* on the remote system that a file was sent.
- 111259 **-r** Do not start the file transfer; just queue the job.

111260 **OPERANDS**

111261 The following operands shall be supported:

111262 *destination-file, source-file*

111263 A pathname of a file to be copied to, or from, respectively. Either name can be a  
 111264 pathname on the local machine, or can have the form:

111265 *system-name!pathname*

111266 where *system-name* is taken from a list of system names that *uucp* knows about.  
 111267 The destination *system-name* can also be a list of names such as:

111268 *system-name!system-name!...!system-name!pathname*

111269 in which case, an attempt is made to send the file via the specified route to the



111270 destination. Care should be taken to ensure that intermediate nodes in the route  
111271 are willing to forward information.

111272 The shell pattern matching notation characters ' ? ' , ' \* ' , and " [ . . . ] " appearing  
111273 in *pathname* shall be expanded on the appropriate system.

111274 Pathnames can be one of:

- 111275 1. An absolute pathname.
- 111276 2. A pathname preceded by *~user* where *user* is a login name on the specified  
111277 system and is replaced by that user's login directory. Note that if an invalid  
111278 login is specified, the default is to the public directory (called *PUBDIR*; the  
111279 actual location of *PUBDIR* is implementation-defined).
- 111280 3. A pathname preceded by *~/destination* where *destination* is appended to  
111281 *PUBDIR*.

111282 **Note:** This destination is treated as a filename unless more than one file is being  
111283 transferred by this request or the destination is already a directory. To  
111284 ensure that it is a directory, follow the destination with a ' / ' . For  
111285 example, *~/dan/* as the destination makes the directory **PUBDIR/dan** if it  
111286 does not exist and puts the requested files in that directory.

- 111287 4. Anything else shall be prefixed by the current directory.

111288 If the result is an erroneous pathname for the remote system, the copy shall fail. If  
111289 the *destination-file* is a directory, the last part of the *source-file* name shall be used.

111290 The read, write, and execute permissions given by *uucp* are implementation-  
111291 defined.

## 111292 STDIN

111293 Not used.

## 111294 INPUT FILES

111295 The files to be copied are regular files.

## 111296 ENVIRONMENT VARIABLES

111297 The following environment variables shall affect the execution of *uucp*:

111298 *LANG* Provide a default value for the internationalization variables that are unset or null.  
111299 (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
111300 variables used to determine the values of locale categories.)

111301 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
111302 internationalization variables.

111303 *LC\_COLLATE*  
111304 Determine the locale for the behavior of ranges, equivalence classes, and multi-  
111305 character collating elements within bracketed filename patterns.

111306 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
111307 characters (for example, single-byte as opposed to multi-byte characters in  
111308 arguments and input files) and the behavior of character classes within bracketed  
111309 filename patterns (for example, " ' [ : lower : ] \* ' ").

111310 *LC\_MESSAGES*  
111311 Determine the locale that should be used to affect the format and contents of  
111312 diagnostic messages written to standard error, and informative messages written  
111313 to standard output.

- 111314 *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.
- 111315 **ASYNCHRONOUS EVENTS**
- 111316 Default.
- 111317 **STDOUT**
- 111318 Not used.
- 111319 **STDERR**
- 111320 The standard error shall be used only for diagnostic messages.
- 111321 **OUTPUT FILES**
- 111322 The output files (which may be on other systems) are copies of the input files.
- 111323 If *-m* is used, mail files are modified.
- 111324 **EXTENDED DESCRIPTION**
- 111325 None.
- 111326 **EXIT STATUS**
- 111327 The following exit values shall be returned:
- 111328 0 Successful completion.
- 111329 >0 An error occurred.
- 111330 **CONSEQUENCES OF ERRORS**
- 111331 Default.
- 111332 **APPLICATION USAGE**
- 111333 This utility is part of the UUCP Utilities option and need not be supported by all implementations.
- 111334
- 111335 The domain of remotely accessible files can (and for obvious security reasons usually should) be severely restricted.
- 111336
- 111337 Note that the *'!'* character in addresses has to be escaped when using *cs**h* as a command interpreter because of its history substitution syntax. For *ksh* and *sh* the escape is not necessary, but may be used.
- 111338
- 111339
- 111340 As noted above, shell metacharacters appearing in pathnames are expanded on the appropriate system. On an internationalized system, this is done under the control of local settings of *LC\_COLLATE* and *LC\_CTYPE*. Thus, care should be taken when using bracketed filename patterns, as collation and typing rules may vary from one system to another. Also be aware that certain types of expression (that is, equivalence classes, character classes, and collating symbols) need not be supported on non-internationalized systems.
- 111341
- 111342
- 111343
- 111344
- 111345
- 111346 **EXAMPLES**
- 111347 None.
- 111348 **RATIONALE**
- 111349 None.
- 111350 **FUTURE DIRECTIONS**
- 111351 None.
- 111352 **SEE ALSO**
- 111353 *mailx*, *uuencode*, *uustat*, *uux*
- 111354 XBD [Chapter 8](#) (on page 173), [Chapter 11](#) (on page 199), [Section 12.2](#) (on page 215)

111355 **CHANGE HISTORY**

111356 First released in Issue 2.

111357 **Issue 6**111358 The *LC\_TIME* and *TZ* entries are removed from the ENVIRONMENT VARIABLES section.111359 The UN margin codes and associated shading are removed from the *-C*, *-f*, *-j*, *-n*, and *-r*  
111360 options in response to The Open Group Base Resolution bwg2001-003.111361 **Issue 7**

111362 SD5-XCU-ERN-46 is applied, moving this utility to the UUCP Utilities Option Group.

111363 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

111364 **NAME**

111365       uudecode — decode a binary file

111366 **SYNOPSIS**111367       uudecode [-o *outfile*] [*file*]111368 **DESCRIPTION**

111369       The *uudecode* utility shall read a file, or standard input if no file is specified, that includes data  
 111370       created by the *uuencode* utility. The *uudecode* utility shall scan the input file, searching for data  
 111371       compatible with one of the formats specified in *uuencode*, and attempt to create or overwrite the  
 111372       file described by the data (or overridden by the **-o** option). The pathname shall be contained in  
 111373       the data or specified by the **-o** option. The file access permission bits and contents for the file to  
 111374       be produced shall be contained in that data. The mode bits of the created file (other than  
 111375       standard output) shall be set from the file access permission bits contained in the data; that is,  
 111376       other attributes of the mode, including the file mode creation mask (see *umask*), shall not affect  
 111377       the file being produced. If either of the *op* characters '+' and '-' (see *chmod*) are specified in  
 111378       symbolic mode, the initial mode on which those operations are based is unspecified.

111379       If the pathname of the file to be produced exists, and the user does not have write permission on  
 111380       that file, *uudecode* shall terminate with an error. If the pathname of the file to be produced exists,  
 111381       and the user has write permission on that file, the existing file shall be overwritten.

111382       If the input data was produced by *uuencode* on a system with a different number of bits per byte  
 111383       than on the target system, the results of *uudecode* are unspecified.

111384 **OPTIONS**111385       The *uudecode* utility shall conform to XBD [Section 12.2](#) (on page 215).

111386       The following option shall be supported by the implementation:

111387       **-o *outfile***    A pathname of a file that shall be used instead of any pathname contained in the  
 111388       input data. Specifying an *outfile* option-argument of **/dev/stdout** shall indicate  
 111389       standard output.

111390 **OPERANDS**

111391       The following operand shall be supported:

111392       *file*            The pathname of a file containing the output of *uuencode*.111393 **STDIN**

111394       See the INPUT FILES section.

111395 **INPUT FILES**111396       The input files shall be files containing the output of *uuencode*.111397 **ENVIRONMENT VARIABLES**111398       The following environment variables shall affect the execution of *uudecode*:

111399       **LANG**            Provide a default value for the internationalization variables that are unset or null.  
 111400       (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 111401       variables used to determine the values of locale categories.)

111402       **LC\_ALL**          If set to a non-empty string value, override the values of all the other  
 111403       internationalization variables.

111404       **LC\_CTYPE**       Determine the locale for the interpretation of sequences of bytes of text data as  
 111405       characters (for example, single-byte as opposed to multi-byte characters in  
 111406       arguments and input files).

111407 *LC\_MESSAGES*  
 111408 Determine the locale that should be used to affect the format and contents of  
 111409 diagnostic messages written to standard error.

111410 XSI *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

111411 **ASYNCHRONOUS EVENTS**  
 111412 Default.

111413 **STDOUT**  
 111414 If the file data header encoded by *uuencode* is `-` or `/dev/stdout`, or the `-o /dev/stdout` option  
 111415 overrides the file data, the standard output shall be in the same format as the file originally  
 111416 encoded by *uuencode*. Otherwise, the standard output shall not be used.

111417 **STDERR**  
 111418 The standard error shall be used only for diagnostic messages.

111419 **OUTPUT FILES**  
 111420 The output file shall be in the same format as the file originally encoded by *uuencode*.

111421 **EXTENDED DESCRIPTION**  
 111422 None.

111423 **EXIT STATUS**  
 111424 The following exit values shall be returned:  
 111425 0 Successful completion.  
 111426 >0 An error occurred.

111427 **CONSEQUENCES OF ERRORS**  
 111428 Default.

111429 **APPLICATION USAGE**  
 111430 The user who is invoking *uudecode* must have write permission on any file being created.

111431 The output of *uuencode* is essentially an encoded bit stream that is not cognizant of byte  
 111432 boundaries. It is possible that a 9-bit byte target machine can process input from an 8-bit source,  
 111433 if it is aware of the requirement, but the reverse is unlikely to be satisfying. Of course, the only  
 111434 data that is meaningful for such a transfer between architectures is generally character data.

111435 **EXAMPLES**  
 111436 None.

111437 **RATIONALE**  
 111438 Input files are not necessarily text files, as stated by an early proposal. Although the *uuencode*  
 111439 output is a text file, that output could have been wrapped within another file or mail message  
 111440 that is not a text file.

111441 The `-o` option is not historical practice, but was added at the request of WG15 so that the user  
 111442 could override the target pathname without having to edit the input data itself.

111443 In early drafts, the `[-o outfile]` option-argument allowed the use of `-` to mean standard output.  
 111444 The symbol `-` has only been used previously in POSIX.1-2008 as a standard input indicator. The  
 111445 standard developers did not wish to overload the meaning of `-` in this manner. The `/dev/stdout`  
 111446 concept exists on most modern systems. The `/dev/stdout` syntax does not refer to a new special  
 111447 file. It is just a magic cookie to specify standard output.

111448 **FUTURE DIRECTIONS**

111449 None.

111450 **SEE ALSO**111451 *chmod*, *umask*, *uuencode*111452 XBD [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 215)111453 **CHANGE HISTORY**

111454 First released in Issue 4.

111455 **Issue 6**

111456 This utility is marked as part of the User Portability Utilities option.

111457 The **-o** *outfile* option is added, as specified in the IEEE P1003.2b draft standard.

111458 The normative text is reworded to avoid use of the term “must” for application requirements.

111459 IEEE Std 1003.1-2001/Cor 2-2004, item XCU/TC2/D6/35 is applied, clarifying in the  
111460 DESCRIPTION that the initial mode used if either of the *op* characters is '+' or '-' is  
111461 unspecified.

111462 **Issue 7**

111463 The *uudecode* utility is moved from the User Portability Utilities option to the Base. User  
111464 Portability Utilities is now an option for interactive utilities.

111465 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

111466 **NAME**

111467 uuencode — encode a binary file

111468 **SYNOPSIS**111469 uuencode [-m] [*file*] *decode\_pathname*111470 **DESCRIPTION**

111471 The *uuencode* utility shall write an encoded version of the named input file, or standard input if  
 111472 no *file* is specified, to standard output. The output shall be encoded using one of the algorithms  
 111473 described in the STDOUT section and shall include the file access permission bits (in *chmod* octal  
 111474 or symbolic notation) of the input file and the *decode\_pathname*, for re-creation of the file on  
 111475 another system that conforms to this volume of POSIX.1-2008.

111476 **OPTIONS**111477 The *uuencode* utility shall conform to XBD [Section 12.2](#) (on page 215).

111478 The following option shall be supported by the implementation:

111479 **-m** Encode the output using the MIME Base64 algorithm described in STDOUT. If **-m**  
 111480 is not specified, the historical algorithm described in STDOUT shall be used.

111481 **OPERANDS**

111482 The following operands shall be supported:

111483 *decode\_pathname*

111484 The pathname of the file into which the *uudecode* utility shall place the decoded  
 111485 file. Specifying a *decode\_pathname* operand of **/dev/stdout** shall indicate that  
 111486 *uudecode* is to use standard output. If there are characters in *decode\_pathname* that  
 111487 are not in the portable filename character set the results are unspecified.

111488 *file* A pathname of the file to be encoded.

111489 **STDIN**

111490 See the INPUT FILES section.

111491 **INPUT FILES**

111492 Input files can be files of any type.

111493 **ENVIRONMENT VARIABLES**111494 The following environment variables shall affect the execution of *uuencode*:

111495 **LANG** Provide a default value for the internationalization variables that are unset or null.  
 111496 (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 111497 variables used to determine the values of locale categories.)

111498 **LC\_ALL** If set to a non-empty string value, override the values of all the other  
 111499 internationalization variables.

111500 **LC\_CTYPE** Determine the locale for the interpretation of sequences of bytes of text data as  
 111501 characters (for example, single-byte as opposed to multi-byte characters in  
 111502 arguments and input files).

111503 **LC\_MESSAGES**

111504 Determine the locale that should be used to affect the format and contents of  
 111505 diagnostic messages written to standard error.

111506 **XS** **NLSPATH** Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

111507 **ASYNCHRONOUS EVENTS**

111508 Default.

111509 **STDOUT**111510 **uuencode Base64 Algorithm**

111511 The standard output shall be a text file (encoded in the character set of the current locale) that  
 111512 begins with the line:

111513 "begin-base64Δ%sΔs\n", <mode>, <decode\_pathname>

111514 and ends with the line:

111515 "====\n"

111516 In both cases, the lines shall have no preceding or trailing <blank> characters.

111517 The encoding process represents 24-bit groups of input bits as output strings of four encoded  
 111518 characters. Proceeding from left to right, a 24-bit input group shall be formed by concatenating  
 111519 three 8-bit input groups. Each 24-bit input group then shall be treated as four concatenated 6-bit  
 111520 groups, each of which shall be translated into a single digit in the Base64 alphabet. When  
 111521 encoding a bit stream via the Base64 encoding, the bit stream shall be presumed to be ordered  
 111522 with the most-significant bit first. That is, the first bit in the stream shall be the high-order bit in  
 111523 the first byte, and the eighth bit shall be the low-order bit in the first byte, and so on. Each 6-bit  
 111524 group is used as an index into an array of 64 printable characters, as shown in [Table 4-22](#).

111525 **Table 4-22 uuencode Base64 Values**

| Value | Encoding | Value | Encoding | Value | Encoding | Value | Encoding |
|-------|----------|-------|----------|-------|----------|-------|----------|
| 0     | A        | 17    | R        | 34    | i        | 51    | z        |
| 1     | B        | 18    | S        | 35    | j        | 52    | 0        |
| 2     | C        | 19    | T        | 36    | k        | 53    | 1        |
| 3     | D        | 20    | U        | 37    | l        | 54    | 2        |
| 4     | E        | 21    | V        | 38    | m        | 55    | 3        |
| 5     | F        | 22    | W        | 39    | n        | 56    | 4        |
| 6     | G        | 23    | X        | 40    | o        | 57    | 5        |
| 7     | H        | 24    | Y        | 41    | p        | 58    | 6        |
| 8     | I        | 25    | Z        | 42    | q        | 59    | 7        |
| 9     | J        | 26    | a        | 43    | r        | 60    | 8        |
| 10    | K        | 27    | b        | 44    | s        | 61    | 9        |
| 11    | L        | 28    | c        | 45    | t        | 62    | +        |
| 12    | M        | 29    | d        | 46    | u        | 63    | /        |
| 13    | N        | 30    | e        | 47    | v        | (pad) | =        |
| 14    | O        | 31    | f        | 48    | w        |       |          |
| 15    | P        | 32    | g        | 49    | x        |       |          |
| 16    | Q        | 33    | h        | 50    | y        |       |          |

111544 The character referenced by the index shall be placed in the output string.

111545 The output stream (encoded bytes) shall be represented in lines of no more than 76 characters  
 111546 each. All line breaks or other characters not found in the table shall be ignored by decoding  
 111547 software (see [uudecode](#)).

111548 Special processing shall be performed if fewer than 24 bits are available at the end of a message  
 111549 or encapsulated part of a message. A full encoding quantum shall always be completed at the



end of a message. When fewer than 24 input bits are available in an input group, zero bits shall be added (on the right) to form an integral number of 6-bit groups. Output character positions that are not required to represent actual input data shall be set to the character '='. Since all Base64 input is an integral number of octets, only the following cases can arise:

1. The final quantum of encoding input is an integral multiple of 24 bits; here, the final unit of encoded output shall be an integral multiple of 4 characters with no '=' padding.
2. The final quantum of encoding input is exactly 16 bits; here, the final unit of encoded output shall be three characters followed by one '=' padding character.
3. The final quantum of encoding input is exactly 8 bits; here, the final unit of encoded output shall be two characters followed by two '=' padding characters.

A terminating "====" evaluates to nothing and denotes the end of the encoded data.

### uuencode Historical Algorithm

The standard output shall be a text file (encoded in the character set of the current locale) that begins with the line:

```
"beginΔ%sΔ%s\n" <mode>, <decode_pathname>
```

and ends with the line:

```
"end\n"
```

In both cases, the lines shall have no preceding or trailing <blank> characters.

The algorithm that shall be used for lines in between **begin** and **end** takes three octets as input and writes four characters of output by splitting the input at six-bit intervals into four octets, containing data in the lower six bits only. These octets shall be converted to characters by adding a value of 0x20 to each octet, so that each octet is in the range [0x20,0x5f], and then it shall be assumed to represent a printable character in the ISO/IEC 646:1991 standard encoded character set. It then shall be translated into the corresponding character codes for the codeset in use in the current locale. (For example, the octet 0x41, representing 'A', would be translated to 'A' in the current codeset, such as 0xc1 if it were EBCDIC.)

Where the bits of two octets are combined, the least significant bits of the first octet shall be shifted left and combined with the most significant bits of the second octet shifted right. Thus the three octets *A*, *B*, *C* shall be converted into the four octets:

```
0x20 + ((A >> 2) & 0x3F)
0x20 + (((A << 4) | ((B >> 4) & 0xF)) & 0x3F)
0x20 + (((B << 2) | ((C >> 6) & 0x3)) & 0x3F)
0x20 + ((C) & 0x3F)
```

These octets then shall be translated into the local character set.

Each encoded line contains a length character, equal to the number of characters to be decoded plus 0x20 translated to the local character set as described above, followed by the encoded characters. The maximum number of octets to be encoded on each line shall be 45.

### STDERR

The standard error shall be used only for diagnostic messages.

**111589 OUTPUT FILES**

111590       None.

**111591 EXTENDED DESCRIPTION**

111592       None.

**111593 EXIT STATUS**

111594       The following exit values shall be returned:

111595       0   Successful completion.

111596       >0  An error occurred.

**111597 CONSEQUENCES OF ERRORS**

111598       Default.

**111599 APPLICATION USAGE**

111600       The file is expanded by 35 percent (each three octets become four, plus control information) causing it to take longer to transmit.

111602       Since this utility is intended to create files to be used for data interchange between systems with possibly different codesets, and to represent binary data as a text file, the ISO/IEC 646:1991 standard was chosen for a midpoint in the algorithm as a known reference point. The output from *uuencode* is a text file on the local system. If the output were in the ISO/IEC 646:1991 standard codeset, it might not be a text file (at least because the <newline> characters might not match), and the goal of creating a text file would be defeated. If this text file was then carried to another machine with the same codeset, it would be perfectly compatible with that system's *uudecode*. If it was transmitted over a mail system or sent to a machine with a different codeset, it is assumed that, as for every other text file, some translation mechanism would convert it (by the time it reached a user on the other system) into an appropriate codeset. This translation only makes sense from the local codeset, not if the file has been put into a ISO/IEC 646:1991 standard representation first. Similarly, files processed by *uuencode* can be placed in *pax* archives, intermixed with other text files in the same codeset.

**111615 EXAMPLES**

111616       None.

**111617 RATIONALE**

111618       A new algorithm was added at the request of the international community to parallel work in RFC 2045 (MIME). As with the historical *uuencode* format, the Base64 Content-Transfer-Encoding is designed to represent arbitrary sequences of octets in a form that is not humanly readable. A 65-character subset of the ISO/IEC 646:1991 standard is used, enabling 6 bits to be represented per printable character. (The extra 65th character, '=', is used to signify a special processing function.)

111624       This subset has the important property that it is represented identically in all versions of the ISO/IEC 646:1991 standard, including US ASCII, and all characters in the subset are also represented identically in all versions of EBCDIC. The historical *uuencode* algorithm does not share this property, which is the reason that a second algorithm was added to the ISO POSIX-2 standard.

111629       The string "====" was used for the termination instead of the end used in the original format because the latter is a string that could be valid encoded input.

111631       In an early draft, the *-m* option was named *-b* (for Base64), but it was renamed to reflect its relationship to the RFC 2045. A *-u* was also present to invoke the default algorithm, but since this was not historical practice, it was omitted as being unnecessary.

111634 See the RATIONALE section in *uuencode* for the derivation of the **/dev/stdout** symbol.

111635 **FUTURE DIRECTIONS**

111636 None.

111637 **SEE ALSO**

111638 *chmod*, *mailx*, *uuencode*

111639 XBD [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 215)

111640 **CHANGE HISTORY**

111641 First released in Issue 4.

111642 **Issue 6**

111643 This utility is marked as part of the User Portability Utilities option.

111644 The Base64 algorithm and the ability to output to **/dev/stdout** are added as specified in the  
111645 IEEE P1003.2b draft standard.

111646 **Issue 7**

111647 The *uuencode* utility is moved from the User Portability Utilities option to the Base. User  
111648 Portability Utilities is now an option for interactive utilities.

111649 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

111650 **NAME**

111651 uustat — uucp status enquiry and job control

111652 **SYNOPSIS**111653 UU uustat [-q|-k *jobid*|-r *jobid*]111654 uustat [-s *system*] [-u *user*]111655 **DESCRIPTION**

111656 The *uustat* utility shall display the status of, or cancel, previously specified *uucp* requests, or  
 111657 provide general status on *uucp* connections to other systems.

111658 When no options are given, *uustat* shall write to standard output the status of all *uucp* requests  
 111659 issued by the current user.

111660 Typical implementations of this utility require a communications line configured to use XBD  
 111661 [Chapter 11](#) (on page 199), but other communications means may be used. On systems where  
 111662 there are no available communications means (either temporarily or permanently), this utility  
 111663 shall write an error message describing the problem and exit with a non-zero exit status.

111664 **OPTIONS**111665 The *uustat* utility shall conform to XBD [Section 12.2](#) (on page 215).

111666 The following options shall be supported:

111667 -q Write the jobs queued for each machine.

111668 -k *jobid* Kill the *uucp* request whose job identification is *jobid*. The application shall ensure  
 111669 that the killed *uucp* request belongs to the person invoking *uustat* unless that user  
 111670 has appropriate privileges.

111671 -r *jobid* Rejuvenate *jobid*. The files associated with *jobid* are touched so that their  
 111672 modification time is set to the current time. This prevents the cleanup program  
 111673 from deleting the job until the jobs modification time reaches the limit imposed by  
 111674 the program.

111675 -s *system* Write the status of all *uucp* requests for remote system *system*.111676 -u *user* Write the status of all *uucp* requests issued by *user*.111677 **OPERANDS**

111678 None.

111679 **STDIN**

111680 Not used.

111681 **INPUT FILES**

111682 None.

111683 **ENVIRONMENT VARIABLES**111684 The following environment variables shall affect the execution of *uustat*:

111685 *LANG* Provide a default value for the internationalization variables that are unset or null.  
 111686 (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 111687 variables used to determine the values of locale categories.)

111688 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
 111689 internationalization variables.

111690 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
111691 characters (for example, single-byte as opposed to multi-byte characters in  
111692 arguments).

111693 *LC\_MESSAGES*  
111694 Determine the locale that should be used to affect the format and contents of  
111695 diagnostic messages written to standard error, and informative messages written  
111696 to standard output.

111697 *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

111698 **ASYNCHRONOUS EVENTS**  
111699 Default.

111700 **STDOUT**  
111701 The standard output shall consist of information about each job selected, in an unspecified  
111702 format. The information shall include at least the job ID, the user ID or name, and the remote  
111703 system name.

111704 **STDERR**  
111705 The standard error shall be used only for diagnostic messages.

111706 **OUTPUT FILES**  
111707 None.

111708 **EXTENDED DESCRIPTION**  
111709 None.

111710 **EXIT STATUS**  
111711 The following exit values shall be returned:  
111712 0 Successful completion.  
111713 >0 An error occurred.

111714 **CONSEQUENCES OF ERRORS**  
111715 Default.

111716 **APPLICATION USAGE**  
111717 This utility is part of the UUCP Utilities option and need not be supported by all  
111718 implementations.

111719 **EXAMPLES**  
111720 None.

111721 **RATIONALE**  
111722 None.

111723 **FUTURE DIRECTIONS**  
111724 None.

111725 **SEE ALSO**  
111726 *uucp*  
111727 XBD [Chapter 8](#) (on page 173), [Chapter 11](#) (on page 199), [Section 12.2](#) (on page 215)

111728 **CHANGE HISTORY**  
111729 First released in Issue 2.

111730 **Issue 6**

111731 The normative text is reworded to avoid use of the term “must” for application requirements.

111732 The *LC\_TIME* and *TZ* entries are removed from the ENVIRONMENT VARIABLES section.

111733 The UN margin code and associated shading are removed from the *-q* option in response to The  
111734 Open Group Base Resolution bwg2001-003.

111735 **Issue 7**

111736 SD5-XCU-ERN-46 is applied, moving this utility to the UUCP Utilities Option Group.

111737 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

111738 **NAME**

111739       uux — remote command execution

111740 **SYNOPSIS**111741 UU       uux [-jnp] *command-string*111742 **DESCRIPTION**

111743       The *uux* utility shall gather zero or more files from various systems, execute a shell pipeline (see  
 111744       [Section 2.9](#), on page 2338) on a specified system, and then send the standard output of the  
 111745       command to a file on a specified system. Only the first command of a pipeline can have a *system-*  
 111746       *name!* prefix. All other commands in the pipeline shall be executed on the system of the first  
 111747       command.

111748       The following restrictions are applicable to the shell pipeline processed by *uux*:

- 111749       • In gathering files from different systems, pathname expansion shall not be performed by  
 111750       *uux*. Thus, a request such as:

111751       uux "c99 remsys!~/\*.c"

111752       would attempt to copy the file named literally \*.c to the local system.

- 111753       • The redirection operators ">>", "<<", ">|", and ">&" shall not be accepted. Any use of  
 111754       these redirection operators shall cause this utility to write an error message describing the  
 111755       problem and exit with a non-zero exit status.
- 111756       • The reserved word ! cannot be used at the head of the pipeline to modify the exit status.  
 111757       (See the *command-string* operand description below.)
- 111758       • Alias substitution shall not be performed.

111759       A filename can be specified as for *uucp*; it can be an absolute pathname, a pathname preceded by  
 111760       ~*name* (which is replaced by the corresponding login directory), a pathname specified as ~/*dest*  
 111761       (*dest* is prefixed by the public directory called *PUBDIR*; the actual location of *PUBDIR* is  
 111762       implementation-defined), or a simple filename (which is prefixed by *uux* with the current  
 111763       directory). See *uucp* for the details.

111764       The execution of commands on remote systems shall take place in an execution directory known  
 111765       to the *uucp* system. All files required for the execution shall be put into this directory unless they  
 111766       already reside on that machine. Therefore, the application shall ensure that non-local filenames  
 111767       (without path or machine reference) are unique within the *uux* request.

111768       The *uux* utility shall attempt to get all files to the execution system. For files that are output files,  
 111769       the application shall ensure that the filename is escaped using parentheses.

111770       The remote system shall notify the user by mail if the requested command on the remote system  
 111771       was disallowed or the files were not accessible. This notification can be turned off by the -n  
 111772       option.

111773       Typical implementations of this utility require a communications line configured to use XBD  
 111774       [Chapter 11](#) (on page 199), but other communications means may be used. On systems where  
 111775       there are no available communications means (either temporarily or permanently), this utility  
 111776       shall write an error message describing the problem and exit with a non-zero exit status.

111777       The *uux* utility cannot guarantee support for all character encodings in all circumstances. For  
 111778       example, transmission data may be restricted to 7 bits by the underlying network, 8-bit data and  
 111779       filenames need not be portable to non-internationalized systems, and so on. Under these  
 111780       circumstances, it is recommended that only characters defined in the ISO/IEC 646:1991  
 111781       standard International Reference Version (equivalent to ASCII) 7-bit range of characters be used

111782 and that only characters defined in the portable filename character set be used for naming files.

#### 111783 OPTIONS

111784 The *uux* utility shall conform to XBD [Section 12.2](#) (on page 215).

111785 The following options shall be supported:

111786 **-j** Write the job identification string to standard output. This job identification can be  
111787 used by *uustat* to obtain the status or terminate a job.

111788 **-n** Do not notify the user if the command fails.

111789 **-p** Make the standard input to *uux* the standard input to the *command-string*.

#### 111790 OPERANDS

111791 The following operand shall be supported:

111792 *command-string*

111793 A string made up of one or more arguments that are similar to normal command  
111794 arguments, except that the command and any filenames can be prefixed by *system-*  
111795 *name!*. A null *system-name* shall be interpreted as the local system.

#### 111796 STDIN

111797 The standard input shall not be used unless the **'-'** or **-p** option is specified; in those cases, the  
111798 standard input shall be made the standard input of the *command-string*.

#### 111799 INPUT FILES

111800 Input files shall be selected according to the contents of *command-string*.

#### 111801 ENVIRONMENT VARIABLES

111802 The following environment variables shall affect the execution of *uux*:

111803 **LANG** Provide a default value for the internationalization variables that are unset or null.  
111804 (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
111805 variables used to determine the values of locale categories.)

111806 **LC\_ALL** If set to a non-empty string value, override the values of all the other  
111807 internationalization variables.

111808 **LC\_CTYPE** Determine the locale for the interpretation of sequences of bytes of text data as  
111809 characters (for example, single-byte as opposed to multi-byte characters in  
111810 arguments).

111811 **LC\_MESSAGES**

111812 Determine the locale that should be used to affect the format and contents of  
111813 diagnostic messages written to standard error.

111814 **NLSPATH** Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

#### 111815 ASYNCHRONOUS EVENTS

111816 Default.

#### 111817 STDOUT

111818 The standard output shall not be used unless the **-j** option is specified; in that case, the job  
111819 identification string shall be written to standard output in the following format:

111820 "%s\n", <*jobid*>



111821 **STDERR**

111822       The standard error shall be used only for diagnostic messages.

111823 **OUTPUT FILES**

111824       Output files shall be created or written, or both, according to the contents of *command-string*.

111825       If **-n** is not used, mail files shall be modified following any command or file-access failures on  
111826       the remote system.

111827 **EXTENDED DESCRIPTION**

111828       None.

111829 **EXIT STATUS**

111830       The following exit values shall be returned:

111831       0   Successful completion.

111832       >0  An error occurred.

111833 **CONSEQUENCES OF ERRORS**

111834       Default.

111835 **APPLICATION USAGE**

111836       This utility is part of the UUCP Utilities option and need not be supported by all  
111837       implementations.

111838       Note that, for security reasons, many installations limit the list of commands executable on  
111839       behalf of an incoming request from *uux*. Many sites permit little more than the receipt of mail  
111840       via *uux*.

111841       Any characters special to the command interpreter should be quoted either by quoting the entire  
111842       *command-string* or quoting the special characters as individual arguments.

111843       As noted in *uucp*, shell pattern matching notation characters appearing in pathnames are  
111844       expanded on the appropriate local system. This is done under the control of local settings of  
111845       *LC\_COLLATE* and *LC\_CTYPE*. Thus, care should be taken when using bracketed filename  
111846       patterns, as collation and typing rules may vary from one system to another. Also be aware that  
111847       certain types of expression (that is, equivalence classes, character classes, and collating symbols)  
111848       need not be supported on non-internationalized systems.

111849 **EXAMPLES**

111850       1. The following command gets **file1** from system **a** and **file2** from system **b**, executes *diff* on  
111851       the local system, and puts the results in **file.diff** in the local *PUBDIR* directory. (*PUBDIR*  
111852       is the *uucp* public directory on the local system.)

111853       uux "!diff a!/usr/file1 b!/a4/file2 >!~/file.diff"

111854       2. The following command fails because *uux* places all files copied to a system in the same  
111855       working directory. Although the files **xyz** are from two different systems, their filenames  
111856       are the same and conflict.

111857       uux "!diff a!/usr1/xyz b!/usr2/xyz >!~/xyz.diff"

111858       3. The following command succeeds (assuming *diff* is permitted on system **a**) because the  
111859       file local to system **a** is not copied to the working directory, and hence does not conflict  
111860       with the file from system **c**.

111861       uux "a!diff a!/usr/xyz c!/usr/xyz >!~/xyz.diff"

111862 **RATIONALE**

111863 None.

111864 **FUTURE DIRECTIONS**

111865 None.

111866 **SEE ALSO**111867 [Chapter 2](#) (on page 2319), [uucp](#), [uuencode](#), [uustat](#)111868 XBD [Chapter 8](#) (on page 173), [Chapter 11](#) (on page 199), [Section 12.2](#) (on page 215)111869 **CHANGE HISTORY**

111870 First released in Issue 2.

111871 **Issue 6**

111872 The obsolescent SYNOPSIS is removed.

111873 The normative text is reworded to avoid use of the term “must” for application requirements.

111874 The UN margin code and associated shading are removed from the `-j` option in response to The  
111875 Open Group Base Resolution bwg2001-003.111876 **Issue 7**

111877 SD5-XCU-ERN-46 is applied, moving this utility to the UUCP Utilities Option Group.

111878 **NAME**111879       val — validate SCCS files (**DEVELOPMENT**)111880 **SYNOPSIS**

```
111881 XSI val -
111882 val [-s] [-m name] [-r SID] [-y type] file...
```

111883 **DESCRIPTION**

111884       The *val* utility shall determine whether the specified *file* is an SCCS file meeting the  
 111885       characteristics specified by the options.

111886 **OPTIONS**

111887       The *val* utility shall conform to XBD [Section 12.2](#) (on page 215), except that the usage of the ‘-’  
 111888       operand is not strictly as intended by the guidelines (that is, reading options and operands from  
 111889       standard input).

111890       The following options shall be supported:

- 111891       **-m name**     Specify a *name*, which is compared with the SCCS %M% keyword in *file*; see [get](#).
- 111892       **-r SID**       Specify a *SID* (SCCS Identification String), an SCCS delta number. A check shall be  
 111893       made to determine whether the *SID* is ambiguous (for example, **-r 1** is ambiguous  
 111894       because it physically does not exist but implies 1.1, 1.2, and so on, which may exist)  
 111895       or invalid (for example, **-r 1.0** or **-r 1.1.0** are invalid because neither case can exist  
 111896       as a valid delta number). If the *SID* is valid and not ambiguous, a check shall be  
 111897       made to determine whether it actually exists.
- 111898       **-s**           Silence the diagnostic message normally written to standard output for any error  
 111899       that is detected while processing each named file on a given command line.
- 111900       **-y type**     Specify a *type*, which shall be compared with the SCCS %Y% keyword in *file*; see  
 111901       [get](#).

111902 **OPERANDS**

111903       The following operands shall be supported:

- 111904       *file*        A pathname of an existing SCCS file. If exactly one *file* operand appears, and it is  
 111905       ‘-’, the standard input shall be read: each line shall be independently processed  
 111906       as if it were a command line argument list. (However, the line is not subjected to  
 111907       any of the shell word expansions, such as parameter expansion or quote removal.)

111908 **STDIN**

111909       The standard input shall be a text file used only when the *file* operand is specified as ‘-’.

111910 **INPUT FILES**

111911       Any SCCS files processed shall be files of an unspecified format.

111912 **ENVIRONMENT VARIABLES**

111913       The following environment variables shall affect the execution of *val*:

- 111914       **LANG**       Provide a default value for the internationalization variables that are unset or null.  
 111915       (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 111916       variables used to determine the values of locale categories.)
- 111917       **LC\_ALL**     If set to a non-empty string value, override the values of all the other  
 111918       internationalization variables.

111919 **LC\_CTYPE** Determine the locale for the interpretation of sequences of bytes of text data as  
 111920 characters (for example, single-byte as opposed to multi-byte characters in  
 111921 arguments and input files).

111922 **LC\_MESSAGES**  
 111923 Determine the locale that should be used to affect the format and contents of  
 111924 diagnostic messages written to standard error, and informative messages written  
 111925 to standard output.

111926 **NLSPATH** Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

## 111927 **ASYNCHRONOUS EVENTS**

111928 Default.

## 111929 **STDOUT**

111930 The standard output shall consist of informative messages about either:

- 111931 1. Each file processed
- 111932 2. Each command line read from standard input

111933 If the standard input is not used, for each *file* operand yielding a discrepancy, the output line  
 111934 shall have the following format:

111935 "%s: %s\n", <pathname>, <unspecified string>

111936 If the standard input is used, for each input line yielding a discrepancy, the output shall have the  
 111937 following format:

111938 "%s\n\n %s: %s\n", <input>, <pathname>, <unspecified string>

111939 where <input> is the input line minus its terminating <newline>.

## 111940 **STDERR**

111941 Not used.

## 111942 **OUTPUT FILES**

111943 None.

## 111944 **EXTENDED DESCRIPTION**

111945 None.

## 111946 **EXIT STATUS**

111947 The 8-bit code returned by *val* shall be a disjunction of the possible errors; that is, it can be  
 111948 interpreted as a bit string where set bits are interpreted as follows:

|        |      |   |                                     |
|--------|------|---|-------------------------------------|
| 111949 | 0x80 | = | Missing file argument.              |
| 111950 | 0x40 | = | Unknown or duplicate option.        |
| 111951 | 0x20 | = | Corrupted SCCS file.                |
| 111952 | 0x10 | = | Cannot open file or file not SCCS.  |
| 111953 | 0x08 | = | <i>SID</i> is invalid or ambiguous. |
| 111954 | 0x04 | = | <i>SID</i> does not exist.          |
| 111955 | 0x02 | = | %Y%, -y mismatch.                   |
| 111956 | 0x01 | = | %M%, -m mismatch.                   |

111957 Note that *val* can process two or more files on a given command line and can process multiple  
 111958 command lines (when reading the standard input). In these cases an aggregate code shall be  
 111959 returned: a logical OR of the codes generated for each command line and file processed.

111960 **CONSEQUENCES OF ERRORS**

111961 Default.

111962 **APPLICATION USAGE**

111963 Since the *val* exit status sets the 0x80 bit, shell applications checking "\$?" cannot tell if it  
 111964 terminated due to a missing file argument or receipt of a signal.

111965 **EXAMPLES**

111966 In a directory with three SCCS files—*s.x* (of *t* type “text”), *s.y*, and *s.z* (a corrupted file)—the  
 111967 following command could produce the output shown:

```
111968 val - <<EOF
111969 -y source s.x
111970 -m y s.y
111971 s.z
111972 EOF
111973 -y source s.x
111974 s.x: %Y%, -y mismatch
111975 s.z
111976 s.z: corrupted SCCS file
```

111977 **RATIONALE**

111978 None.

111979 **FUTURE DIRECTIONS**

111980 None.

111981 **SEE ALSO**111982 *admin, delta, get, prs*111983 XBD [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 215)111984 **CHANGE HISTORY**

111985 First released in Issue 2.

111986 **Issue 6**

111987 The Open Group Corrigendum U025/4 is applied, correcting a typographical error in the EXIT  
 111988 STATUS.

111989 **Issue 7**

111990 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

111991 POSIX.1-2008, Technical Corrigendum 1, XCU/TC1-2008/0147 [416] and XCU/TC1-2008/0148 +  
 111992 [416] are applied.

111993 **NAME**

111994 vi — screen-oriented (visual) display editor

111995 **SYNOPSIS**111996 UP `vi [-rR] [-c command] [-t tagstring] [-w size] [file...]`111997 **DESCRIPTION**111998 This utility shall be provided on systems that both support the User Portability Utilities option  
111999 and define the POSIX2\_CHAR\_TERM symbol. On other systems it is optional.112000 The *vi* (visual) utility is a screen-oriented text editor. Only the open and visual modes of the  
112001 editor are described in POSIX.1-2008; see the line editor *ex* for additional editing capabilities  
112002 used in *vi*. The user can switch back and forth between *vi* and *ex* and execute *ex* commands from  
112003 within *vi*.112004 This reference page uses the term *edit buffer* to describe the current working text. No specific  
112005 implementation is implied by this term. All editing changes are performed on the edit buffer,  
112006 and no changes to it shall affect any file until an editor command writes the file.112007 When using *vi*, the terminal screen acts as a window into the editing buffer. Changes made to  
112008 the editing buffer shall be reflected in the screen display; the position of the cursor on the screen  
112009 shall indicate the position within the editing buffer.112010 Certain terminals do not have all the capabilities necessary to support the complete *vi* definition.  
112011 When these commands cannot be supported on such terminals, this condition shall not produce  
112012 an error message such as “not an editor command” or report a syntax error. The implementation  
112013 may either accept the commands and produce results on the screen that are the result of an  
112014 unsuccessful attempt to meet the requirements of this volume of POSIX.1-2008 or report an error  
112015 describing the terminal-related deficiency.112016 **OPTIONS**112017 The *vi* utility shall conform to XBD [Section 12.2](#) (on page 215), except that ‘+’ may be  
112018 recognized as an option delimiter as well as ‘-’.

112019 The following options shall be supported:

112020 **-c *command*** See the *ex* command description of the **-c** option.112021 **-r** See the *ex* command description of the **-r** option.112022 **-R** See the *ex* command description of the **-R** option.112023 **-t *tagstring*** See the *ex* command description of the **-t** option.112024 **-w *size*** See the *ex* command description of the **-w** option.112025 **OPERANDS**112026 See the OPERANDS section of the *ex* command for a description of the operands supported by  
112027 the *vi* command.112028 **STDIN**112029 If standard input is not a terminal device, the results are undefined. The standard input consists  
112030 of a series of commands and input text, as described in the EXTENDED DESCRIPTION section.112031 If a read from the standard input returns an error, or if the editor detects an end-of-file condition  
112032 from the standard input, it shall be equivalent to a SIGHUP asynchronous event.

112033 **INPUT FILES**

112034 See the INPUT FILES section of the *ex* command for a description of the input files supported by  
 112035 the *vi* command.

112036 **ENVIRONMENT VARIABLES**

112037 See the ENVIRONMENT VARIABLES section of the *ex* command for the environment variables  
 112038 that affect the execution of the *vi* command.

112039 **ASYNCHRONOUS EVENTS**

112040 See the ASYNCHRONOUS EVENTS section of the *ex* for the asynchronous events that affect the  
 112041 execution of the *vi* command.

112042 **STDOUT**

112043 If standard output is not a terminal device, undefined results occur.

112044 Standard output may be used for writing prompts to the user, for informational messages, and  
 112045 for writing lines from the file.

112046 **STDERR**

112047 If standard output is not a terminal device, undefined results occur.

112048 The standard error shall be used only for diagnostic messages.

112049 **OUTPUT FILES**

112050 See the OUTPUT FILES section of the *ex* command for a description of the output files  
 112051 supported by the *vi* command.

112052 **EXTENDED DESCRIPTION**

112053 If the terminal does not have the capabilities necessary to support an unspecified portion of the  
 112054 *vi* definition, implementations shall start initially in *ex* mode or open mode. Otherwise, after  
 112055 initialization, *vi* shall be in command mode; text input mode can be entered by one of several  
 112056 commands used to insert or change text. In text input mode, <ESC> can be used to return to  
 112057 command mode; other uses of <ESC> are described later in this section; see [Terminate](#)  
 112058 [Command or Input Mode](#) (on page 3346).

112059 **Initialization in *ex* and *vi***

112060 See [Initialization in \*ex\* and \*vi\*](#) (on page 2666) for a description of *ex* and *vi* initialization for the *vi*  
 112061 utility.

112062 **Command Descriptions in *vi***

112063 The following symbols are used in this reference page to represent arguments to commands.

112064 *buffer* See the description of *buffer* in the EXTENDED DESCRIPTION section of the *ex* utility;  
 112065 see [Command Descriptions in \*ex\*](#) (on page 2675).

112066 In open and visual mode, when a command synopsis shows both [*buffer*] and [*count*]  
 112067 preceding the command name, they can be specified in either order.

112068 *count* A positive integer used as an optional argument to most commands, either to give a  
 112069 repeat count or as a size. This argument is optional and shall default to 1 unless  
 112070 otherwise specified.

112071 The Synopsis lines for the *vi* commands <control>-G, <control>-L, <control>-R,  
 112072 <control>-], %, &, ^, D, m, M, Q, u, U, and ZZ do not have *count* as an optional  
 112073 argument. Regardless, it shall not be an error to specify a *count* to these commands, and  
 112074 any specified *count* shall be ignored.

*motion* An optional trailing argument used by the **!**, **<**, **>**, **c**, **d**, and **y** commands, which is used to indicate the region of text that shall be affected by the command. The motion can be either one of the command characters repeated or one of several other *vi* commands (listed in the following table). Each of the applicable commands specifies the region of text matched by repeating the command; each command that can be used as a motion command specifies the region of text it affects.

Commands that take *motion* arguments operate on either lines or characters, depending on the circumstances. When operating on lines, all lines that fall partially or wholly within the text region specified for the command shall be affected. When operating on characters, only the exact characters in the specified text region shall be affected. Each motion command specifies this individually.

When commands that may be motion commands are not used as motion commands, they shall set the current position to the current line and column as specified.

The following commands shall be valid cursor motion commands:

|                   |   |    |   |   |
|-------------------|---|----|---|---|
| <apostrophe>      | ( | -  | j | H |
| <carriage-return> | ) | \$ | k | L |
| <comma>           | [ | %  | l | M |
| <control>-H       | ] | _  | n | N |
| <control>-N       | { | ;  | t | T |
| <control>-P       | } | ?  | w | W |
| <grave-accent>    | ^ | b  | B |   |
| <newline>         | + | e  | E |   |
| <space>           |   | f  | F |   |
| <zero>            | / | h  | G |   |

Any *count* that is specified to a command that has an associated motion command shall be applied to the motion command. If a *count* is applied to both the command and its associated motion command, the effect shall be multiplicative.

The following symbols are used in this section to specify locations in the edit buffer:

*current character*

The character that is currently indicated by the cursor.

*end of a line*

The point located between the last non-<newline> (if any) and the terminating <newline> of a line. For an empty line, this location coincides with the beginning of the line.

*end of the edit buffer*

The location corresponding to the end of the last line in the edit buffer.

The following symbols are used in this section to specify command actions:

*bigword* In the POSIX locale, *vi* shall recognize four kinds of *bigwords*:

1. A maximal sequence of non-<blank> characters preceded and followed by <blank> characters or the beginning or end of a line or the edit buffer
2. One or more sequential blank lines
3. The first character in the edit buffer



- 112117 4. The last non-<newline> in the edit buffer
- 112118 *word* In the POSIX locale, *vi* shall recognize five kinds of words:
- 112119 1. A maximal sequence of letters, digits, and underscores, delimited at both ends
- 112120 by:
- 112121 — Characters other than letters, digits, or underscores
- 112122 — The beginning or end of a line
- 112123 — The beginning or end of the edit buffer
- 112124 2. A maximal sequence of characters other than letters, digits, underscores, or
- 112125 <blank> characters, delimited at both ends by:
- 112126 — A letter, digit, underscore
- 112127 — <blank> characters
- 112128 — The beginning or end of a line
- 112129 — The beginning or end of the edit buffer
- 112130 3. One or more sequential blank lines
- 112131 4. The first character in the edit buffer
- 112132 5. The last non-<newline> in the edit buffer
- 112133 *section boundary*
- 112134 A *section boundary* is one of the following:
- 112135 1. A line whose first character is a <form-feed>
- 112136 2. A line whose first character is an open curly brace ( ' { ' )
- 112137 3. A line whose first character is a <period> and whose second and third characters
- 112138 match a two-character pair in the **sections** edit option (see *ed*)
- 112139 4. A line whose first character is a <period> and whose only other character
- 112140 matches the first character of a two-character pair in the **sections** edit option,
- 112141 where the second character of the two-character pair is a <space>
- 112142 5. The first line of the edit buffer
- 112143 6. The last line of the edit buffer if the last line of the edit buffer is empty or if it is a
- 112144 ]] or } command; otherwise, the last non-<newline> of the last line of the edit
- 112145 buffer
- 112146 *paragraph boundary*
- 112147 A *paragraph boundary* is one of the following:
- 112148 1. A section boundary
- 112149 2. A line whose first character is a <period> and whose second and third characters
- 112150 match a two-character pair in the **paragraphs** edit option (see *ed*)
- 112151 3. A line whose first character is a <period> and whose only other character
- 112152 matches the first character of a two-character pair in the *paragraphs* edit option,
- 112153 where the second character of the two-character pair is a <space>

#### 4. One or more sequential blank lines

*remembered search direction*

See the description of *remembered search direction* in *ed*.

*sentence boundary*

A *sentence boundary* is one of the following:

1. A paragraph boundary
2. The first non-`<blank>` that occurs after a paragraph boundary
3. The first non-`<blank>` that occurs after a `<period>` (`'.'`), `<exclamation-mark>` (`'!'`), or `<question-mark>` (`'?'`), followed by two `<space>` characters or the end of a line; any number of closing parenthesis (`' ) '`), closing brackets (`' ] '`), double-quote (`' "'`), or single-quote (`<apostrophe>`) characters can appear between the punctuation mark and the two `<space>` characters or end-of-line

In the remainder of the description of the *vi* utility, the term “buffer line” refers to a line in the edit buffer and the term “display line” refers to the line or lines on the display screen used to display one buffer line. The term “current line” refers to a specific “buffer line”.

If there are display lines on the screen for which there are no corresponding buffer lines because they correspond to lines that would be after the end of the file, they shall be displayed as a single `<tilde>` (`'~'`) character, plus the terminating `<newline>`.

The last line of the screen shall be used to report errors or display informational messages. It shall also be used to display the input for “line-oriented commands” (`/`, `?`, `:`, and `!`). When a line-oriented command is executed, the editor shall enter text input mode on the last line on the screen, using the respective command characters as prompt characters. (In the case of the `!` command, the associated motion shall be entered by the user before the editor enters text input mode.) The line entered by the user shall be terminated by a `<newline>`, a non-`<control>`-V-escaped `<carriage-return>`, or unescaped `<ESC>`. It is unspecified if more characters than require a display width minus one column number of screen columns can be entered.

If any command is executed that overwrites a portion of the screen other than the last line of the screen (for example, the *ex* **suspend** or `!` commands), other than the *ex* **shell** command, the user shall be prompted for a character before the screen is refreshed and the edit session continued.

`<tab>` characters shall take up the number of columns on the screen set by the **tabstop** edit option (see *ed*), unless there are less than that number of columns before the display margin that will cause the displayed line to be folded; in this case, they shall only take up the number of columns up to that boundary.

The cursor shall be placed on the current line and relative to the current column as specified by each command described in the following sections.

In open mode, if the current line is not already displayed, then it shall be displayed.

In visual mode, if the current line is not displayed, then the lines that are displayed shall be expanded, scrolled, or redrawn to cause an unspecified portion of the current line to be displayed. If the screen is redrawn, no more than the number of display lines specified by the value of the **window** edit option shall be displayed (unless the current line cannot be completely displayed in the number of display lines specified by the **window** edit option) and the current line shall be positioned as close to the center of the displayed lines as possible (within the constraints imposed by the distance of the line from the beginning or end of the edit buffer). If the current line is before the first line in the display and the screen is scrolled, an unspecified

112199 portion of the current line shall be placed on the first line of the display. If the current line is after  
 112200 the last line in the display and the screen is scrolled, an unspecified portion of the current line  
 112201 shall be placed on the last line of the display.

112202 In visual mode, if a line from the edit buffer (other than the current line) does not entirely fit into  
 112203 the lines at the bottom of the display that are available for its presentation, the editor may choose  
 112204 not to display any portion of the line. The lines of the display that do not contain text from the  
 112205 edit buffer for this reason shall each consist of a single ' @ ' character.

112206 In visual mode, the editor may choose for unspecified reasons to not update lines in the display  
 112207 to correspond to the underlying edit buffer text. The lines of the display that do not correctly  
 112208 correspond to text from the edit buffer for this reason shall consist of a single ' @ ' character (plus  
 112209 the terminating <newline>), and the <control>-R command shall cause the editor to update the  
 112210 screen to correctly represent the edit buffer.

112211 Open and visual mode commands that set the current column set it to a column position in the  
 112212 display, and not a character position in the line. In this case, however, the column position in the  
 112213 display shall be calculated for an infinite width display; for example, the column related to a  
 112214 character that is part of a line that has been folded onto additional screen lines will be offset from  
 112215 the display line column where the buffer line begins, not from the beginning of a particular  
 112216 display line.

112217 The display cursor column in the display is based on the value of the current column, as follows,  
 112218 with each rule applied in turn:

- 112219 1. If the current column is after the last display line column used by the displayed line, the  
 112220 display cursor column shall be set to the last display line column occupied by the last  
 112221 non-<newline> in the current line; otherwise, the display cursor column shall be set to the  
 112222 current column.
- 112223 2. If the character of which some portion is displayed in the display line column specified by  
 112224 the display cursor column requires more than a single display line column:
  - 112225 a. If in text input mode, the display cursor column shall be adjusted to the first  
 112226 display line column in which any portion of that character is displayed.
  - 112227 b. Otherwise, the display cursor column shall be adjusted to the last display line  
 112228 column in which any portion of that character is displayed.

112229 The current column shall not be changed by these adjustments to the display cursor column.

112230 If an error occurs during the parsing or execution of a *vi* command:

- 112231 • The terminal shall be alerted. Execution of the *vi* command shall stop, and the cursor (for  
 112232 example, the current line and column) shall not be further modified.
- 112233 • Unless otherwise specified by the following command sections, it is unspecified whether  
 112234 an informational message shall be displayed.
- 112235 • Any partially entered *vi* command shall be discarded.
- 112236 • If the *vi* command resulted from a **map** expansion, all characters from that **map** expansion  
 112237 shall be discarded, except as otherwise specified by the **map** command (see *ed*).
- 112238 • If the *vi* command resulted from the execution of a buffer, no further commands caused by  
 112239 the execution of the buffer shall be executed.

## Page Backwards

**Synopsis:**     [*count*] <control>-B

If in open mode, the <control>-B command shall behave identically to the **z** command. Otherwise, if the current line is the first line of the edit buffer, it shall be an error.

If the **window** edit option is less than 3, display a screen where the last line of the display shall be some portion of:

(*current first line*) -1

otherwise, display a screen where the first line of the display shall be some portion of:

(*current first line*) - *count* x ((*window edit option*) -2)

If this calculation would result in a line that is before the first line of the edit buffer, the first line of the display shall display some portion of the first line of the edit buffer.

*Current line:* If no lines from the previous display remain on the screen, set to the last line of the display; otherwise, set to (*line* - the number of new lines displayed on this screen).

*Current column:* Set to non-<blank>.

## Scroll Forward

**Synopsis:**     [*count*] <control>-D

If the current line is the last line of the edit buffer, it shall be an error.

If no *count* is specified, *count* shall default to the *count* associated with the previous <control>-D or <control>-U command. If there was no previous <control>-D or <control>-U command, *count* shall default to the value of the **scroll** edit option.

If in open mode, write lines starting with the line after the current line, until *count* lines or the last line of the file have been written.

*Current line:* If the current line + *count* is past the last line of the edit buffer, set to the last line of the edit buffer; otherwise, set to the current line + *count*.

*Current column:* Set to non-<blank>.

## Scroll Forward by Line

**Synopsis:**     [*count*] <control>-E

Display the line *count* lines after the last line currently displayed.

If the last line of the edit buffer is displayed, it shall be an error. If there is no line *count* lines after the last line currently displayed, the last line of the display shall display some portion of the last line of the edit buffer.

*Current line:* Unchanged if the previous current character is displayed; otherwise, set to the first line displayed.

*Current column:* Unchanged.

**Page Forward**

*Synopsis:*     [*count*] <control>-F

If in open mode, the <control>-F command shall behave identically to the **z** command. Otherwise, if the current line is the last line of the edit buffer, it shall be an error.

If the **window** edit option is less than 3, display a screen where the first line of the display shall be some portion of:

*(current last line) +1*

otherwise, display a screen where the first line of the display shall be some portion of:

*(current first line) + count x ((window edit option) -2)*

If this calculation would result in a line that is after the last line of the edit buffer, the last line of the display shall display some portion of the last line of the edit buffer.

*Current line:* If no lines from the previous display remain on the screen, set to the first line of the display; otherwise, set to (*line* + the number of new lines displayed on this screen).

*Current column:* Set to non-<blank>.

**Display Information**

*Synopsis:*     <control>-G

This command shall be equivalent to the **ex file** command.

**Move Cursor Backwards**

*Synopsis:*     [*count*] <control>-H

[*count*] h

the current erase character (see **stty**)

If there are no characters before the current character on the current line, it shall be an error. If there are less than *count* previous characters on the current line, *count* shall be adjusted to the number of previous characters on the line.

If used as a motion command:

1. The text region shall be from the character before the starting cursor up to and including the *count*th character before the starting cursor.
2. Any text copied to a buffer shall be in character mode.

If not used as a motion command:

*Current line:* Unchanged.

*Current column:* Set to (*column* – the number of columns occupied by *count* characters ending with the previous current column).

**Move Down**

*Synopsis:*     [*count*] <newline>  
                  [*count*] <control>-J  
                  [*count*] <control>-M  
                  [*count*] <control>-N  
                  [*count*] j  
                  [*count*] <carriage-return>  
                  [*count*] +

If there are less than *count* lines after the current line in the edit buffer, it shall be an error.

If used as a motion command:

1. The text region shall include the starting line and the next *count* – 1 lines.
2. Any text copied to a buffer shall be in line mode.

If not used as a motion command:

*Current line:* Set to *current line* + *count*.

*Current column:* Set to non-<blank> for the <carriage-return>, <control>-M, and + commands; otherwise, unchanged.

**Clear and Redisplay**

*Synopsis:*     <control>-L

If in open mode, clear the screen and redisplay the current line. Otherwise, clear and redisplay the screen.

*Current line:* Unchanged.

*Current column:* Unchanged.

**Move Up**

*Synopsis:*     [*count*] <control>-P  
                  [*count*] k  
                  [*count*] –

If there are less than *count* lines before the current line in the edit buffer, it shall be an error.

If used as a motion command:

1. The text region shall include the starting line and the previous *count* lines.
2. Any text copied to a buffer shall be in line mode.

If not used as a motion command:

*Current line:* Set to *current line* – *count*.

*Current column:* Set to non-<blank> for the – command; otherwise, unchanged.

## 112339 **Redraw Screen**

112340 *Synopsis:*      <control>-R

112341 If any lines have been deleted from the display screen and flagged as deleted on the terminal  
112342 using the @ convention (see the beginning of the EXTENDED DESCRIPTION section), they shall  
112343 be redisplayed to match the contents of the edit buffer.

112344 It is unspecified whether lines flagged with @ because they do not fit on the terminal display  
112345 shall be affected.

112346 *Current line:* Unchanged.

112347 *Current column:* Unchanged.

## 112348 **Scroll Backward**

112349 *Synopsis:*      [*count*] <control>-U

112350 If the current line is the first line of the edit buffer, it shall be an error.

112351 If no *count* is specified, *count* shall default to the *count* associated with the previous <control>-D  
112352 or <control>-U command. If there was no previous <control>-D or <control>-U command, *count*  
112353 shall default to the value of the **scroll** edit option.

112354 *Current line:* If *count* is greater than the current line, set to 1; otherwise, set to the current line –  
112355 *count*.

112356 *Current column:* Set to non-<blank>.

## 112357 **Scroll Backward by Line**

112358 *Synopsis:*      [*count*] <control>-Y

112359 Display the line *count* lines before the first line currently displayed.

112360 If the current line is the first line of the edit buffer, it shall be an error. If this calculation would  
112361 result in a line that is before the first line of the edit buffer, the first line of the display shall  
112362 display some portion of the first line of the edit buffer.

112363 *Current line:* Unchanged if the previous current character is displayed; otherwise, set to the first  
112364 line displayed.

112365 *Current column:* Unchanged.

## 112366 **Edit the Alternate File**

112367 *Synopsis:*      <control>-^

112368 This command shall be equivalent to the *ex* **edit** command, with the alternate pathname as its  
112369 argument.

**Terminate Command or Input Mode**

*Synopsis:*      <ESC>

If a partial *vi* command (as defined by at least one, non-*count* character) has been entered, discard the *count* and the command character(s).

Otherwise, if no command characters have been entered, and the <ESC> was the result of a map expansion, the terminal shall be alerted and the <ESC> character shall be discarded, but it shall not be an error.

Otherwise, it shall be an error.

*Current line:* Unchanged.

*Current column:* Unchanged.

**Search for tagstring**

*Synopsis:*      <control>-]

If the current character is not a word or <blank>, it shall be an error.

This command shall be equivalent to the *ex tag* command, with the argument to that command defined as follows.

If the current character is a <blank>:

1. Skip all <blank> characters after the cursor up to the end of the line.
2. If the end of the line is reached, it shall be an error.

Then, the argument to the *ex tag* command shall be the current character and all subsequent characters, up to the first non-word character or the end of the line.

**Move Cursor Forward**

*Synopsis:*      [*count*] <space>  
                 [*count*] 1 (ell)

If there are less than *count* non-<newline> characters after the cursor on the current line, *count* shall be adjusted to the number of non-<newline> characters after the cursor on the line.

If used as a motion command:

1. If the current or *count*th character after the cursor is the last non-<newline> in the line, the text region shall be comprised of the current character up to and including the last non-<newline> in the line. Otherwise, the text region shall be from the current character up to, but not including, the *count*th character after the cursor.
2. Any text copied to a buffer shall be in character mode.

If not used as a motion command:

If there are no non-<newline> characters after the current character on the current line, it shall be an error.

*Current line:* Unchanged.

*Current column:* Set to the last column that displays any portion of the *count*th character after the current character.



**Replace Text with Results from Shell Command**

*Synopsis:*     [*count*] ! *motion shell-commands* <newline>

If the motion command is the ! command repeated:

1. If the edit buffer is empty and no *count* was supplied, the command shall be the equivalent of the *ex :read !* command, with the text input, and no text shall be copied to any buffer.
2. Otherwise:
  - a. If there are less than *count* -1 lines after the current line in the edit buffer, it shall be an error.
  - b. The text region shall be from the current line up to and including the next *count* -1 lines.

Otherwise, the text region shall be the lines in which any character of the text region specified by the motion command appear.

Any text copied to a buffer shall be in line mode.

This command shall be equivalent to the *ex !* command for the specified lines.

**Move Cursor to End-of-Line**

*Synopsis:*     [*count*] \$

It shall be an error if there are less than (*count* -1) lines after the current line in the edit buffer.

If used as a motion command:

1. If *count* is 1:
  - a. It shall be an error if the line is empty.
  - b. Otherwise, the text region shall consist of all characters from the starting cursor to the last non-<newline> in the line, inclusive, and any text copied to a buffer shall be in character mode.
2. Otherwise, if the starting cursor position is at or before the first non-<blank> in the line, the text region shall consist of the current and the next *count* -1 lines, and any text saved to a buffer shall be in line mode.
3. Otherwise, the text region shall consist of all characters from the starting cursor to the last non-<newline> in the line that is *count* -1 lines forward from the current line, and any text copied to a buffer shall be in character mode.

If not used as a motion command:

*Current line:* Set to the *current line* + *count* -1.

*Current column:* The current column is set to the last display line column of the last non-<newline> in the line, or column position 1 if the line is empty.

The current column shall be adjusted to be on the last display line column of the last non-<newline> of the current line as subsequent commands change the current line, until a command changes the current column.

**Move to Matching Character***Synopsis:*     %

If the character at the current position is not a parenthesis, bracket, or curly brace, search forward in the line to the first one of those characters. If no such character is found, it shall be an error.

The matching character shall be the parenthesis, bracket, or curly brace matching the parenthesis, bracket, or curly brace, respectively, that was at the current position or that was found on the current line.

Matching shall be determined as follows, for an open parenthesis:

1. Set a counter to 1.
2. Search forwards until a parenthesis is found or the end of the edit buffer is reached.
3. If the end of the edit buffer is reached, it shall be an error.
4. If an open parenthesis is found, increment the counter by 1.
5. If a close parenthesis is found, decrement the counter by 1.
6. If the counter is zero, the current character is the matching character.

Matching for a close parenthesis shall be equivalent, except that the search shall be backwards, from the starting character to the beginning of the buffer, a close parenthesis shall increment the counter by 1, and an open parenthesis shall decrement the counter by 1.

Matching for brackets and curly braces shall be equivalent, except that searching shall be done for open and close brackets or open and close curly braces. It is implementation-defined whether other characters are searched for and matched as well.

If used as a motion command:

1. If the matching cursor was after the starting cursor in the edit buffer, and the starting cursor position was at or before the first non-`<blank>` non-`<newline>` in the starting line, and the matching cursor position was at or after the last non-`<blank>` non-`<newline>` in the matching line, the text region shall consist of the current line to the matching line, inclusive, and any text copied to a buffer shall be in line mode.
2. If the matching cursor was before the starting cursor in the edit buffer, and the starting cursor position was at or after the last non-`<blank>` non-`<newline>` in the starting line, and the matching cursor position was at or before the first non-`<blank>` non-`<newline>` in the matching line, the text region shall consist of the current line to the matching line, inclusive, and any text copied to a buffer shall be in line mode.
3. Otherwise, the text region shall consist of the starting character to the matching character, inclusive, and any text copied to a buffer shall be in character mode.

If not used as a motion command:

*Current line:* Set to the line where the matching character is located.

*Current column:* Set to the last column where any portion of the matching character is displayed.

**Repeat Substitution**

*Synopsis:*     &

Repeat the previous substitution command. This command shall be equivalent to the *ex &* command with the current line as its addresses, and without *options, count, or flags*.

**Return to Previous Context at Beginning of Line**

*Synopsis:*     ' *character*

It shall be an error if there is no line in the edit buffer marked by *character*.

If used as a motion command:

1. If the starting cursor is after the marked cursor, then the locations of the starting cursor and the marked cursor in the edit buffer shall be logically swapped.
2. The text region shall consist of the starting line up to and including the marked line, and any text copied to a buffer shall be in line mode.

If not used as a motion command:

*Current line:* Set to the line referenced by the mark.

*Current column:* Set to non-<blank>.

**Return to Previous Context**

*Synopsis:*     ` *character*

It shall be an error if the marked line is no longer in the edit buffer. If the marked line no longer contains a character in the saved numbered character position, it shall be as if the marked position is the first non-<blank>.

If used as a motion command:

1. It shall be an error if the marked cursor references the same character in the edit buffer as the starting cursor.
2. If the starting cursor is after the marked cursor, then the locations of the starting cursor and the marked cursor in the edit buffer shall be logically swapped.
3. If the starting line is empty or the starting cursor is at or before the first non-<blank> non-<newline> of the starting line, and the marked cursor line is empty or the marked cursor references the first character of the marked cursor line, the text region shall consist of all lines containing characters from the starting cursor to the line before the marked cursor line, inclusive, and any text copied to a buffer shall be in line mode.
4. Otherwise, if the marked cursor line is empty or the marked cursor references a character at or before the first non-<blank> non-<newline> of the marked cursor line, the region of text shall be from the starting cursor to the last non-<newline> of the line before the marked cursor line, inclusive, and any text copied to a buffer shall be in character mode.
5. Otherwise, the region of text shall be from the starting cursor (inclusive), to the marked cursor (exclusive), and any text copied to a buffer shall be in character mode.

If not used as a motion command:

*Current line:* Set to the line referenced by the mark.

*Current column:* Set to the last column in which any portion of the character referenced by the

112520 mark is displayed.

## 112521 **Return to Previous Section**

112522 *Synopsis:*     [*count*]   [ [

112523 Move the cursor backward through the edit buffer to the first character of the previous section  
112524 boundary, *count* times.

112525 If used as a motion command:

- 112526     1. If the starting cursor was at the first character of the starting line or the starting line was  
112527       empty, and the first character of the boundary was the first character of the boundary line,  
112528       the text region shall consist of the current line up to and including the line where the  
112529       *countth* next boundary starts, and any text copied to a buffer shall be in line mode.
- 112530     2. If the boundary was the last line of the edit buffer or the last non-<newline> of the last  
112531       line of the edit buffer, the text region shall consist of the last character in the edit buffer up  
112532       to and including the starting character, and any text saved to a buffer shall be in character  
112533       mode.
- 112534     3. Otherwise, the text region shall consist of the starting character up to but not including  
112535       the first character in the *countth* next boundary, and any text copied to a buffer shall be in  
112536       character mode.

112537 If not used as a motion command:

112538 *Current line:* Set to the line where the *countth* next boundary in the edit buffer starts.

112539 *Current column:* Set to the last column in which any portion of the first character of the *countth*  
112540 next boundary is displayed, or column position 1 if the line is empty.

## 112541 **Move to Next Section**

112542 *Synopsis:*     [*count*]   ] ]

112543 Move the cursor forward through the edit buffer to the first character of the next section  
112544 boundary, *count* times.

112545 If used as a motion command:

- 112546     1. If the starting cursor was at the first character of the starting line or the starting line was  
112547       empty, and the first character of the boundary was the first character of the boundary line,  
112548       the text region shall consist of the current line up to and including the line where the  
112549       *countth* previous boundary starts, and any text copied to a buffer shall be in line mode.
- 112550     2. If the boundary was the first line of the edit buffer, the text region shall consist of the first  
112551       character in the edit buffer up to but not including the starting character, and any text  
112552       copied to a buffer shall be in character mode.
- 112553     3. Otherwise, the text region shall consist of the first character in the *countth* previous  
112554       section boundary up to but not including the starting character, and any text copied to a  
112555       buffer shall be in character mode.

112556 If not used as a motion command:

112557 *Current line:* Set to the line where the *countth* previous boundary in the edit buffer starts.

112558 *Current column:* Set to the last column in which any portion of the first character of the *countth*  
112559 previous boundary is displayed, or column position 1 if the line is empty.

**Move to First Non-<blank> Position on Current Line***Synopsis:*     `^`

If used as a motion command:

1. If the line has no non-<blank> non-<newline> characters, or if the cursor is at the first non-<blank> non-<newline> of the line, it shall be an error.
2. If the cursor is before the first non-<blank> non-<newline> of the line, the text region shall be comprised of the current character, up to, but not including, the first non-<blank> non-<newline> of the line.
3. If the cursor is after the first non-<blank> non-<newline> of the line, the text region shall be from the character before the starting cursor up to and including the first non-<blank> non-<newline> of the line.
4. Any text copied to a buffer shall be in character mode.

If not used as a motion command:

*Current line:* Unchanged.*Current column:* Set to non-<blank>.**Current and Line Above***Synopsis:*     `[count] _`If there are less than *count* -1 lines after the current line in the edit buffer, it shall be an error.

If used as a motion command:

1. If *count* is less than 2, the text region shall be the current line.
2. Otherwise, the text region shall include the starting line and the next *count* -1 lines.
3. Any text copied to a buffer shall be in line mode.

If not used as a motion command:

*Current line:* Set to current line + *count* -1.*Current column:* Set to non-<blank>.**Move Back to Beginning of Sentence***Synopsis:*     `[count] (`

Move backward to the beginning of a sentence. This command shall be equivalent to the `[]` command, with the exception that sentence boundaries shall be used instead of section boundaries.

**Move Forward to Beginning of Sentence***Synopsis:*     `[count] )`

Move forward to the beginning of a sentence. This command shall be equivalent to the `]` command, with the exception that sentence boundaries shall be used instead of section boundaries.

**Move Back to Preceding Paragraph**

*Synopsis:*     [*count*] {

Move back to the beginning of the preceding paragraph. This command shall be equivalent to the `[[` command, with the exception that paragraph boundaries shall be used instead of section boundaries.

**Move Forward to Next Paragraph**

*Synopsis:*     [*count*] }

Move forward to the beginning of the next paragraph. This command shall be equivalent to the `]]` command, with the exception that paragraph boundaries shall be used instead of section boundaries.

**Move to Specific Column Position**

*Synopsis:*     [*count*] |

For the purposes of this command, lines that are too long for the current display and that have been folded shall be treated as having a single, 1-based, number of columns.

If there are less than *count* columns in which characters from the current line are displayed on the screen, *count* shall be adjusted to be the last column in which any portion of the line is displayed on the screen.

If used as a motion command:

1. If the line is empty, or the cursor character is the same as the character on the *count*th column of the line, it shall be an error.
2. If the cursor is before the *count*th column of the line, the text region shall be comprised of the current character, up to but not including the character on the *count*th column of the line.
3. If the cursor is after the *count*th column of the line, the text region shall be from the character before the starting cursor up to and including the character on the *count*th column of the line.
4. Any text copied to a buffer shall be in character mode.

If not used as a motion command:

*Current line:* Unchanged.

*Current column:* Set to the last column in which any portion of the character that is displayed in the *count* column of the line is displayed.

**Reverse Find Character**

*Synopsis:*     [*count*] ,

If the last **F**, **f**, **T**, or **t** command was **F**, **f**, **T**, or **t**, this command shall be equivalent to an **f**, **F**, **t**, or **T** command, respectively, with the specified *count* and the same search character.

If there was no previous **F**, **f**, **T**, or **t** command, it shall be an error.

**Repeat**

*Synopsis:*      [*count*] .

Repeat the last **!**, **<**, **>**, **A**, **C**, **D**, **I**, **J**, **O**, **P**, **R**, **S**, **X**, **Y**, **a**, **c**, **d**, **i**, **o**, **p**, **r**, **s**, **x**, **y**, or **~** command. It shall be an error if none of these commands have been executed. Commands (other than commands that enter text input mode) executed as a result of map expansions, shall not change the value of the last repeatable command.

Repeated commands with associated motion commands shall repeat the motion command as well; however, any specified *count* shall replace the *count*(s) that were originally specified to the repeated command or its associated motion command.

If the motion component of the repeated command is **f**, **F**, **t**, or **T**, the repeated command shall not set the remembered search character for the **;** and **,** commands.

If the repeated command is **p** or **P**, and the buffer associated with that command was a numeric buffer named with a number less than 9, the buffer associated with the repeated command shall be set to be the buffer named by the name of the previous buffer logically incremented by 1.

If the repeated character is a text input command, the input text associated with that command is repeated literally:

- Input characters are neither macro or abbreviation-expanded.
- Input characters are not interpreted in any special way with the exception that <newline>, <carriage-return>, and <control>-T behave as described in [Input Mode Commands in vi](#) (on page 3371).

*Current line:* Set as described for the repeated command.

*Current column:* Set as described for the repeated command.

**Find Regular Expression**

*Synopsis:*      /

If the input line contains no non-<newline> characters, it shall be equivalent to a line containing only the last regular expression encountered. The enhanced regular expressions supported by *vi* are described in [Regular Expressions in ex](#) (on page 2699).

Otherwise, the line shall be interpreted as one or more regular expressions, optionally followed by an address offset or a *vi* **z** command.

If the regular expression is not the last regular expression on the line, or if a line offset or **z** command is specified, the regular expression shall be terminated by an unescaped **'/'** character, which shall not be used as part of the regular expression. If the regular expression is not the first regular expression on the line, it shall be preceded by zero or more <blank> characters, a <semicolon>, zero or more <blank> characters, and a leading **'/'** character, which shall not be interpreted as part of the regular expression. It shall be an error to precede any regular expression with any characters other than these.

Each search shall begin from the character after the first character of the last match (or, if it is the first search, after the cursor). If the **wraps**can edit option is set, the search shall continue to the character before the starting cursor character; otherwise, to the end of the edit buffer. It shall be an error if any search fails to find a match, and an informational message to this effect shall be displayed.

An optional address offset (see [Addressing in ex](#), on page 2668) can be specified after the last regular expression by including a trailing **'/'** character after the regular expression and

specifying the address offset. This offset will be from the line containing the match for the last regular expression specified. It shall be an error if the line offset would indicate a line address less than 1 or greater than the last line in the edit buffer. An address offset of zero shall be supported. It shall be an error to follow the address offset with any other characters than <blank> characters.

If not used as a motion command, an optional **z** command (see [Redraw Window](#), on page 3370) can be specified after the last regular expression by including a trailing ' / ' character after the regular expression, zero or more <blank> characters, a ' z ' , zero or more <blank> characters, an optional new **window** edit option value, zero or more <blank> characters, and a location character. The effect shall be as if the **z** command was executed after the / command. It shall be an error to follow the **z** command with any other characters than <blank> characters.

The remembered search direction shall be set to forward.

If used as a motion command:

1. It shall be an error if the last match references the same character in the edit buffer as the starting cursor.
2. If any address offset is specified, the last match shall be adjusted by the specified offset as described previously.
3. If the starting cursor is after the last match, then the locations of the starting cursor and the last match in the edit buffer shall be logically swapped.
4. If any address offset is specified, the text region shall consist of all lines containing characters from the starting cursor to the last match line, inclusive, and any text copied to a buffer shall be in line mode.
5. Otherwise, if the starting line is empty or the starting cursor is at or before the first non-<blank> non-<newline> of the starting line, and the last match line is empty or the last match starts at the first character of the last match line, the text region shall consist of all lines containing characters from the starting cursor to the line before the last match line, inclusive, and any text copied to a buffer shall be in line mode.
6. Otherwise, if the last match line is empty or the last match begins at a character at or before the first non-<blank> non-<newline> of the last match line, the region of text shall be from the current cursor to the last non-<newline> of the line before the last match line, inclusive, and any text copied to a buffer shall be in character mode.
7. Otherwise, the region of text shall be from the current cursor (inclusive), to the first character of the last match (exclusive), and any text copied to a buffer shall be in character mode.

If not used as a motion command:

*Current line*: If a match is found, set to the last matched line plus the address offset, if any; otherwise, unchanged.

*Current column*: Set to the last column on which any portion of the first character in the last matched string is displayed, if a match is found; otherwise, unchanged.



**Move to First Character in Line**

*Synopsis:*     0   (zero)

Move to the first character on the current line. The character '0' shall not be interpreted as a command if it is immediately preceded by a digit.

If used as a motion command:

1. If the cursor character is the first character in the line, it shall be an error.
2. The text region shall be from the character before the cursor character up to and including the first character in the line.
3. Any text copied to a buffer shall be in character mode.

If not used as a motion command:

*Current line:* Unchanged.

*Current column:* The last column in which any portion of the first character in the line is displayed, or if the line is empty, unchanged.

**Execute an ex Command**

*Synopsis:*     :

Execute one or more *ex* commands.

If any portion of the screen other than the last line of the screen was overwritten by any *ex* command (except **shell**), *vi* shall display a message indicating that it is waiting for an input from the user, and shall then read a character. This action may also be taken for other, unspecified reasons.

If the next character entered is a ' : ', another *ex* command shall be accepted and executed. Any other character shall cause the screen to be refreshed and *vi* shall return to command mode.

*Current line:* As specified for the *ex* command.

*Current column:* As specified for the *ex* command.

**Repeat Find**

*Synopsis:*     [count] ;

This command shall be equivalent to the last **F**, **f**, **T**, or **t** command, with the specified *count*, and with the same search character used for the last **F**, **f**, **T**, or **t** command. If there was no previous **F**, **f**, **T**, or **t** command, it shall be an error.

**Shift Left**

*Synopsis:*     [count] < *motion*

If the motion command is the < command repeated:

1. If there are less than *count* - 1 lines after the current line in the edit buffer, it shall be an error.
2. The text region shall be from the current line, up to and including the next *count* - 1 lines.

Shift any line in the text region specified by the *count* and motion command one shiftwidth (see the *ex* **shiftwidth** option) toward the start of the line, as described by the *ex* < command. The unshifted lines shall be copied to the unnamed buffer in line mode.

*Current line:* If the motion was from the current cursor position toward the end of the edit buffer, unchanged. Otherwise, set to the first line in the edit buffer that is part of the text region specified by the motion command.

*Current column:* Set to non-<blank>.

## Shift Right

*Synopsis:*     [*count*] > *motion*

If the motion command is the > command repeated:

1. If there are less than *count* - 1 lines after the current line in the edit buffer, it shall be an error.
2. The text region shall be from the current line, up to and including the next *count* - 1 lines.

Shift any line with characters in the text region specified by the *count* and motion command one shiftwidth (see the *ex* **shiftwidth** option) away from the start of the line, as described by the *ex* > command. The unshifted lines shall be copied into the unnamed buffer in line mode.

*Current line:* If the motion was from the current cursor position toward the end of the edit buffer, unchanged. Otherwise, set to the first line in the edit buffer that is part of the text region specified by the motion command.

*Current column:* Set to non-<blank>.

## Scan Backwards for Regular Expression

*Synopsis:*     ?

Scan backwards; the ? command shall be equivalent to the / command (see [Find Regular Expression](#), on page 3353) with the following exceptions:

1. The input prompt shall be a ' ? '.
2. Each search shall begin from the character before the first character of the last match (or, if it is the first search, the character before the cursor character).
3. The search direction shall be from the cursor toward the beginning of the edit buffer, and the **wrapscan** edit option shall affect whether the search wraps to the end of the edit buffer and continues.
4. The remembered search direction shall be set to backward.

## Execute

*Synopsis:*     @*buffer*

If the *buffer* is specified as @, the last buffer executed shall be used. If no previous buffer has been executed, it shall be an error.

Behave as if the contents of the named buffer were entered as standard input. After each line of a line-mode buffer, and all but the last line of a character mode buffer, behave as if a <newline> were entered as standard input.

If an error occurs during this process, an error message shall be written, and no more characters resulting from the execution of this command shall be processed.

If a *count* is specified, behave as if that count were entered as user input before the characters from the @ buffer were entered.

112790 *Current line*: As specified for the individual commands.

112791 *Current column*: As specified for the individual commands.

## 112792 **Reverse Case**

112793 *Synopsis*:     [*count*] ~

112794 Reverse the case of the current character and the next *count* -1 characters, such that lowercase  
112795 characters that have uppercase counterparts shall be changed to uppercase characters, and  
112796 uppercase characters that have lowercase counterparts shall be changed to lowercase characters,  
112797 as prescribed by the current locale. No other characters shall be affected by this command.

112798 If there are less than *count* -1 characters after the cursor in the edit buffer, *count* shall be adjusted  
112799 to the number of characters after the cursor in the edit buffer minus 1.

112800 For the purposes of this command, the next character after the last non-<newline> on the line  
112801 shall be the next character in the edit buffer.

112802 *Current line*: Set to the line including the (*count*-1)th character after the cursor.

112803 *Current column*: Set to the last column in which any portion of the (*count*-1)th character after the  
112804 cursor is displayed.

## 112805 **Append**

112806 *Synopsis*:     [*count*] a

112807 Enter text input mode after the current cursor position. No characters already in the edit buffer  
112808 shall be affected by this command. A *count* shall cause the input text to be appended *count* -1  
112809 more times to the end of the input.

112810 *Current line/column*: As specified for the text input commands (see [Input Mode Commands in vi](#),  
112811 on page 3371).

## 112812 **Append at End-of-Line**

112813 *Synopsis*:     [*count*] A

112814 This command shall be equivalent to the *vi* command:

112815 \$ [ *count* ] a

112816 (see [Append](#)).

## 112817 **Move Backward to Preceding Word**

112818 *Synopsis*:     [*count*] b

112819 With the exception that words are used as the delimiter instead of bigwords, this command shall  
112820 be equivalent to the **B** command.

## Move Backward to Preceding Bigword

*Synopsis:*     [*count*] B

If the edit buffer is empty or the cursor is on the first character of the edit buffer, it shall be an error. If less than *count* bigwords begin between the cursor and the start of the edit buffer, *count* shall be adjusted to the number of bigword beginnings between the cursor and the start of the edit buffer.

If used as a motion command:

1. The text region shall be from the first character of the *count*th previous bigword beginning up to but not including the cursor character.
2. Any text copied to a buffer shall be in character mode.

If not used as a motion command:

*Current line:* Set to the line containing the *current column*.

*Current column:* Set to the last column upon which any part of the first character of the *count*th previous bigword is displayed.

## Change

*Synopsis:*     [*buffer*] [*count*] c *motion*

If the motion command is the **c** command repeated:

1. The buffer text shall be in line mode.
2. If there are less than *count* - 1 lines after the current line in the edit buffer, it shall be an error.
3. The text region shall be from the current line up to and including the next *count* - 1 lines.

Otherwise, the buffer text mode and text region shall be as specified by the motion command.

The replaced text shall be copied into *buffer*, if specified, and into the unnamed buffer. If the text to be replaced contains characters from more than a single line, or the buffer text is in line mode, the replaced text shall be copied into the numeric buffers as well.

If the buffer text is in line mode:

1. Any lines that contain characters in the region shall be deleted, and the editor shall enter text input mode at the beginning of a new line which shall replace the first line deleted.
2. If the **autoindent** edit option is set, **autoindent** characters equal to the **autoindent** characters on the first line deleted shall be inserted as if entered by the user.

Otherwise, if characters from more than one line are in the region of text:

1. The text shall be deleted.
2. Any text remaining in the last line in the text region shall be appended to the first line in the region, and the last line in the region shall be deleted.
3. The editor shall enter text input mode after the last character not deleted from the first line in the text region, if any; otherwise, on the first column of the first line in the region.

Otherwise:

1. If the glyph for ' \$ ' is smaller than the region, the end of the region shall be marked with a ' \$ '.
2. The editor shall enter text input mode, overwriting the region of text.

*Current line/column:* As specified for the text input commands (see [Input Mode Commands in vi](#), on page 3371).

### Change to End-of-Line

*Synopsis:*     [*buffer*] [*count*] C

This command shall be equivalent to the *vi* command:

[*buffer*] [*count*] c\$

See the **c** command.

### Delete

*Synopsis:*     [*buffer*] [*count*] d *motion*

If the motion command is the **d** command repeated:

1. The buffer text shall be in line mode.
2. If there are less than *count* - 1 lines after the current line in the edit buffer, it shall be an error.
3. The text region shall be from the current line up to and including the next *count* - 1 lines.

Otherwise, the buffer text mode and text region shall be as specified by the motion command.

If in open mode, and the current line is deleted, and the line remains on the display, an '@' character shall be displayed as the first glyph of that line.

Delete the region of text into *buffer*, if specified, and into the unnamed buffer. If the text to be deleted contains characters from more than a single line, or the buffer text is in line mode, the deleted text shall be copied into the numeric buffers, as well.

*Current line:* Set to the first text region line that appears in the edit buffer, unless that line has been deleted, in which case it shall be set to the last line in the edit buffer, or line 1 if the edit buffer is empty.

*Current column:*

1. If the line is empty, set to column position 1.
2. Otherwise, if the buffer text is in line mode or the motion was from the cursor toward the end of the edit buffer:
  - a. If a character from the current line is displayed in the current column, set to the last column that displays any portion of that character.
  - b. Otherwise, set to the last column in which any portion of any character in the line is displayed.
3. Otherwise, if a character is displayed in the column that began the text region, set to the last column that displays any portion of that character.

- 112894 4. Otherwise, set to the last column in which any portion of any character in the line is  
112895 displayed.

#### 112896 **Delete to End-of-Line**

112897 *Synopsis:*     [*buffer*] D

112898 Delete the text from the current position to the end of the current line; equivalent to the *vi*  
112899 command:

112900 [*buffer*] d\$

#### 112901 **Move to End-of-Word**

112902 *Synopsis:*     [*count*] e

112903 With the exception that words are used instead of bigwords as the delimiter, this command shall  
112904 be equivalent to the **E** command.

#### 112905 **Move to End-of-Bigword**

112906 *Synopsis:*     [*count*] E

112907 If the edit buffer is empty it shall be an error. If less than *count* bigwords end between the cursor  
112908 and the end of the edit buffer, *count* shall be adjusted to the number of bigword endings between  
112909 the cursor and the end of the edit buffer.

112910 If used as a motion command:

- 112911 1. The text region shall be from the last character of the *count*th next bigword up to and  
112912 including the cursor character.
- 112913 2. Any text copied to a buffer shall be in character mode.

112914 If not used as a motion command:

112915 *Current line:* Set to the line containing the current column.

112916 *Current column:* Set to the last column upon which any part of the last character of the *count*th  
112917 next bigword is displayed.

#### 112918 **Find Character in Current Line (Forward)**

112919 *Synopsis:*     [*count*] f *character*

112920 It shall be an error if *count* occurrences of the character do not occur after the cursor in the line.

112921 If used as a motion command:

- 112922 1. The text range shall be from the cursor character up to and including the *count*th  
112923 occurrence of the specified character after the cursor.
- 112924 2. Any text copied to a buffer shall be in character mode.

112925 If not used as a motion command:

112926 *Current line:* Unchanged.

112927 *Current column:* Set to the last column in which any portion of the *count*th occurrence of the  
112928 specified character after the cursor appears in the line.

**Find Character in Current Line (Reverse)**

*Synopsis:*     [*count*] F *character*

It shall be an error if *count* occurrences of the character do not occur before the cursor in the line.

If used as a motion command:

1. The text region shall be from the *count*th occurrence of the specified character before the cursor, up to, but not including the cursor character.
2. Any text copied to a buffer shall be in character mode.

If not used as a motion command:

*Current line:* Unchanged.

*Current column:* Set to the last column in which any portion of the *count*th occurrence of the specified character before the cursor appears in the line.

**Move to Line**

*Synopsis:*     [*count*] G

If *count* is not specified, it shall default to the last line of the edit buffer. If *count* is greater than the last line of the edit buffer, it shall be an error.

If used as a motion command:

1. The text region shall be from the cursor line up to and including the specified line.
2. Any text copied to a buffer shall be in line mode.

If not used as a motion command:

*Current line:* Set to *count* if *count* is specified; otherwise, the last line.

*Current column:* Set to non-<blank>.

**Move to Top of Screen**

*Synopsis:*     [*count*] H

If the beginning of the line *count* greater than the first line of which any portion appears on the display does not exist, it shall be an error.

If used as a motion command:

1. If in open mode, the text region shall be the current line.
2. Otherwise, the text region shall be from the starting line up to and including (the first line of the display + *count* - 1).
3. Any text copied to a buffer shall be in line mode.

If not used as a motion command:

If in open mode, this command shall set the current column to non-<blank> and do nothing else.

Otherwise, it shall set the current line and current column as follows.

*Current line:* Set to (the first line of the display + *count* - 1).

*Current column:* Set to non-<blank>.

**Insert Before Cursor**

*Synopsis:*     [*count*] i

Enter text input mode before the current cursor position. No characters already in the edit buffer shall be affected by this command. A *count* shall cause the input text to be appended *count* -1 more times to the end of the input.

*Current line/column:* As specified for the text input commands (see [Input Mode Commands in vi](#), on page 3371).

**Insert at Beginning of Line**

*Synopsis:*     [*count*] I

This command shall be equivalent to the *vi* command `^[count]i`.

**Join**

*Synopsis:*     [*count*] J

If the current line is the last line in the edit buffer, it shall be an error.

This command shall be equivalent to the *ex* **join** command with no addresses, and an *ex* command *count* value of 1 if *count* was not specified or if a *count* of 1 was specified, and an *ex* command *count* value of *count* -1 for any other value of *count*, except that the current line and column shall be set as follows.

*Current line:* Unchanged.

*Current column:* The last column in which any portion of the character following the last character in the initial line is displayed, or the last non-<newline> in the line if no characters were appended.

**Move to Bottom of Screen**

*Synopsis:*     [*count*] L

If the beginning of the line *count* less than the last line of which any portion appears on the display does not exist, it shall be an error.

If used as a motion command:

1. If in open mode, the text region shall be the current line.
2. Otherwise, the text region shall include all lines from the starting cursor line to (the last line of the display -(*count* -1)).
3. Any text copied to a buffer shall be in line mode.

If not used as a motion command:

1. If in open mode, this command shall set the current column to non-<blank> and do nothing else.
2. Otherwise, it shall set the current line and current column as follows.

*Current line:* Set to (the last line of the display -(*count* -1)).

*Current column:* Set to non-<blank>.



## 113000 **Mark Position**

113001 *Synopsis:*     m letter

113002 This command shall be equivalent to the *ex* **mark** command with the specified character as an  
113003 argument.

## 113004 **Move to Middle of Screen**

113005 *Synopsis:*     M

113006 The middle line of the display shall be calculated as follows:

113007 (the top line of the display) + (((number of lines displayed) + 1) / 2) - 1

113008 If used as a motion command:

- 113009     1. If in open mode, the text region shall be the current line.
- 113010     2. Otherwise, the text region shall include all lines from the starting cursor line up to and  
113011         including the middle line of the display.
- 113012     3. Any text copied to a buffer shall be in line mode.

113013 If not used as a motion command:

113014 If in open mode, this command shall set the current column to non-<blank> and do nothing else.

113015 Otherwise, it shall set the current line and current column as follows.

113016 *Current line:* Set to the middle line of the display.

113017 *Current column:* Set to non-<blank>.

## 113018 **Repeat Regular Expression Find (Forward)**

113019 *Synopsis:*     n

113020 If the remembered search direction was forward, the **n** command shall be equivalent to the *vi* /  
113021 command with no characters entered by the user. Otherwise, it shall be equivalent to the *vi* ?  
113022 command with no characters entered by the user.

113023 If the **n** command is used as a motion command for the **!** command, the editor shall not enter  
113024 text input mode on the last line on the screen, and shall behave as if the user entered a single  
113025 '!' character as the text input.

## 113026 **Repeat Regular Expression Find (Reverse)**

113027 *Synopsis:*     N

113028 Scan for the next match of the last pattern given to / or ?, but in the reverse direction; this is the  
113029 reverse of **n**.

113030 If the remembered search direction was forward, the **N** command shall be equivalent to the *vi* ?  
113031 command with no characters entered by the user. Otherwise, it shall be equivalent to the *vi* /  
113032 command with no characters entered by the user. If the **N** command is used as a motion  
113033 command for the **!** command, the editor shall not enter text input mode on the last line on the  
113034 screen, and shall behave as if the user entered a single **!** character as the text input.

**Insert Empty Line Below**

*Synopsis:*      ○

Enter text input mode in a new line appended after the current line. A *count* shall cause the input text to be appended *count* -1 more times to the end of the already added text, each time starting on a new, appended line.

*Current line/column:* As specified for the text input commands (see [Input Mode Commands in vi](#), on page 3371).

**Insert Empty Line Above**

*Synopsis:*      ○

Enter text input mode in a new line inserted before the current line. A *count* shall cause the input text to be appended *count* -1 more times to the end of the already added text, each time starting on a new, appended line.

*Current line/column:* As specified for the text input commands (see [Input Mode Commands in vi](#), on page 3371).

**Put from Buffer Following**

*Synopsis:*      [*buffer*] p

If no *buffer* is specified, the unnamed buffer shall be used.

If the buffer text is in line mode, the text shall be appended below the current line, and each line of the buffer shall become a new line in the edit buffer. A *count* shall cause the buffer text to be appended *count* -1 more times to the end of the already added text, each time starting on a new, appended line.

If the buffer text is in character mode, the text shall be appended into the current line after the cursor, and each line of the buffer other than the first and last shall become a new line in the edit buffer. A *count* shall cause the buffer text to be appended *count* -1 more times to the end of the already added text, each time starting after the last added character.

*Current line:* If the buffer text is in line mode, set the line to line +1; otherwise, unchanged.

*Current column:* If the buffer text is in line mode:

1. If there is a non-<blank> in the first line of the buffer, set to the last column on which any portion of the first non-<blank> in the line is displayed.
2. If there is no non-<blank> in the first line of the buffer, set to the last column on which any portion of the last non-<newline> in the first line of the buffer is displayed.

If the buffer text is in character mode:

1. If the text in the buffer is from more than a single line, then set to the last column on which any portion of the first character from the buffer is displayed.
2. Otherwise, if the buffer is the unnamed buffer, set to the last column on which any portion of the last character from the buffer is displayed.
3. Otherwise, set to the first column on which any portion of the first character from the buffer is displayed.

**Put from Buffer Before**

*Synopsis:*      **[buffer]** P

If no *buffer* is specified, the unnamed buffer shall be used.

If the buffer text is in line mode, the text shall be inserted above the current line, and each line of the buffer shall become a new line in the edit buffer. A *count* shall cause the buffer text to be appended *count* -1 more times to the end of the already added text, each time starting on a new, appended line.

If the buffer text is in character mode, the text shall be inserted into the current line before the cursor, and each line of the buffer other than the first and last shall become a new line in the edit buffer. A *count* shall cause the buffer text to be appended *count* -1 more times to the end of the already added text, each time starting after the last added character.

*Current line:* Unchanged.

*Current column:* If the buffer text is in line mode:

1. If there is a non-<blank> in the first line of the buffer, set to the last column on which any portion of that character is displayed.
2. If there is no non-<blank> in the first line of the buffer, set to the last column on which any portion of the last non-<newline> in the first line of the buffer is displayed.

If the buffer text is in character mode:

1. If the text in the buffer is from more than a single line, then set to the last column on which any portion of the first character from the buffer is displayed.
2. Otherwise, if the buffer is the unnamed buffer, set to the last column on which any portion of the last character from the buffer is displayed.
3. Otherwise, set to the first column on which any portion of the first character from the buffer is displayed.

**Enter ex Mode**

*Synopsis:*      Q

Leave visual or open mode and enter *ex* command mode.

*Current line:* Unchanged.

*Current column:* Unchanged.

**Replace Character**

*Synopsis:*      **[count]** r *character*

Replace the *count* characters at and after the cursor with the specified character. If there are less than *count* non-<newline> characters at and after the cursor on the line, it shall be an error.

If character is <control>-V, any next character other than the <newline> shall be stripped of any special meaning and used as a literal character.

If character is <ESC>, no replacement shall be made and the current line and current column shall be unchanged.

If character is <carriage-return> or <newline>, *count* new lines shall be appended to the current line. All but the last of these lines shall be empty. *count* characters at and after the cursor shall be

discarded, and any remaining characters after the cursor in the current line shall be moved to the last of the new lines. If the **autoindent** edit option is set, they shall be preceded by the same number of **autoindent** characters found on the line from which the command was executed.

*Current line*: Unchanged unless the replacement character is a <carriage-return> or <newline>, in which case it shall be set to line + *count*.

*Current column*: Set to the last column position on which a portion of the last replaced character is displayed, or if the replacement character caused new lines to be created, set to non-<blank>.

### Replace Characters

*Synopsis*:       R

Enter text input mode at the current cursor position possibly replacing text on the current line. A *count* shall cause the input text to be appended *count* - 1 more times to the end of the input.

*Current line/column*: As specified for the text input commands (see [Input Mode Commands in vi](#), on page 3371).

### Substitute Character

*Synopsis*:       [*buffer*] [*count*] s

This command shall be equivalent to the *vi* command:

[*buffer*] [*count*] c<space>

### Substitute Lines

*Synopsis*:       [*buffer*] [*count*] S

This command shall be equivalent to the *vi* command:

[*buffer*] [*count*] c\_

### Move Cursor to Before Character (Forward)

*Synopsis*:       [*count*] t *character*

It shall be an error if *count* occurrences of the character do not occur after the cursor in the line.

If used as a motion command:

1. The text region shall be from the cursor up to but not including the *count*th occurrence of the specified character after the cursor.
2. Any text copied to a buffer shall be in character mode.

If not used as a motion command:

*Current line*: Unchanged.

*Current column*: Set to the last column in which any portion of the character before the *count*th occurrence of the specified character after the cursor appears in the line.

**Move Cursor to After Character (Reverse)**

*Synopsis:*     `[count] T character`

It shall be an error if *count* occurrences of the character do not occur before the cursor in the line.

If used as a motion command:

1. If the character before the cursor is the specified character, it shall be an error.
2. The text region shall be from the character before the cursor up to but not including the *count*th occurrence of the specified character before the cursor.
3. Any text copied to a buffer shall be in character mode.

If not used as a motion command:

*Current line:* Unchanged.

*Current column:* Set to the last column in which any portion of the character after the *count*th occurrence of the specified character before the cursor appears in the line.

**Undo**

*Synopsis:*     `u`

This command shall be equivalent to the *ex* **undo** command except that the current line and current column shall be set as follows:

*Current line:* Set to the first line added or changed if any; otherwise, move to the line preceding any deleted text if one exists; otherwise, move to line 1.

*Current column:* If undoing an *ex* command, set to the first non-<blank>.

Otherwise, if undoing a text input command:

1. If the command was a **C**, **c**, **O**, **o**, **R**, **S**, or **s** command, the current column shall be set to the value it held when the text input command was entered.
2. Otherwise, set to the last column in which any portion of the first character after the deleted text is displayed, or, if no non-<newline> characters follow the text deleted from this line, set to the last column in which any portion of the last non-<newline> in the line is displayed, or 1 if the line is empty.

Otherwise, if a single line was modified (that is, not added or deleted) by the **u** command:

1. If text was added or changed, set to the last column in which any portion of the first character added or changed is displayed.
2. If text was deleted, set to the last column in which any portion of the first character after the deleted text is displayed, or, if no non-<newline> characters follow the deleted text, set to the last column in which any portion of the last non-<newline> in the line is displayed, or 1 if the line is empty.

Otherwise, set to non-<blank>.

**Undo Current Line**

*Synopsis:*     U

Restore the current line to its state immediately before the most recent time that it became the current line.

*Current line:* Unchanged.

*Current column:* Set to the first column in the line in which any portion of the first character in the line is displayed.

**Move to Beginning of Word**

*Synopsis:*     [count] w

With the exception that words are used as the delimiter instead of bigwords, this command shall be equivalent to the **W** command.

**Move to Beginning of Bigword**

*Synopsis:*     [count] W

If the edit buffer is empty, it shall be an error. If there are less than *count* bigwords between the cursor and the end of the edit buffer, *count* shall be adjusted to move the cursor to the last bigword in the edit buffer.

If used as a motion command:

1. If the associated command is **c**, *count* is 1, and the cursor is on a <blank>, the region of text shall be the current character and no further action shall be taken.
2. If there are less than *count* bigwords between the cursor and the end of the edit buffer, then the command shall succeed, and the region of text shall include the last character of the edit buffer.
3. If there are <blank> characters or an end-of-line that precede the *count*th bigword, and the associated command is **c**, the region of text shall be up to and including the last character before the preceding <blank> characters or end-of-line.
4. If there are <blank> characters or an end-of-line that precede the bigword, and the associated command is **d** or **y**, the region of text shall be up to and including the last <blank> before the start of the bigword or end-of-line.
5. Any text copied to a buffer shall be in character mode.

If not used as a motion command:

1. If the cursor is on the last character of the edit buffer, it shall be an error.

*Current line:* Set to the line containing the current column.

*Current column:* Set to the last column in which any part of the first character of the *count*th next bigword is displayed.

**Delete Character at Cursor**

*Synopsis:*      `[buffer] [count] x`

Delete the *count* characters at and after the current character into *buffer*, if specified, and into the unnamed buffer.

If the line is empty, it shall be an error. If there are less than *count* non-<newline> characters at and after the cursor on the current line, *count* shall be adjusted to the number of non-<newline> characters at and after the cursor.

*Current line:* Unchanged.

*Current column:* If the line is empty, set to column position 1. Otherwise, if there were *count* or less non-<newline> characters at and after the cursor on the current line, set to the last column that displays any part of the last non-<newline> of the line. Otherwise, unchanged.

**Delete Character Before Cursor**

*Synopsis:*      `[buffer] [count] X`

Delete the *count* characters before the current character into *buffer*, if specified, and into the unnamed buffer.

If there are no characters before the current character on the current line, it shall be an error. If there are less than *count* previous characters on the current line, *count* shall be adjusted to the number of previous characters on the line.

*Current line:* Unchanged.

*Current column:* Set to (current column – the width of the deleted characters).

**Yank**

*Synopsis:*      `[buffer] [count] y motion`

Copy (yank) the region of text into *buffer*, if specified, and into the unnamed buffer.

If the motion command is the *y* command repeated:

1. The buffer shall be in line mode.
2. If there are less than *count* – 1 lines after the current line in the edit buffer, it shall be an error.
3. The text region shall be from the current line up to and including the next *count* – 1 lines.

Otherwise, the buffer text mode and text region shall be as specified by the motion command.

*Current line:* If the motion was from the current cursor position toward the end of the edit buffer, unchanged. Otherwise, set to the first line in the edit buffer that is part of the text region specified by the motion command.

*Current column:*

1. If the motion was from the current cursor position toward the end of the edit buffer, unchanged.
2. Otherwise, if the current line is empty, set to column position 1.
3. Otherwise, set to the last column that displays any part of the first character in the file that is part of the text region specified by the motion command.

**Yank Current Line**

*Synopsis:*     [*buffer*] [*count*] Y

This command shall be equivalent to the *vi* command:

[*buffer*] [*count*] y\_

**Redraw Window**

If in open mode, the **z** command shall have the Synopsis:

*Synopsis:*     [*count*] z

If *count* is not specified, it shall default to the **window** edit option *-1*. The **z** command shall be equivalent to the *ex z* command, with a type character of **=** and a *count* of *count -2*, except that the current line and current column shall be set as follows, and the **window** edit option shall not be affected. If the calculation for the *count* argument would result in a negative number, the *count* argument to the *ex z* command shall be zero. A blank line shall be written after the last line is written.

*Current line:* Unchanged.

*Current column:* Unchanged.

If not in open mode, the **z** command shall have the following Synopsis:

*Synopsis:*     [*line*] z [*count*] *character*

If *line* is not specified, it shall default to the current line. If *line* is specified, but is greater than the number of lines in the edit buffer, it shall default to the number of lines in the edit buffer.

If *count* is specified, the value of the **window** edit option shall be set to *count* (as described in the *ex window* command), and the screen shall be redrawn.

*line* shall be placed as specified by the following characters:

<newline>, <carriage-return>

Place the beginning of the line on the first line of the display.

. Place the beginning of the line in the center of the display. The middle line of the display shall be calculated as described for the **M** command.

– Place an unspecified portion of the line on the last line of the display.

+ If *line* was specified, equivalent to the <newline> case. If *line* was not specified, display a screen where the first line of the display shall be (current last line) +1. If there are no lines after the last line in the display, it shall be an error.

^ If *line* was specified, display a screen where the last line of the display shall contain an unspecified portion of the first line of a display that had an unspecified portion of the specified line on the last line of the display. If this calculation results in a line before the beginning of the edit buffer, display the first screen of the edit buffer.

Otherwise, display a screen where the last line of the display shall contain an unspecified portion of (current first line *-1*). If this calculation results in a line before the beginning of the edit buffer, it shall be an error.



Current line: If *line* and the ' ^ ' character were specified:

1. If the first screen was displayed as a result of the command attempting to display lines before the beginning of the edit buffer: if the first screen was already displayed, unchanged; otherwise, set to (current first line -1).
2. Otherwise, set to the last line of the display.

If *line* and the ' +' character were specified, set to the first line of the display.

Otherwise, if *line* was specified, set to *line*.

Otherwise, unchanged.

Current column: Set to non-<blank>.

## Exit

Synopsis:      ZZ

This command shall be equivalent to the *ex xit* command with no addresses, trailing *!*, or filename (see the *ex xit* command).

## Input Mode Commands in vi

In text input mode, the current line shall consist of zero or more of the following categories, plus the terminating <newline>:

1. Characters preceding the text input entry point

Characters in this category shall not be modified during text input mode.

2. **autoindent** characters

**autoindent** characters shall be automatically inserted into each line that is created in text input mode, either as a result of entering a <newline> or <carriage-return> while in text input mode, or as an effect of the command itself; for example, **O** or **o** (see the *ex autoindent* command), as if entered by the user.

It shall be possible to erase **autoindent** characters with the <control>-D command; it is unspecified whether they can be erased by <control>-H, <control>-U, and <control>-W characters. Erasing any **autoindent** character turns the glyph into erase-columns and deletes the character from the edit buffer, but does not change its representation on the screen.

3. Text input characters

Text input characters are the characters entered by the user. Erasing any text input character turns the glyph into erase-columns and deletes the character from the edit buffer, but does not change its representation on the screen.

Each text input character entered by the user (that does not have a special meaning) shall be treated as follows:

- a. The text input character shall be appended to the last character in the edit buffer from the first, second, or third categories.
- b. If there are no erase-columns on the screen, the text input command was the **R** command, and characters in the fifth category from the original line follow the cursor, the next such character shall be deleted from the edit buffer. If the **slowopen** edit option is not set, the corresponding glyph on the screen shall

- 113327 become erase-columns.
- 113328 c. If there are erase-columns on the screen, as many columns as they occupy, or as are  
113329 necessary, shall be overwritten to display the text input character. (If only part of a  
113330 multi-column glyph is overwritten, the remainder shall be left on the screen, and  
113331 continue to be treated as erase-columns; it is unspecified whether the remainder of  
113332 the glyph is modified in any way.)
- 113333 d. If additional display line columns are needed to display the text input character:
- 113334 i. If the **slowopen** edit option is set, the text input characters shall be  
113335 displayed on subsequent display line columns, overwriting any characters  
113336 displayed in those columns.
- 113337 ii. Otherwise, any characters currently displayed on or after the column on the  
113338 display line where the text input character is to be displayed shall be  
113339 pushed ahead the number of display line columns necessary to display the  
113340 rest of the text input character.
- 113341 4. Erase-columns
- 113342 Erase-columns are not logically part of the edit buffer, appearing only on the screen, and  
113343 may be overwritten on the screen by subsequent text input characters. When text input  
113344 mode ends, all erase-columns shall no longer appear on the screen.
- 113345 Erase-columns are initially the region of text specified by the **c** command (see [Change](#), on  
113346 page 3358); however, erasing **autoindent** or text input characters causes the glyphs of the  
113347 erased characters to be treated as erase-columns.
- 113348 5. Characters following the text region for the **c** command, or the text input entry point for  
113349 all other commands
- 113350 Characters in this category shall not be modified during text input mode, except as  
113351 specified in category 3.b. for the **R** text input command, or as <blank> characters deleted  
113352 when a <newline> or <carriage-return> is entered.
- 113353 It is unspecified whether it is an error to attempt to erase past the beginning of a line that was  
113354 created by the entry of a <newline> or <carriage-return> during text input mode. If it is not an  
113355 error, the editor shall behave as if the erasing character was entered immediately after the last  
113356 text input character entered on the previous line, and all of the non-<newline> characters on the  
113357 current line shall be treated as erase-columns.
- 113358 When text input mode is entered, or after a text input mode character is entered (except as  
113359 specified for the special characters below), the cursor shall be positioned as follows:
- 113360 1. On the first column that displays any part of the first erase-column, if one exists
- 113361 2. Otherwise, if the **slowopen** edit option is set, on the first display line column after the last  
113362 character in the first, second, or third categories, if one exists
- 113363 3. Otherwise, the first column that displays any part of the first character in the fifth  
113364 category, if one exists
- 113365 4. Otherwise, the display line column after the last character in the first, second, or third  
113366 categories, if one exists
- 113367 5. Otherwise, on column position 1
- 113368 The characters that are updated on the screen during text input mode are unspecified, other than  
113369 that the last text input character shall always be updated, and, if the **slowopen** edit option is not

113370 set, the current cursor character shall always be updated.

113371 The following specifications are for command characters entered during text input mode.

113372 **NUL**

113373 *Synopsis:* NUL

113374 If the first character of the text input is a NUL, the most recently input text shall be input as if  
 113375 entered by the user, and then text input mode shall be exited. The text shall be input literally;  
 113376 that is, characters are neither macro or abbreviation expanded, nor are any characters interpreted  
 113377 in any special manner. It is unspecified whether implementations shall support more than 256  
 113378 bytes of remembered input text.

113379 **<control>-D**

113380 *Synopsis:* <control>-D

113381 The <control>-D character shall have no special meaning when in text input mode for a line-  
 113382 oriented command (see [Command Descriptions in vi](#), on page 3337).

113383 This command need not be supported on block-mode terminals.

113384 If the cursor does not follow an **autoindent** character, or an **autoindent** character and a '0' or  
 113385 '^' character:

- 113386 1. If the cursor is in column position 1, the <control>-D character shall be discarded and no  
 113387 further action taken.
- 113388 2. Otherwise, the <control>-D character shall have no special meaning.

113389 If the last input character was a '0', the cursor shall be moved to column position 1.

113390 Otherwise, if the last input character was a '^', the cursor shall be moved to column position 1.  
 113391 In addition, the **autoindent** level for the next input line shall be derived from the same line from  
 113392 which the **autoindent** level for the current input line was derived.

113393 Otherwise, the cursor shall be moved back to the column after the previous shiftwidth (see the  
 113394 *ex* **shiftwidth** command) boundary.

113395 All of the glyphs on columns between the starting cursor position and (inclusively) the ending  
 113396 cursor position shall become erase-columns as described in [Input Mode Commands in vi](#) (on  
 113397 page 3371).

113398 *Current line:* Unchanged.

113399 *Current column:* Set to 1 if the <control>-D was preceded by a '^' or '0'; otherwise, set to  
 113400 (column -1) - ((column -2) % **shiftwidth**).

113401 **<control>-H**

113402 *Synopsis:* <control>-H

113403 If in text input mode for a line-oriented command, and there are no characters to erase, text  
 113404 input mode shall be terminated, no further action shall be done for this command, and the  
 113405 current line and column shall be unchanged.

113406 If there are characters other than **autoindent** characters that have been input on the current line  
 113407 before the cursor, the cursor shall move back one character.

113408 Otherwise, if there are **autoindent** characters on the current line before the cursor, it is

implementation-defined whether the <control>-H command is an error or if the cursor moves back one **autoindent** character.

Otherwise, if the cursor is in column position 1 and there are previous lines that have been input, it is implementation-defined whether the <control>-H command is an error or if it is equivalent to entering <control>-H after the last input character on the previous input line.

Otherwise, it shall be an error.

All of the glyphs on columns between the starting cursor position and (inclusively) the ending cursor position shall become erase-columns as described in [Input Mode Commands in vi](#) (on page 3371).

The current erase character (see *stty*) shall cause an equivalent action to the <control>-H command, unless the previously inserted character was a <backslash>, in which case it shall be as if the literal current erase character had been inserted instead of the <backslash>.

*Current line*: Unchanged, unless previously input lines are erased, in which case it shall be set to line -1.

*Current column*: Set to the first column that displays any portion of the character backed up over.

#### <newline>

*Synopsis*:     <newline>  
                   <carriage-return>  
                   <control>-J  
                   <control>-M

If input was part of a line-oriented command, text input mode shall be terminated and the command shall continue execution with the input provided.

Otherwise, terminate the current line. If there are no characters other than **autoindent** characters on the line, all characters on the line shall be discarded. Otherwise, it is unspecified whether the **autoindent** characters in the line are modified by entering these characters.

Continue text input mode on a new line appended after the current line. If the **slowopen** edit option is set, the lines on the screen below the current line shall not be pushed down, but the first of them shall be cleared and shall appear to be overwritten. Otherwise, the lines of the screen below the current line shall be pushed down.

If the **autoindent** edit option is set, an appropriate number of **autoindent** characters shall be added as a prefix to the line as described by the *ex* **autoindent** edit option.

All columns after the cursor that are erase-columns (as described in [Input Mode Commands in vi](#), on page 3371) shall be discarded.

If the **autoindent** edit option is set, all <blank> characters immediately following the cursor shall be discarded.

All remaining characters after the cursor shall be transferred to the new line, positioned after any **autoindent** characters.

*Current line*: Set to current line +1.

*Current column*: Set to the first column that displays any portion of the first character after the **autoindent** characters on the new line, if any, or the first column position after the last **autoindent** character, if any, or column position 1.

113450 **<control>-T**113451 *Synopsis:*      <control>-T113452 The <control>-T character shall have no special meaning when in text input mode for a line-  
113453 oriented command (see [Command Descriptions in vi](#), on page 3337).

113454 This command need not be supported on block-mode terminals.

113455 Behave as if the user entered the minimum number of <blank> characters necessary to move the  
113456 cursor forward to the column position after the next **shiftwidth** (see the *ex* **shiftwidth**  
113457 command) boundary.113458 *Current line:* Unchanged.113459 *Current column:* Set to  $column + shiftwidth - ((column - 1) \% shiftwidth)$ .113460 **<control>-U**113461 *Synopsis:*      <control>-U113462 If there are characters other than **autoindent** characters that have been input on the current line  
113463 before the cursor, the cursor shall move to the first character input after the **autoindent**  
113464 characters.113465 Otherwise, if there are **autoindent** characters on the current line before the cursor, it is  
113466 implementation-defined whether the <control>-U command is an error or if the cursor moves to  
113467 the first column position on the line.113468 Otherwise, if the cursor is in column position 1 and there are previous lines that have been  
113469 input, it is implementation-defined whether the <control>-U command is an error or if it is  
113470 equivalent to entering <control>-U after the last input character on the previous input line.

113471 Otherwise, it shall be an error.

113472 All of the glyphs on columns between the starting cursor position and (inclusively) the ending  
113473 cursor position shall become erase-columns as described in [Input Mode Commands in vi](#) (on  
113474 page 3371).113475 The current *kill* character (see *stty*) shall cause an equivalent action to the <control>-U command,  
113476 unless the previously inserted character was a <backslash>, in which case it shall be as if the  
113477 literal current *kill* character had been inserted instead of the <backslash>.113478 *Current line:* Unchanged, unless previously input lines are erased, in which case it shall be set to  
113479 line -1.113480 *Current column:* Set to the first column that displays any portion of the last character backed up  
113481 over.113482 **<control>-V**113483 *Synopsis:*      <control>-V  
113484                      <control>-Q113485 Allow the entry of any subsequent character, other than <control>-J or the <newline>, as a literal  
113486 character, removing any special meaning that it may have to the editor in text input mode. If a  
113487 <control>-V or <control>-Q is entered before a <control>-J or <newline>, the <control>-V or  
113488 <control>-Q character shall be discarded, and the <control>-J or <newline> shall behave as  
113489 described in the <newline> command character during input mode.

113490 For purposes of the display only, the editor shall behave as if a ' ^ ' character was entered, and

113491 the cursor shall be positioned as if overwriting the ' ^ ' character. When a subsequent character  
 113492 is entered, the editor shall behave as if that character was entered instead of the original  
 113493 <control>-V or <control>-Q character.

113494 *Current line:* Unchanged.

113495 *Current column:* Unchanged.

#### 113496 <control>-W

113497 *Synopsis:* <control>-W

113498 If there are characters other than **autoindent** characters that have been input on the current line  
 113499 before the cursor, the cursor shall move back over the last word preceding the cursor (including  
 113500 any <blank> characters between the end of the last word and the current cursor); the cursor shall  
 113501 not move to before the first character after the end of any **autoindent** characters.

113502 Otherwise, if there are **autoindent** characters on the current line before the cursor, it is  
 113503 implementation-defined whether the <control>-W command is an error or if the cursor moves to  
 113504 the first column position on the line.

113505 Otherwise, if the cursor is in column position 1 and there are previous lines that have been  
 113506 input, it is implementation-defined whether the <control>-W command is an error or if it is  
 113507 equivalent to entering <control>-W after the last input character on the previous input line.

113508 Otherwise, it shall be an error.

113509 All of the glyphs on columns between the starting cursor position and (inclusively) the ending  
 113510 cursor position shall become erase-columns as described in [Input Mode Commands in vi](#) (on  
 113511 page 3371).

113512 *Current line:* Unchanged, unless previously input lines are erased, in which case it shall be set to  
 113513 line -1.

113514 *Current column:* Set to the first column that displays any portion of the last character backed up  
 113515 over.

#### 113516 <ESC>

113517 *Synopsis:* <ESC>

113518 If input was part of a line-oriented command:

- 113519 1. If *interrupt* was entered, text input mode shall be terminated and the editor shall return to  
 113520 command mode. The terminal shall be alerted.
- 113521 2. If <ESC> was entered, text input mode shall be terminated and the command shall  
 113522 continue execution with the input provided.

113523 Otherwise, terminate text input mode and return to command mode.

113524 Any **autoindent** characters entered on newly created lines that have no other non-<newline>  
 113525 characters shall be deleted.

113526 Any leading **autoindent** and <blank> characters on newly created lines shall be rewritten to be  
 113527 the minimum number of <blank> characters possible.

113528 The screen shall be redisplayed as necessary to match the contents of the edit buffer.

113529 *Current line:* Unchanged.

113530 *Current column:*

- 113531 1. If there are text input characters on the current line, the column shall be set to the last  
113532 column where any portion of the last text input character is displayed.
- 113533 2. Otherwise, if a character is displayed in the current column, unchanged.
- 113534 3. Otherwise, set to column position 1.

#### 113535 **EXIT STATUS**

113536 The following exit values shall be returned:

- 113537 0 Successful completion.
- 113538 >0 An error occurred.

#### 113539 **CONSEQUENCES OF ERRORS**

113540 When any error is encountered and the standard input is not a terminal device file, *vi* shall not  
113541 write the file or return to command or text input mode, and shall terminate with a non-zero exit  
113542 status.

113543 Otherwise, when an unrecoverable error is encountered it shall be equivalent to a SIGHUP  
113544 asynchronous event.

113545 Otherwise, when an error is encountered, the editor shall behave as specified in [Command](#)  
113546 [Descriptions in vi](#) (on page 3337).

#### 113547 **APPLICATION USAGE**

113548 None.

#### 113549 **EXAMPLES**

113550 None.

#### 113551 **RATIONALE**

113552 See the RATIONALE for [ex](#) for more information on *vi*. Major portions of the *vi* utility  
113553 specification point to *ex* to avoid inadvertent divergence. While *ex* and *vi* have historically been  
113554 implemented as a single utility, this is not required by POSIX.1-2008.

113555 It is recognized that portions of *vi* would be difficult, if not impossible, to implement  
113556 satisfactorily on a block-mode terminal, or a terminal without any form of cursor addressing,  
113557 thus it is not a mandatory requirement that such features should work on all terminals. It is the  
113558 intention, however, that a *vi* implementation should provide the full set of capabilities on all  
113559 terminals capable of supporting them.

113560 Historically, *vi* exited immediately if the standard input was not a terminal. POSIX.1-2008  
113561 permits, but does not require, this behavior. An end-of-file condition is not equivalent to an end-  
113562 of-file character. A common end-of-file character, <control>-D, is historically a *vi* command.

113563 The text in the STDOUT section reflects the usage of the verb *display* in this section; some  
113564 implementations of *vi* use standard output to write to the terminal, but POSIX.1-2008 does not  
113565 require that to be the case.

113566 Historically, implementations reverted to open mode if the terminal was incapable of supporting  
113567 full visual mode. POSIX.1-2008 requires this behavior. Historically, the open mode of *vi* behaved  
113568 roughly equivalently to the visual mode, with the exception that only a single line from the edit  
113569 buffer (one “buffer line”) was kept current at any time. This line was normally displayed on the  
113570 next-to-last line of a terminal with cursor addressing (and the last line performed its normal  
113571 visual functions for line-oriented commands and messages). In addition, some few commands  
113572 behaved differently in open mode than in visual mode. POSIX.1-2008 requires conformance to  
113573 historical practice.

Historically, *ex* and *vi* implementations have expected text to proceed in the usual European/Latin order of left to right, top to bottom. There is no requirement in POSIX.1-2008 that this be the case. The specification was deliberately written using words like “before”, “after”, “first”, and “last” in order to permit implementations to support the natural text order of the language.

Historically, lines past the end of the edit buffer were marked with single <tilde> ('~') characters; that is, if the one-based display was 20 lines in length, and the last line of the file was on line one, then lines 2-20 would contain only a single '~' character.

Historically, the *vi* editor attempted to display only complete lines at the bottom of the screen (it did display partial lines at the top of the screen). If a line was too long to fit in its entirety at the bottom of the screen, the screen lines where the line would have been displayed were displayed as single '@' characters, instead of displaying part of the line. POSIX.1-2008 permits, but does not require, this behavior. Implementations are encouraged to attempt always to display a complete line at the bottom of the screen when doing scrolling or screen positioning by buffer lines.

Historically, lines marked with '@' were also used to minimize output to dumb terminals over slow lines; that is, changes local to the cursor were updated, but changes to lines on the screen that were not close to the cursor were simply marked with an '@' sign instead of being updated to match the current text. POSIX.1-2008 permits, but does not require this feature because it is used ever less frequently as terminals become smarter and connections are faster.

### Initialization in *ex* and *vi*

Historically, *vi* always had a line in the edit buffer, even if the edit buffer was “empty”. For example:

1. The *ex* command = executed from visual mode wrote “1” when the buffer was empty.
2. Writes from visual mode of an empty edit buffer wrote files of a single character (a <newline>), while writes from *ex* mode of an empty edit buffer wrote empty files.
3. Put and read commands into an empty edit buffer left an empty line at the top of the edit buffer.

For consistency, POSIX.1-2008 does not permit any of these behaviors.

Historically, *vi* did not always return the terminal to its original modes; for example, ICRNL was modified if it was not originally set. POSIX.1-2008 does not permit this behavior.

### Command Descriptions in *vi*

Motion commands are among the most complicated aspects of *vi* to describe. With some exceptions, the text region and buffer type effect of a motion command on a *vi* command are described on a case-by-case basis. The descriptions of text regions in POSIX.1-2008 are not intended to imply direction; that is, an inclusive region from line *n* to line *n*+5 is identical to a region from line *n*+5 to line *n*. This is of more than academic interest—movements to marks can be in either direction, and, if the **wraps** option is set, so can movements to search points. Historically, lines are always stored into buffers in text order; that is, from the start of the edit buffer to the end. POSIX.1-2008 requires conformance to historical practice.

Historically, command counts were applied to any associated motion, and were multiplicative to any supplied motion count. For example, **2cw** is the same as **c2w**, and **2c3w** is the same as **c6w**. POSIX.1-2008 requires this behavior. Historically, *vi* commands that used bigwords, words, paragraphs, and sentences as objects treated groups of empty lines, or lines that contained only



<blank> characters, inconsistently. Some commands treated them as a single entity, while others treated each line separately. For example, the **w**, **W**, and **B** commands treated groups of empty lines as individual words; that is, the command would move the cursor to each new empty line. The **e** and **E** commands treated groups of empty lines as a single word; that is, the first use would move past the group of lines. The **b** command would just beep at the user, or if done from the start of the line as a motion command, fail in unexpected ways. If the lines contained only (or ended with) <blank> characters, the **w** and **W** commands would just beep at the user, the **E** and **e** commands would treat the group as a single word, and the **B** and **b** commands would treat the lines as individual words. For consistency and simplicity of specification, POSIX.1-2008 requires that all *vi* commands treat groups of empty or blank lines as a single entity, and that movement through lines ending with <blank> characters be consistent with other movements.

Historically, *vi* documentation indicated that any number of double-quotes were skipped after punctuation marks at sentence boundaries; however, implementations only skipped single-quotes. POSIX.1-2008 requires both to be skipped.

Historically, the first and last characters in the edit buffer were word boundaries. This historical practice is required by POSIX.1-2008.

Historically, *vi* attempted to update the minimum number of columns on the screen possible, which could lead to misleading information being displayed. POSIX.1-2008 makes no requirements other than that the current character being entered is displayed correctly, leaving all other decisions in this area up to the implementation.

Historically, lines were arbitrarily folded between columns of any characters that required multiple column positions on the screen, with the exception of tabs, which terminated at the right-hand margin. POSIX.1-2008 permits the former and requires the latter. Implementations that do not arbitrarily break lines between columns of characters that occupy multiple column positions should not permit the cursor to rest on a column that does not contain any part of a character.

The historical *vi* had a problem in that all movements were by buffer lines, not by display or screen lines. This is often the right thing to do; for example, single line movements, such as **j** or **k**, should work on buffer lines. Commands like **dj**, or **j.**, where **.** is a change command, only make sense for buffer lines. It is not, however, the right thing to do for screen motion or scrolling commands like <control>-D, <control>-F, and **H**. If the window is fairly small, using buffer lines in these cases can result in completely random motion; for example, **1<control>-D** can result in a completely changed screen, without any overlap. This is clearly not what the user wanted. The problem is even worse in the case of the **H**, **L**, and **M** commands—as they position the cursor at the first non-<blank> of the line, they may all refer to the same location in large lines, and will result in no movement at all.

In addition, if the line is larger than the screen, using buffer lines can make it impossible to display parts of the line—there are not any commands that do not display the beginning of the line in historical *vi*, and if both the beginning and end of the line cannot be on the screen at the same time, the user suffers. Finally, the page and half-page scrolling commands historically moved to the first non-<blank> in the new line. If the line is approximately the same size as the screen, this is inadequate because the cursor before and after a <control>-D command will refer to the same location on the screen.

Implementations of *ex* and *vi* exist that do not have these problems because the relevant commands (<control>-B, <control>-D, <control>-F, <control>-U, <control>-Y, <control>-E, **H**, **L**, and **M**) operate on display (screen) lines, not (edit) buffer lines.

POSIX.1-2008 does not permit this behavior by default because the standard developers believed that users would find it too confusing. However, historical practice has been relaxed. For

example, *ex* and *vi* historically attempted, albeit sometimes unsuccessfully, to never put part of a line on the last lines of a screen; for example, if a line would not fit in its entirety, no part of the line was displayed, and the screen lines corresponding to the line contained single '@' characters. This behavior is permitted, but not required by POSIX.1-2008, so that it is possible for implementations to support long lines in small screens more reasonably without changing the commands to be oriented to the display (instead of oriented to the buffer). POSIX.1-2008 also permits implementations to refuse to edit any edit buffer containing a line that will not fit on the screen in its entirety.

The display area (for example, the value of the **window** edit option) has historically been “grown”, or expanded, to display new text when local movements are done in displays where the number of lines displayed is less than the maximum possible. Expansion has historically been the first choice, when the target line is less than the maximum possible expansion value away. Scrolling has historically been the next choice, done when the target line is less than half a display away, and otherwise, the screen was redrawn. There were exceptions, however, in that *ex* commands generally always caused the screen to be redrawn. POSIX.1-2008 does not specify a standard behavior because there may be external issues, such as connection speed, the number of characters necessary to redraw as opposed to scroll, or terminal capabilities that implementations will have to accommodate.

The current line in POSIX.1-2008 maps one-to-one to a buffer line in the file. The current column does not. There are two different column values that are described by POSIX.1-2008. The first is the current column value as set by many of the *vi* commands. This value is remembered for the lifetime of the editor. The second column value is the actual position on the screen where the cursor rests. The two are not always the same. For example, when the cursor is backed by a multi-column character, the actual cursor position on the screen has historically been the last column of the character in command mode, and the first column of the character in input mode.

Commands that set the current line, but that do not set the current cursor value (for example, **j** and **k**) attempt to get as close as possible to the remembered column position, so that the cursor tends to restrict itself to a vertical column as the user moves around in the edit buffer. POSIX.1-2008 requires conformance to historical practice, requiring that the display location of the cursor on the display line be adjusted from the current column value as necessary to support this historical behavior.

Historically, only a single line (and for some terminals, a single line minus 1 column) of characters could be entered by the user for the line-oriented commands; that is, **:**, **!**, **/**, or **?**. POSIX.1-2008 permits, but does not require, this limitation.

Historically, “soft” errors in *vi* caused the terminal to be alerted, but no error message was displayed. As a general rule, no error message was displayed for errors in command execution in *vi*, when the error resulted from the user attempting an invalid or impossible action, or when a searched-for object was not found. Examples of soft errors included **h** at the left margin, **<control>-B** or **[** at the beginning of the file, **2G** at the end of the file, and so on. In addition, errors such as **%**, **]**, **]**, **)**, **N**, **n**, **f**, **F**, **t**, and **T** failing to find the searched-for object were soft as well. Less consistently, **/** and **?** displayed an error message if the pattern was not found, **/**, **?**, **N**, and **n** displayed an error message if no previous regular expression had been specified, and **;** did not display an error message if no previous **f**, **F**, **t**, or **T** command had occurred. Also, behavior in this area might reasonably be based on a runtime evaluation of the speed of a network connection. Finally, some implementations have provided error messages for soft errors in order to assist naive users, based on the value of a verbose edit option. POSIX.1-2008 does not list specific errors for which an error message shall be displayed. Implementations should conform to historical practice in the absence of any strong reason to diverge.

## Page Backwards

The <control>-B and <control>-F commands historically considered it an error to attempt to page past the beginning or end of the file, whereas the <control>-D and <control>-U commands simply moved to the beginning or end of the file. For consistency, POSIX.1-2008 requires the latter behavior for all four commands. All four commands still consider it an error if the current line is at the beginning (<control>-B, <control>-U) or end (<control>-F, <control>-D) of the file. Historically, the <control>-B and <control>-F commands skip two lines in order to include overlapping lines when a single command is entered. This makes less sense in the presence of a *count*, as there will be, by definition, no overlapping lines. The actual calculation used by historical implementations of the *vi* editor for <control>-B was:

```
((current first line) - count x (window edit option)) +2
```

and for <control>-F was:

```
((current first line) + count x (window edit option)) -2
```

This calculation does not work well when intermixing commands with and without counts; for example, **3<control>-F is not equivalent to entering the <control>-F command three times, and is not reversible by entering the <control>-B command three times. For consistency with other *vi* commands that take counts, POSIX.1-2008 requires a different calculation.**

## Scroll Forward

The 4BSD and System V implementations of *vi* differed on the initial value used by the **scroll** command. 4BSD used:

```
((window edit option) +1) /2
```

while System V used the value of the **scroll** edit option. The System V version is specified by POSIX.1-2008 because the standard developers believed that it was more intuitive and permitted the user a method of setting the scroll value initially without also setting the number of lines that are displayed.

## Scroll Forward by Line

Historically, the <control>-E and <control>-Y commands considered it an error if the last and first lines, respectively, were already on the screen. POSIX.1-2008 requires conformance to historical practice. Historically, the <control>-E and <control>-Y commands had no effect in open mode. For simplicity and consistency of specification, POSIX.1-2008 requires that they behave as usual, albeit with a single line screen.

## Clear and Redisplay

The historical <control>-L command refreshed the screen exactly as it was supposed to be currently displayed, replacing any '@' characters for lines that had been deleted but not updated on the screen with refreshed '@' characters. The intent of the <control>-L command is to refresh when the screen has been accidentally overwritten; for example, by a **write** command from another user, or modem noise.

## Redraw Screen

The historical <control>-R command redisplayed only when necessary to update lines that had been deleted but not updated on the screen and that were flagged with '@' characters. There is no requirement that the screen be in any way refreshed if no lines of this form are currently displayed. POSIX.1-2008 permits implementations to extend this command to refresh lines on the screen flagged with '@' characters because they are too long to be displayed in the current framework; however, the current line and column need not be modified.

## Search for tagstring

Historically, the first non-<blank> at or after the cursor was the first character, and all subsequent characters that were word characters, up to the end of the line, were included. For example, with the cursor on the leading <space> or on the '#' character in the text "#bar@", the tag was "#bar". On the character 'b' it was "bar", and on the 'a' it was "ar". POSIX.1-2008 requires this behavior.

## Replace Text with Results from Shell Command

Historically, the <, >, and ! commands considered most cursor motions other than line-oriented motions an error; for example, the command >/foo<CR> succeeded, while the command >I failed, even though the text region described by the two commands might be identical. For consistency, all three commands only consider entire lines and not partial lines, and the region is defined as any line that contains a character that was specified by the motion.

## Move to Matching Character

Other matching characters have been left implementation-defined in order to allow extensions such as matching '<' and '>' for searching HTML, or #ifdef, #else, and #endif for searching C source.

## Repeat Substitution

POSIX.1-2008 requires that any c and g flags specified to the previous substitute command be ignored; however, the r flag may still apply, if supported by the implementation.

## Return to Previous (Context or Section)

The [I, ]I, (, ), {, and } commands are all affected by "section boundaries", but in some historical implementations not all of the commands recognize the same section boundaries. This is a bug, not a feature, and a unique section-boundary algorithm was not described for each command. One special case that is preserved is that the sentence command moves to the end of the last line of the edit buffer while the other commands go to the beginning, in order to preserve the traditional character cut semantics of the sentence command. Historically, vi section boundaries at the beginning and end of the edit buffer were the first non-<blank> on the first and last lines of the edit buffer if one exists; otherwise, the last character of the first and last lines of the edit buffer if one exists. To increase consistency with other section locations, this has been simplified by POSIX.1-2008 to the first character of the first and last lines of the edit buffer, or the first and the last lines of the edit buffer if they are empty.

Sentence boundaries were problematic in the historical vi. They were not only the boundaries as defined for the section and paragraph commands, but they were the first non-<blank> that occurred after those boundaries, as well. Historically, the vi section commands were documented as taking an optional window size as a count preceding the command. This was not implemented in historical versions, so POSIX.1-2008 requires that the count repeat the command,

113794 for consistency with other *vi* commands.

### 113795 **Repeat**

113796 Historically, mapped commands other than text input commands could not be repeated using  
113797 the **period** command. POSIX.1-2008 requires conformance to historical practice.

113798 The restrictions on the interpretation of special characters (for example, <control>-H) in the  
113799 repetition of text input mode commands is intended to match historical practice. For example,  
113800 given the input sequence:

```
113801 iab<control>-H<control>-H<control>-Hdef<escape>
```

113802 the user should be informed of an error when the sequence is first entered, but not during a  
113803 command repetition. The character <control>-T is specifically exempted from this restriction.  
113804 Historical implementations of *vi* ignored <control>-T characters that were input in the original  
113805 command during command repetition. POSIX.1-2008 prohibits this behavior.

### 113806 **Find Regular Expression**

113807 Historically, commands did not affect the line searched to or from if the motion command was a  
113808 search (*/*, *?*, *N*, *n*) and the final position was the start/end of the line. There were some special  
113809 cases and *vi* was not consistent. POSIX.1-2008 does not permit this behavior, for consistency.  
113810 Historical implementations permitted but were unable to handle searches as motion commands  
113811 that wrapped (that is, due to the edit option **wraps**) to the original location. POSIX.1-2008  
113812 requires that this behavior be treated as an error.

113813 Historically, the syntax *"/RE/0"* was used to force the command to cut text in line mode.  
113814 POSIX.1-2008 requires conformance to historical practice.

113815 Historically, in open mode, a **z** specified to a search command redisplayed the current line  
113816 instead of displaying the current screen with the current line highlighted. For consistency and  
113817 simplicity of specification, POSIX.1-2008 does not permit this behavior.

113818 Historically, trailing **z** commands were permitted and ignored if entered as part of a search used  
113819 as a motion command. For consistency and simplicity of specification, POSIX.1-2008 does not  
113820 permit this behavior.

### 113821 **Execute an ex Command**

113822 Historically, *vi* implementations restricted the commands that could be entered on the colon  
113823 command line (for example, **append** and **change**), and some other commands were known to  
113824 cause them to fail catastrophically. For consistency, POSIX.1-2008 does not permit these  
113825 restrictions. When executing an *ex* command by entering *:*, it is not possible to enter a <newline>  
113826 as part of the command because it is considered the end of the command. A different approach  
113827 is to enter *ex* command mode by using the *vi* **Q** command (and later resuming visual mode with  
113828 the *ex* **vi** command). In *ex* command mode, the single-line limitation does not exist. So, for  
113829 example, the following is valid:

```
113830 Q
113831 s/break here/break\
113832 here/
113833 vi
```

113834 POSIX.1-2008 requires that, if the *ex* command overwrites any part of the screen that would be  
113835 erased by a refresh, *vi* pauses for a character from the user. Historically, this character could be  
113836 any character; for example, a character input by the user before the message appeared, or even a

mapped character. This is probably a bug, but implementations that have tried to be more rigorous by requiring that the user enter a specific character, or that the user enter a character after the message was displayed, have been forced by user indignation back into historical behavior. POSIX.1-2008 requires conformance to historical practice.

### Shift Left (Right)

Refer to the Rationale for the **!** and **/** commands. Historically, the **<** and **>** commands sometimes moved the cursor to the first non-**<blank>** (for example if the command was repeated or with **\_** as the motion command), and sometimes left it unchanged. POSIX.1-2008 does not permit this inconsistency, requiring instead that the cursor always move to the first non-**<blank>**. Historically, the **<** and **>** commands did not support buffer arguments, although some implementations allow the specification of an optional buffer. This behavior is neither required nor disallowed by POSIX.1-2008.

### Execute

Historically, buffers could execute other buffers, and loops, infinite and otherwise, were possible. POSIX.1-2008 requires conformance to historical practice. The *\*buffer* syntax of *ex* is not required in *vi*, because it is not historical practice and has been used in some *vi* implementations to support additional scripting languages.

### Reverse Case

Historically, the **~** command ignored any associated *count*, and acted only on the characters in the current line. For consistency with other *vi* commands, POSIX.1-2008 requires that an associated *count* act on the next *count* characters, and that the command move to subsequent lines if warranted by *count*, to make it possible to modify large pieces of text in a reasonably efficient manner. There exist *vi* implementations that optionally require an associated motion command for the **~** command. Implementations supporting this functionality are encouraged to base it on the **tildedop** edit option and handle the text regions and cursor positioning identically to the **yank** command.

### Append

Historically, *counts* specified to the **A**, **a**, **I**, and **i** commands repeated the input of the first line *count* times, and did not repeat the subsequent lines of the input text. POSIX.1-2008 requires that the entire text input be repeated *count* times.

### Move Backward to Preceding Word

Historically, *vi* became confused if word commands were used as motion commands in empty files. POSIX.1-2008 requires that this be an error. Historical implementations of *vi* had a large number of bugs in the word movement commands, and they varied greatly in behavior in the presence of empty lines, “words” made up of a single character, and lines containing only **<blank>** characters. For consistency and simplicity of specification, POSIX.1-2008 does not permit this behavior.

### Change to End-of-Line

Some historical implementations of the **C** command did not behave as described by POSIX.1-2008 when the **\$** key was remapped because they were implemented by pushing the **\$** key onto the input queue and reprocessing it. POSIX.1-2008 does not permit this behavior. Historically, the **C**, **S**, and **s** commands did not copy replaced text into the numeric buffers. For consistency and simplicity of specification, POSIX.1-2008 requires that they behave like their respective **c** commands in all respects.

### Delete

Historically, lines in open mode that were deleted were scrolled up, and an **@** glyph written over the beginning of the line. In the case of terminals that are incapable of the necessary cursor motions, the editor erased the deleted line from the screen. POSIX.1-2008 requires conformance to historical practice; that is, if the terminal cannot display the '**@**' character, the line cannot remain on the screen.

### Delete to End-of-Line

Some historical implementations of the **D** command did not behave as described by POSIX.1-2008 when the **\$** key was remapped because they were implemented by pushing the **\$** key onto the input queue and reprocessing it. POSIX.1-2008 does not permit this behavior.

### Join

An historical oddity of *vi* is that the commands **J**, **1J**, and **2J** are all equivalent. POSIX.1-2008 requires conformance to historical practice. The *vi* **J** command is specified in terms of the *ex* **join** command with an *ex* command *count* value. The address correction for a *count* that is past the end of the edit buffer is necessary for historical compatibility for both *ex* and *vi*.

### Mark Position

Historical practice is that only lowercase letters, plus backquote and single-quote, could be used to mark a cursor position. POSIX.1-2008 requires conformance to historical practice, but encourages implementations to support other characters as marks as well.

### Repeat Regular Expression Find (Forward and Reverse)

Historically, the **N** and **n** commands could not be used as motion components for the **c** command. With the exception of the **cN** command, which worked if the search crossed a line boundary, the text region would be discarded, and the user would not be in text input mode. For consistency and simplicity of specification, POSIX.1-2008 does not permit this behavior.

### Insert Empty Line (Below and Above)

Historically, counts to the **O** and **o** commands were used as the number of physical lines to open, if the terminal was dumb and the **slowopen** option was not set. This was intended to minimize traffic over slow connections and repainting for dumb terminals. POSIX.1-2008 does not permit this behavior, requiring that a *count* to the open command behave as for other text input commands. This change to historical practice was made for consistency, and because a superset of the functionality is provided by the **slowopen** edit option.

### Put from Buffer (Following and Before)

Historically, *counts* to the **p** and **P** commands were ignored if the buffer was a line mode buffer, but were (mostly) implemented as described in POSIX.1-2008 if the buffer was a character mode buffer. Because implementations exist that do not have this limitation, and because pasting lines multiple times is generally useful, POSIX.1-2008 requires that *count* be supported for all **p** and **P** commands.

Historical implementations of *vi* were widely known to have major problems in the **p** and **P** commands, particularly when unusual regions of text were copied into the edit buffer. The standard developers viewed these as bugs, and they are not permitted for consistency and simplicity of specification.

Historically, a **P** or **p** command (or an *ex* **put** command executed from open or visual mode) executed in an empty file, left an empty line as the first line of the file. For consistency and simplicity of specification, POSIX.1-2008 does not permit this behavior.

### Replace Character

Historically, the **r** command did not correctly handle the *erase* and *word erase* characters as arguments, nor did it handle an associated *count* greater than 1 with a <carriage-return> argument, for which it replaced *count* characters with a single <newline>. POSIX.1-2008 does not permit these inconsistencies.

Historically, the **r** command permitted the <control>-V escaping of entered characters, such as <ESC> and the <carriage-return>; however, it required two leading <control>-V characters instead of one. POSIX.1-2008 requires that this be changed for consistency with the other text input commands of *vi*.

Historically, it is an error to enter the **r** command if there are less than *count* characters at or after the cursor in the line. While a reasonable and unambiguous extension would be to permit the **r** command on empty lines, it would require that too large a *count* be adjusted to match the number of characters at or after the cursor for consistency, which is sufficiently different from historical practice to be avoided. POSIX.1-2008 requires conformance to historical practice.

### Replace Characters

Historically, if there were **autoindent** characters in the line on which the **R** command was run, and **autoindent** was set, the first <newline> would be properly indented and no characters would be replaced by the <newline>. Each additional <newline> would replace *n* characters, where *n* was the number of characters that were needed to indent the rest of the line to the proper indentation level. This behavior is a bug and is not permitted by POSIX.1-2008.

### Undo

Historical practice for cursor positioning after undoing commands was mixed. In most cases, when undoing commands that affected a single line, the cursor was moved to the start of added or changed text, or immediately after deleted text. However, if the user had moved from the line being changed, the column was either set to the first non-<blank>, returned to the origin of the command, or remained unchanged. When undoing commands that affected multiple lines or entire lines, the cursor was moved to the first character in the first line restored. As an example of how inconsistent this was, a search, followed by an **o** text input command, followed by an **undo** would return the cursor to the location where the **o** command was entered, but a **cw** command followed by an **o** command followed by an **undo** would return the cursor to the first non-<blank> of the line. POSIX.1-2008 requires the most useful of these behaviors, and discards the least useful, in the interest of consistency and simplicity of specification.



## Yank

Historically, the **yank** command did not move to the end of the motion if the motion was in the forward direction. It moved to the end of the motion if the motion was in the backward direction, except for the **\_** command, or for the **G** and **'** commands when the end of the motion was on the current line. This was further complicated by the fact that for a number of motion commands, the **yank** command moved the cursor but did not update the screen; for example, a subsequent command would move the cursor from the end of the motion, even though the cursor on the screen had not reflected the cursor movement for the **yank** command. POSIX.1-2008 requires that all **yank** commands associated with backward motions move the cursor to the end of the motion for consistency, and specifically, to make **'** commands as motions consistent with search patterns as motions.

## Yank Current Line

Some historical implementations of the **Y** command did not behave as described by POSIX.1-2008 when the **' \_ '** key was remapped because they were implemented by pushing the **' \_ '** key onto the input queue and reprocessing it. POSIX.1-2008 does not permit this behavior.

## Redraw Window

Historically, the **z** command always redrew the screen. This is permitted but not required by POSIX.1-2008, because of the frequent use of the **z** command in macros such as **map n nz.** for screen positioning, instead of its use to change the screen size. The standard developers believed that expanding or scrolling the screen offered a better interface for users. The ability to redraw the screen is preserved if the optional new window size is specified, and in the **<control>-L** and **<control>-R** commands.

The semantics of **z^** are confusing at best. Historical practice is that the screen before the screen that ended with the specified line is displayed. POSIX.1-2008 requires conformance to historical practice.

Historically, the **z** command would not display a partial line at the top or bottom of the screen. If the partial line would normally have been displayed at the bottom of the screen, the command worked, but the partial line was replaced with **' @ '** characters. If the partial line would normally have been displayed at the top of the screen, the command would fail. For consistency and simplicity of specification, POSIX.1-2008 does not permit this behavior.

Historically, the **z** command with a line specification of 1 ignored the command. For consistency and simplicity of specification, POSIX.1-2008 does not permit this behavior.

Historically, the **z** command did not set the cursor column to the first non-**<blank>** for the character if the first screen was to be displayed, and was already displayed. For consistency and simplicity of specification, POSIX.1-2008 does not permit this behavior.

## Input Mode Commands in vi

Historical implementations of **vi** did not permit the user to erase more than a single line of input, or to use normal erase characters such as *line erase*, *worderase*, and *erase* to erase **autoindent** characters. As there exist implementations of **vi** that do not have these limitations, both behaviors are permitted, but only historical practice is required. In the case of these extensions, **vi** is required to pause at the **autoindent** and previous line boundaries.

Historical implementations of **vi** updated only the portion of the screen where the current cursor character was displayed. For example, consider the **vi** input keystrokes:

```
iabcd<escape>0C<tab>
```

Historically, the `<tab>` would overwrite the characters "abcd" when it was displayed. Other implementations replace only the 'a' character with the `<tab>`, and then push the rest of the characters ahead of the cursor. Both implementations have problems. The historical implementation is probably visually nicer for the above example; however, for the keystrokes:

```
iabcd<ESC>OR<tab><ESC>
```

the historical implementation results in the string "bcd" disappearing and then magically reappearing when the `<ESC>` character is entered. POSIX.1-2008 requires the former behavior when overwriting erase-columns—that is, overwriting characters that are no longer logically part of the edit buffer—and the latter behavior otherwise.

Historical implementations of *vi* discarded the `<control>-D` and `<control>-T` characters when they were entered at places where their command functionality was not appropriate. POSIX.1-2008 requires that the `<control>-T` functionality always be available, and that `<control>-D` be treated as any other key when not operating on **autoindent** characters.

## NUL

Some historical implementations of *vi* limited the number of characters entered using the NUL input character to 256 bytes. POSIX.1-2008 permits this limitation; however, implementations are encouraged to remove this limit.

## `<control>-D`

See also Rationale for the input mode command `<newline>`. The hidden assumptions in the `<control>-D` command (and in the *vi* **autoindent** specification in general) is that `<space>` characters take up a single column on the screen and that `<tab>` characters are comprised of an integral number of `<space>` characters.

## `<newline>`

Implementations are permitted to rewrite **autoindent** characters in the line when `<newline>`, `<carriage-return>`, `<control>-D`, and `<control>-T` are entered, or when the **shift** commands are used, because historical implementations have both done so and found it necessary to do so. For example, a `<control>-D` when the cursor is preceded by a single `<tab>`, with **tabstop** set to 8, and **shiftwidth** set to 3, will result in the `<tab>` being replaced by several `<space>` characters.

## `<control>-T`

See also the Rationale for the input mode command `<newline>`. Historically, `<control>-T` only worked if no non-`<blank>` characters had yet been input in the current input line. In addition, the characters inserted by `<control>-T` were treated as **autoindent** characters, and could not be erased using normal user erase characters. Because implementations exist that do not have these limitations, and as moving to a column boundary is generally useful, POSIX.1-2008 requires that both limitations be removed.

**<control>-V**

Historically, *vi* used **^V**, regardless of the value of the literal-next character of the terminal. POSIX.1-2008 requires conformance to historical practice.

The uses described for <control>-V can also be accomplished with <control>-Q, which is useful on terminals that use <control>-V for the down-arrow function. However, most historical implementations use <control>-Q for the *termios* START character, so the editor will generally not receive the <control>-Q unless **stty ixon** mode is set to off. (In addition, some historical implementations of *vi* explicitly set **ixon** mode to on, so it was difficult for the user to set it to off.) Any of the command characters described in POSIX.1-2008 can be made ineffective by their selection as *termios* control characters, using the *stty* utility or other methods described in the System Interfaces volume of POSIX.1-2008.

**<ESC>**

Historically, SIGINT alerted the terminal when used to end input mode. This behavior is permitted, but not required, by POSIX.1-2008.

**FUTURE DIRECTIONS**

None.

**SEE ALSO**

*ed*, *ex*, *stty*

XBD [Section 12.2](#) (on page 215)

**CHANGE HISTORY**

First released in Issue 2.

**Issue 5**

The FUTURE DIRECTIONS section is added.

**Issue 6**

This utility is marked as part of the User Portability Utilities option.

The APPLICATION USAGE section is added.

The obsolescent SYNOPSIS is removed.

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The **reindent** command description is added.

The *vi* utility has been extensively rewritten for alignment with the IEEE P1003.2b draft standard.

IEEE PASC Interpretations 1003.2 #57, #62, #63, #64, #78, and #188 are applied.

IEEE PASC Interpretation 1003.2 #207 is applied, clarifying the description of the **R** command in a manner similar to the descriptions of other text input mode commands such as **i**, **o**, and **O**.

The **-l** option is removed.

IEEE Std 1003.1-2001/Cor 1-2002, item XCU/TC1/D6/41 is applied, adding *[count]* to the Synopsis for **[[**.

IEEE Std 1003.1-2001/Cor 1-2002, item XCU/TC1/D6/42 is applied, adding *[count]* to the Synopsis for **]]**.

114076 **Issue 7**

114077 Austin Group Interpretation 1003.1-2001 #027 is applied, clarifying that '+' may be recognized  
114078 as an option delimiter in the OPTIONS section.

114079 Austin Group Interpretation 1003.1-2001 #087 is applied, updating the Put from Buffer Before (P)  
114080 command description to address multi-line requirements.

114081 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

114082 **NAME**

114083       wait — await process completion

114084 **SYNOPSIS**114085       wait [*pid*...]114086 **DESCRIPTION**

114087       When an asynchronous list (see [Section 2.9.3.1](#), on page 2342) is started by the shell, the process  
 114088       ID of the last command in each element of the asynchronous list shall become known in the  
 114089       current shell execution environment; see [Section 2.12](#) (on page 2353).

114090       If the *wait* utility is invoked with no operands, it shall wait until all process IDs known to the  
 114091       invoking shell have terminated and exit with a zero exit status.

114092       If one or more *pid* operands are specified that represent known process IDs, the *wait* utility shall  
 114093       wait until all of them have terminated. If one or more *pid* operands are specified that represent  
 114094       unknown process IDs, *wait* shall treat them as if they were known process IDs that exited with  
 114095       exit status 127. The exit status returned by the *wait* utility shall be the exit status of the process  
 114096       requested by the last *pid* operand.

114097       The known process IDs are applicable only for invocations of *wait* in the current shell execution  
 114098       environment.

114099 **OPTIONS**

114100       None.

114101 **OPERANDS**

114102       The following operand shall be supported:

114103       *pid*           One of the following:

- 114104           1. The unsigned decimal integer process ID of a command, for which the  
 114105           utility is to wait for the termination.
- 114106           2. A job control job ID (see XBD [Section 3.204](#), on page 65) that identifies a  
 114107           background process group to be waited for. The job control job ID notation  
 114108           is applicable only for invocations of *wait* in the current shell execution  
 114109           environment; see [Section 2.12](#) (on page 2353). The exit status of *wait* shall be  
 114110           determined by the last command in the pipeline.

114111       **Note:**       The job control job ID type of *pid* is only available on systems supporting  
 114112       the User Portability Utilities option.

114113 **STDIN**

114114       Not used.

114115 **INPUT FILES**

114116       None.

114117 **ENVIRONMENT VARIABLES**114118       The following environment variables shall affect the execution of *wait*:

114119       *LANG*       Provide a default value for the internationalization variables that are unset or null.  
 114120       (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 114121       variables used to determine the values of locale categories.)

114122       *LC\_ALL*      If set to a non-empty string value, override the values of all the other  
 114123       internationalization variables.

114124 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
 114125 characters (for example, single-byte as opposed to multi-byte characters in  
 114126 arguments).

114127 *LC\_MESSAGES*  
 114128 Determine the locale that should be used to affect the format and contents of  
 114129 diagnostic messages written to standard error.

114130 XSI *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

114131 **ASYNCHRONOUS EVENTS**  
 114132 Default.

114133 **STDOUT**  
 114134 Not used.

114135 **STDERR**  
 114136 The standard error shall be used only for diagnostic messages.

114137 **OUTPUT FILES**  
 114138 None.

114139 **EXTENDED DESCRIPTION**  
 114140 None.

114141 **EXIT STATUS**  
 114142 If one or more operands were specified, all of them have terminated or were not known by the  
 114143 invoking shell, and the status of the last operand specified is known, then the exit status of *wait*  
 114144 shall be the exit status information of the command indicated by the last operand specified. If  
 114145 the process terminated abnormally due to the receipt of a signal, the exit status shall be greater  
 114146 than 128 and shall be distinct from the exit status generated by other signals, but the exact value  
 114147 is unspecified. (See the *kill -l* option.) Otherwise, the *wait* utility shall exit with one of the  
 114148 following values:

114149 0 The *wait* utility was invoked with no operands and all process IDs known by the  
 114150 invoking shell have terminated.

114151 1-126 The *wait* utility detected an error.

114152 127 The command identified by the last *pid* operand specified is unknown.

114153 **CONSEQUENCES OF ERRORS**  
 114154 Default.

114155 **APPLICATION USAGE**  
 114156 On most implementations, *wait* is a shell built-in. If it is called in a subshell or separate utility  
 114157 execution environment, such as one of the following:

114158 (wait)  
 114159 nohup wait ...  
 114160 find . -exec wait ... \;

114161 it returns immediately because there are no known process IDs to wait for in those  
 114162 environments.

114163 Historical implementations of interactive shells have discarded the exit status of terminated  
 114164 background processes before each shell prompt. Therefore, the status of background processes  
 114165 was usually lost unless it terminated while *wait* was waiting for it. This could be a serious  
 114166 problem when a job that was expected to run for a long time actually terminated quickly with a  
 114167 syntax or initialization error because the exit status returned was usually zero if the requested

process ID was not found. This volume of POSIX.1-2008 requires the implementation to keep the status of terminated jobs available until the status is requested, so that scripts like:

```
j1&
p1=$!
j2&
wait $p1
echo Job 1 exited with status $?
wait $!
echo Job 2 exited with status $?
```

work without losing status on any of the jobs. The shell is allowed to discard the status of any process if it determines that the application cannot get the process ID for that process from the shell. It is also required to remember only {CHILD\_MAX} number of processes in this way. Since the only way to get the process ID from the shell is by using the '!' shell parameter, the shell is allowed to discard the status of an asynchronous list if "\$!" was not referenced before another asynchronous list was started. (This means that the shell only has to keep the status of the last asynchronous list started if the application did not reference "\$!". If the implementation of the shell is smart enough to determine that a reference to "\$!" was not saved anywhere that the application can retrieve it later, it can use this information to trim the list of saved information. Note also that a successful call to *wait* with no operands discards the exit status of all asynchronous lists.)

If the exit status of *wait* is greater than 128, there is no way for the application to know if the waited-for process exited with that value or was killed by a signal. Since most utilities exit with small values, there is seldom any ambiguity. Even in the ambiguous cases, most applications just need to know that the asynchronous job failed; it does not matter whether it detected an error and failed or was killed and did not complete its job normally.

#### EXAMPLES

Although the exact value used when a process is terminated by a signal is unspecified, if it is known that a signal terminated a process, a script can still reliably determine which signal by using *kill* as shown by the following script:

```
sleep 1000&
pid=$!
kill -kill $pid
wait $pid
echo $pid was terminated by a SIG$(kill -l $?) signal.
```

If the following sequence of commands is run in less than 31 seconds:

```
sleep 257 | sleep 31 &
jobs -l %%
```

either of the following commands returns the exit status of the second *sleep* in the pipeline:

```
wait <pid of sleep 31>
wait %%
```

#### RATIONALE

The description of *wait* does not refer to the *waitpid()* function from the System Interfaces volume of POSIX.1-2008 because that would needlessly overspecify this interface. However, the wording means that *wait* is required to wait for an explicit process when it is given an argument so that the status information of other processes is not consumed. Historical implementations use the *wait()* function defined in the System Interfaces volume of POSIX.1-2008 until *wait()* returns the requested process ID or finds that the requested process does not exist. Because this

114215 means that a shell script could not reliably get the status of all background children if a second  
114216 background job was ever started before the first job finished, it is recommended that the *wait*  
114217 utility use a method such as the functionality provided by the *waitpid()* function.

114218 The ability to wait for multiple *pid* operands was adopted from the KornShell.

114219 This new functionality was added because it is needed to determine the exit status of any  
114220 asynchronous list accurately. The only compatibility problem that this change creates is for a  
114221 script like

```
114222 while sleep 60 do
114223 job& echo Job started $(date) as $! done
```

114224 which causes the shell to monitor all of the jobs started until the script terminates or runs out of  
114225 memory. This would not be a problem if the loop did not reference "\$!" or if the script would  
114226 occasionally *wait* for jobs it started.

#### 114227 FUTURE DIRECTIONS

114228 None.

#### 114229 SEE ALSO

114230 [Chapter 2](#) (on page 2319), *kill*, *sh*

114231 XBD [Section 3.204](#) (on page 65), [Chapter 8](#) (on page 173)

114232 XSH *wait()*

#### 114233 CHANGE HISTORY

114234 First released in Issue 2.



114235 **NAME**

114236       wc — word, line, and byte or character count

114237 **SYNOPSIS**114238       wc [-c|-m] [-lw] [*file*...]114239 **DESCRIPTION**114240       The *wc* utility shall read one or more input files and, by default, write the number of <newline>  
114241       characters, words, and bytes contained in each input file to the standard output.114242       The utility also shall write a total count for all named files, if more than one input file is  
114243       specified.114244       The *wc* utility shall consider a *word* to be a non-zero-length string of characters delimited by  
114245       white space.114246 **OPTIONS**114247       The *wc* utility shall conform to XBD [Section 12.2](#) (on page 215).

114248       The following options shall be supported:

114249       -c           Write to the standard output the number of bytes in each input file.

114250       -l           Write to the standard output the number of <newline> characters in each input  
114251       file.

114252       -m           Write to the standard output the number of characters in each input file.

114253       -w           Write to the standard output the number of words in each input file.

114254       When any option is specified, *wc* shall report only the information requested by the specified  
114255       options.114256 **OPERANDS**

114257       The following operand shall be supported:

114258       *file*        A pathname of an input file. If no *file* operands are specified, the standard input  
114259       shall be used.114260 **STDIN**114261       The standard input shall be used if no *file* operands are specified, and shall be used if a *file*  
114262       operand is '-' and the implementation treats the '-' as meaning standard input. Otherwise,  
114263       the standard input shall not be used. See the INPUT FILES section.114264 **INPUT FILES**

114265       The input files may be of any type.

114266 **ENVIRONMENT VARIABLES**114267       The following environment variables shall affect the execution of *wc*:114268       LANG        Provide a default value for the internationalization variables that are unset or null.  
114269                   (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
114270                   variables used to determine the values of locale categories.)114271       LC\_ALL      If set to a non-empty string value, override the values of all the other  
114272       internationalization variables.114273       LC\_CTYPE    Determine the locale for the interpretation of sequences of bytes of text data as  
114274       characters (for example, single-byte as opposed to multi-byte characters in  
114275       arguments and input files) and which characters are defined as white-space  
114276       characters.

- 114277 *LC\_MESSAGES*  
 114278 Determine the locale that should be used to affect the format and contents of  
 114279 diagnostic messages written to standard error and informative messages written to  
 114280 standard output.
- 114281 XSI *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.
- 114282 **ASYNCHRONOUS EVENTS**  
 114283 Default.
- 114284 **STDOUT**  
 114285 By default, the standard output shall contain an entry for each input file of the form:  
 114286 "%d %d %d %s\n", <newlines>, <words>, <bytes>, <file>  
 114287 If the **-m** option is specified, the number of characters shall replace the <bytes> field in this  
 114288 format.  
 114289 If any options are specified and the **-l** option is not specified, the number of <newline>  
 114290 characters shall not be written.  
 114291 If any options are specified and the **-w** option is not specified, the number of words shall not be  
 114292 written.  
 114293 If any options are specified and neither **-c** nor **-m** is specified, the number of bytes or characters  
 114294 shall not be written.  
 114295 If no input *file* operands are specified, no name shall be written and no <blank> characters  
 114296 preceding the pathname shall be written.  
 114297 If more than one input *file* operand is specified, an additional line shall be written, of the same  
 114298 format as the other lines, except that the word **total** (in the POSIX locale) shall be written instead  
 114299 of a pathname and the total of each column shall be written as appropriate. Such an additional  
 114300 line, if any, is written at the end of the output.
- 114301 **STDERR**  
 114302 The standard error shall be used only for diagnostic messages.
- 114303 **OUTPUT FILES**  
 114304 None.
- 114305 **EXTENDED DESCRIPTION**  
 114306 None.
- 114307 **EXIT STATUS**  
 114308 The following exit values shall be returned:  
 114309 0 Successful completion.  
 114310 >0 An error occurred.
- 114311 **CONSEQUENCES OF ERRORS**  
 114312 Default.

**114313 APPLICATION USAGE**

114314 The **-m** option is not a switch, but an option at the same level as **-c**. Thus, to produce the full  
114315 default output with character counts instead of bytes, the command required is:

114316 `wc -mlw`

**114317 EXAMPLES**

114318 None.

**114319 RATIONALE**

114320 The output file format *pseudo-printf()* string differs from the System V version of *wc*:

114321 `"%7d%7d%7d %s\n"`

114322 which produces possibly ambiguous and unparseable results for very large files, as it assumes no  
114323 number shall exceed six digits.

114324 Some historical implementations use only `<space>`, `<tab>`, and `<newline>` as word separators.  
114325 The equivalent of the ISO C standard *isspace()* function is more appropriate.

114326 The **-c** option stands for “character” count, even though it counts bytes. This stems from the  
114327 sometimes erroneous historical view that bytes and characters are the same size. Due to  
114328 international requirements, the **-m** option (reminiscent of “multi-byte”) was added to obtain  
114329 actual character counts.

114330 Early proposals only specified the results when input files were text files. The current  
114331 specification more closely matches historical practice. (Bytes, words, and `<newline>` characters  
114332 are counted separately and the results are written when an end-of-file is detected.)

114333 Historical implementations of the *wc* utility only accepted one argument to specify the options  
114334 **-c**, **-l**, and **-w**. Some of them also had multiple occurrences of an option cause the  
114335 corresponding count to be written multiple times and had the order of specification of the  
114336 options affect the order of the fields on output, but did not document either of these. Because  
114337 common usage either specifies no options or only one option, and because none of this was  
114338 documented, the changes required by this volume of POSIX.1-2008 should not break many  
114339 historical applications (and do not break any historical conforming applications).

**114340 FUTURE DIRECTIONS**

114341 None.

**114342 SEE ALSO**

114343 [\*cksum\*](#)

114344 XBD [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 215)

**114345 CHANGE HISTORY**

114346 First released in Issue 2.

**114347 Issue 7**

114348 Austin Group Interpretation 1003.1-2001 #092 is applied.

114349 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

114350 **NAME**114351        what — identify SCCS files (**DEVELOPMENT**)114352 **SYNOPSIS**114353 XSI        `what [-s] file...`114354 **DESCRIPTION**

114355        The *what* utility shall search the given files for all occurrences of the pattern that *get* (see *get*)  
 114356        substitutes for the %Z% keyword ("@(#)") and shall write to standard output what follows  
 114357        until the first occurrence of one of the following:

114358        "    &gt;    newline    \    NUL

114359 **OPTIONS**114360        The *what* utility shall conform to XBD [Section 12.2](#) (on page 215).

114361        The following option shall be supported:

114362        -s            Quit after finding the first occurrence of the pattern in each file.

114363 **OPERANDS**

114364        The following operands shall be supported:

114365        *file*            A pathname of a file to search.114366 **STDIN**

114367        Not used.

114368 **INPUT FILES**

114369        The input files shall be of any file type.

114370 **ENVIRONMENT VARIABLES**114371        The following environment variables shall affect the execution of *what*:

114372        *LANG*            Provide a default value for the internationalization variables that are unset or null.  
 114373                            (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 114374                            variables used to determine the values of locale categories.)

114375        *LC\_ALL*          If set to a non-empty string value, override the values of all the other  
 114376                            internationalization variables.

114377        *LC\_CTYPE*        Determine the locale for the interpretation of sequences of bytes of text data as  
 114378                            characters (for example, single-byte as opposed to multi-byte characters in  
 114379                            arguments and input files).

114380        *LC\_MESSAGES*

114381                            Determine the locale that should be used to affect the format and contents of  
 114382                            diagnostic messages written to standard error.

114383        *NLSPATH*        Determine the location of message catalogs for the processing of *LC\_MESSAGES*.114384 **ASYNCHRONOUS EVENTS**

114385        Default.

114386 **STDOUT**114387        The standard output shall consist of the following for each *file* operand:

114388        "%s:\n\t%s\n", &lt;pathname&gt;, &lt;identification string&gt;

**114389 STDERR**

114390 The standard error shall be used only for diagnostic messages.

**114391 OUTPUT FILES**

114392 None.

**114393 EXTENDED DESCRIPTION**

114394 None.

**114395 EXIT STATUS**

114396 The following exit values shall be returned:

114397 0 Any matches were found.

114398 1 Otherwise.

**114399 CONSEQUENCES OF ERRORS**

114400 Default.

**114401 APPLICATION USAGE**

114402 The *what* utility is intended to be used in conjunction with the SCCS command *get*, which  
114403 automatically inserts identifying information, but it can also be used where the information is  
114404 inserted by any other means.

114405 When the string "@ (#) " is included in a library routine in a shared library, it might not be found  
114406 in an **a.out** file using that library routine.

**114407 EXAMPLES**

114408 If the C-language program in file **f.c** contains:

114409 `char ident[] = "@(#)identification information";`

114410 and **f.c** is compiled to yield **f.o** and **a.out**, then the command:

114411 `what f.c f.o a.out`

114412 writes:

114413 **f.c:**  
114414     identification information  
114415     ...  
114416 **f.o:**  
114417     identification information  
114418     ...  
114419 **a.out:**  
114420     identification information  
114421     ...

**114422 RATIONALE**

114423 None.

**114424 FUTURE DIRECTIONS**

114425 None.

**114426 SEE ALSO**

114427 *get*

114428 XBD [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 215)



114429 **CHANGE HISTORY**  
114430 First released in Issue 2.



114431 NAME

114432            who — display who is on the system

114433 **SYNOPSIS**

```
114434 XSI who [-mTu] [-abdHlprt] [file]
```

```
114435 XSI who [-mu] -s [-bHlprt] [file]
```

```
114436 who -q [file]
```

114437 who am i

|        |          |
|--------|----------|
| 114438 | who am I |
|--------|----------|

114439 DESCRIPTION

114440 The *who* utility shall list various pieces of information about accessible users. The domain of  
114441 accessibility is implementation-defined.

```
114442 XSI Based on the options given, who can also list the user's name, terminal line, login time, elapsed
114443 time since activity occurred on the line, and the process ID of the command interpreter for each
114444 current system user.
```

114445 **OPTIONS**

114446 The *who* utility shall conform to XBD [Section 12.2](#) (on page 215).

114447 The following options shall be supported. The metavariables, such as *<line>*, refer to fields  
114448 described in the STDOUT section.

|        |     |                                                                    |                                                                                                                    |
|--------|-----|--------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------|
| 114449 | XSI | <b>-a</b>                                                          | Process the implementation-defined database or named file with the <b>-b</b> , <b>-d</b> , <b>-l</b> , <b>-p</b> , |
| 114450 |     | <b>-r</b> , <b>-t</b> , <b>-T</b> and <b>-u</b> options turned on. |                                                                                                                    |

|        |     |           |                                                                                  |
|--------|-----|-----------|----------------------------------------------------------------------------------|
| 114451 | XSI | <b>-b</b> | Write the time and date of the last system reboot. The system reboot time is the |
| 114452 |     |           | time at which the implementation is able to commence running processes.          |

114453 XSI **-d** Write a list of all processes that have expired and not been respawned by the *init*  
114454 system process. The *<exit>* field shall appear for dead processes and contain the  
114455 termination and exit values of the dead process. This can be useful in determining  
114456 why a process terminated.

|        |     |    |                                                 |
|--------|-----|----|-------------------------------------------------|
| 114457 | XSI | -H | Write column headings above the regular output. |
|--------|-----|----|-------------------------------------------------|

|        |     |           |                                                                                                                                                                                                                                                                               |
|--------|-----|-----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 114458 | XSI | <b>-1</b> | (The letter ell.) List only those lines on which the system is waiting for someone to login. The <i>&lt;name&gt;</i> field shall be <b>LOGIN</b> in such cases. Other fields shall be the same as for user entries except that the <i>&lt;state&gt;</i> field does not exist. |
| 114459 |     |           |                                                                                                                                                                                                                                                                               |
| 114460 |     |           |                                                                                                                                                                                                                                                                               |

```
114461 -m Output only information about the current terminal.
```

```
114462 XSI -p List any other process that is currently active and has been previously spawned by
114463 init.
```

|        |     |           |                                                                                |
|--------|-----|-----------|--------------------------------------------------------------------------------|
| 114464 | XSI | <b>-q</b> | (Quick.) List only the names and the number of users currently logged on. When |
| 114465 |     |           | this option is used, all other options shall be ignored.                       |

```
114466 XSI -r Write the current run-level of the init process.
```

|            |                                                                                                                                                                                                                      |
|------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 114467 XSI | <div style="background-color: #f0f0f0; padding: 2px 5px; display: inline-block;"><b>-s</b></div> List only the <i>&lt;name&gt;</i> , <i>&lt;line&gt;</i> , and <i>&lt;time&gt;</i> fields. This is the default case. |
|------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

|        |     |           |                                               |
|--------|-----|-----------|-----------------------------------------------|
| 114468 | XSI | <b>-t</b> | Indicate the last change to the system clock. |
|--------|-----|-----------|-----------------------------------------------|

```
114469 -T Show the state of each terminal, as described in the STDOUT section.
```

114470           **-u**           Write “idle time” for each displayed user in addition to any other information. The  
 114471 idle time is the time since any activity occurred on the user’s terminal. The method  
 114472 XSI of determining this is unspecified. This option shall list only those users who are  
 114473 currently logged in. The *<name>* is the user’s login name. The *<line>* is the name  
 114474 of the line as found in the directory */dev*. The *<time>* is the time that the user  
 114475 logged in. The *<activity>* is the number of hours and minutes since activity last  
 114476 occurred on that particular line. A dot indicates that the terminal has seen activity  
 114477 in the last minute and is therefore “current”. If more than twenty-four hours have  
 114478 elapsed or the line has not been used since boot time, the entry shall be marked  
 114479 *<old>*. This field is useful when trying to determine whether a person is working at  
 114480 the terminal or not. The *<pid>* is the process ID of the user’s login process.

#### 114481 OPERANDS

114482 XSI           The following operands shall be supported:

114483           **am i, am I**       In the POSIX locale, limit the output to describing the invoking user, equivalent to  
 114484 the **-m** option. The **am** and **i** or **I** must be separate arguments.

114485           *file*           Specify a pathname of a file to substitute for the implementation-defined database  
 114486 of logged-on users that *who* uses by default.

#### 114487 STDIN

114488           Not used.

#### 114489 INPUT FILES

114490           None.

#### 114491 ENVIRONMENT VARIABLES

114492           The following environment variables shall affect the execution of *who*:

114493           **LANG**           Provide a default value for the internationalization variables that are unset or null.  
 114494 (See XBD Section 8.2 (on page 174) for the precedence of internationalization  
 114495 variables used to determine the values of locale categories.)

114496           **LC\_ALL**          If set to a non-empty string value, override the values of all the other  
 114497 internationalization variables.

114498           **LC\_CTYPE**       Determine the locale for the interpretation of sequences of bytes of text data as  
 114499 characters (for example, single-byte as opposed to multi-byte characters in  
 114500 arguments).

114501           **LC\_MESSAGES**   Determine the locale that should be used to affect the format and contents of  
 114502 diagnostic messages written to standard error.

114503           **LC\_TIME**        Determine the locale used for the format and contents of the date and time strings.

114504           **NLSPATH**       Determine the location of message catalogs for the processing of **LC\_MESSAGES**.

114505 XSI           **TZ**           Determine the timezone used when writing date and time information. If **TZ** is  
 114506 unset or null, an unspecified default timezone shall be used.

#### 114508 ASYNCHRONOUS EVENTS

114509           Default.

#### 114510 STDOUT

114511           The *who* utility shall write its default format to the standard output in an implementation-  
 114512 defined format, subject only to the requirement of containing the information described above.



114513 XSI OF XSI-conformant systems shall write the default information to the standard output in the  
 114514 following general format:

114515 `<name>[<state>]<line><time>[<activity>][<pid>][<comment>][<exit>]`

114516 For the `-b` option, `<line>` shall be "system boot". The `<name>` is unspecified.

114517 The following format shall be used for the `-T` option:

114518 `"%s %c %s %s\n" <name>, <terminal state>, <terminal name>,  
 114519 <time of login>`

114520 where `<terminal state>` is one of the following characters:

114521 + The terminal allows write access to other users.

114522 - The terminal denies write access to other users.

114523 ? The terminal write-access state cannot be determined.

114524 `<space>` This entry is not associated with a terminal.

114525 In the POSIX locale, the `<time of login>` shall be equivalent in format to the output of:

114526 `date +"%b %e %H:%M"`

114527 If the `-u` option is used with `-T`, the idle time shall be added to the end of the previous format in  
 114528 an unspecified format.

#### 114529 **STDERR**

114530 The standard error shall be used only for diagnostic messages.

#### 114531 **OUTPUT FILES**

114532 None.

#### 114533 **EXTENDED DESCRIPTION**

114534 None.

#### 114535 **EXIT STATUS**

114536 The following exit values shall be returned:

114537 0 Successful completion.

114538 >0 An error occurred.

#### 114539 **CONSEQUENCES OF ERRORS**

114540 Default.

#### 114541 **APPLICATION USAGE**

114542 The name *init* used for the system process is the most commonly used on historical systems, but  
 114543 it may vary.

114544 The "domain of accessibility" referred to is a broad concept that permits interpretation either on  
 114545 a very secure basis or even to allow a network-wide implementation like the historical *rwwho*.

#### 114546 **EXAMPLES**

114547 None.

#### 114548 **RATIONALE**

114549 Due to differences between historical implementations, the base options provided were a  
 114550 compromise to allow users to work with those functions. The standard developers also  
 114551 considered removing all the options, but felt that these options offered users valuable  
 114552 functionality. Additional options to match historical systems are available on XSI-conformant

- 114553 systems.
- 114554 It is recognized that the *who* command may be of limited usefulness, especially in a multi-level  
114555 secure environment. The standard developers considered, however, that having some standard  
114556 method of determining the “accessibility” of other users would aid user portability.
- 114557 No format was specified for the default *who* output for systems not supporting the XSI option. In  
114558 such a user-oriented command, designed only for human use, this was not considered to be a  
114559 deficiency.
- 114560 The format of the terminal name is unspecified, but the descriptions of *ps*, *talk*, and *write* require  
114561 that they use the same format.
- 114562 It is acceptable for an implementation to produce no output for an invocation of *who* **mil**.
- 114563 **FUTURE DIRECTIONS**
- 114564 None.
- 114565 **SEE ALSO**
- 114566 *mesg*
- 114567 XBD [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 215)
- 114568 **CHANGE HISTORY**
- 114569 First released in Issue 2.
- 114570 **Issue 6**
- 114571 This utility is marked as part of the User Portability Utilities option.
- 114572 The *TZ* entry is added to the ENVIRONMENT VARIABLES section.
- 114573 **Issue 7**
- 114574 SD5-XCU-ERN-58 is applied, clarifying the **-b** option.
- 114575 The *who* utility is moved from the User Portability Utilities option to the Base. User Portability  
114576 Utilities is now an option for interactive utilities.
- 114577 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

114578 **NAME**

114579       write — write to another user

114580 **SYNOPSIS**114581       write *user\_name* [*terminal*]114582 **DESCRIPTION**

114583       The *write* utility shall read lines from the standard input and write them to the terminal of the  
 114584       specified user. When first invoked, it shall write the message:

114585       **Message from** *sender-login-id* (*sending-terminal*) [*date*]...

114586       to *user\_name*. When it has successfully completed the connection, the sender's terminal shall be  
 114587       alerted twice to indicate that what the sender is typing is being written to the recipient's  
 114588       terminal.

114589       If the recipient wants to reply, this can be accomplished by typing:

114590       write *sender-login-id* [*sending-terminal*]

114591       upon receipt of the initial message. Whenever a line of input as delimited by an NL, EOF, or  
 114592       EOL special character (see XBD [Chapter 11](#), on page 199) is accumulated while in canonical  
 114593       input mode, the accumulated data shall be written on the other user's terminal. Characters shall  
 114594       be processed as follows:

- 114595       • Typing <alert> shall write the <alert> character to the recipient's terminal.
- 114596       • Typing the erase and kill characters shall affect the sender's terminal in the manner  
 114597       described by the **termios** interface in XBD [Chapter 11](#) (on page 199).
- 114598       • Typing the interrupt or end-of-file characters shall cause *write* to write an appropriate  
 114599       message ("EOT\n" in the POSIX locale) to the recipient's terminal and exit.
- 114600       • Typing characters from *LC\_CTYPE* classifications **print** or **space** shall cause those  
 114601       characters to be sent to the recipient's terminal.
- 114602       • When and only when the *stty* **ixten** local mode is enabled, the existence and processing of  
 114603       additional special control characters and multi-byte or single-byte functions is  
 114604       implementation-defined.
- 114605       • Typing other non-printable characters shall cause implementation-defined sequences of  
 114606       printable characters to be written to the recipient's terminal.

114607       To write to a user who is logged in more than once, the *terminal* argument can be used to  
 114608       indicate which terminal to write to; otherwise, the recipient's terminal is selected in an  
 114609       implementation-defined manner and an informational message is written to the sender's  
 114610       standard output, indicating which terminal was chosen.

114611       Permission to be a recipient of a *write* message can be denied or granted by use of the *mesg*  
 114612       utility. However, a user's privilege may further constrain the domain of accessibility of other  
 114613       users' terminals. The *write* utility shall fail when the user lacks appropriate privileges to perform  
 114614       the requested action.

114615 **OPTIONS**

114616       None.

114617 **OPERANDS**

- 114618 The following operands shall be supported:
- 114619 *user\_name* Login name of the person to whom the message shall be written. The application  
114620 shall ensure that this operand is of the form returned by the *who* utility.
- 114621 *terminal* Terminal identification in the same format provided by the *who* utility.
- 114622 **STDIN**
- 114623 Lines to be copied to the recipient's terminal are read from standard input.
- 114624 **INPUT FILES**
- 114625 None.
- 114626 **ENVIRONMENT VARIABLES**
- 114627 The following environment variables shall affect the execution of *write*:
- 114628 *LANG* Provide a default value for the internationalization variables that are unset or null.  
114629 (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
114630 variables used to determine the values of locale categories.)
- 114631 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
114632 internationalization variables.
- 114633 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
114634 characters (for example, single-byte as opposed to multi-byte characters in  
114635 arguments and input files). If the recipient's locale does not use an *LC\_CTYPE*  
114636 equivalent to the sender's, the results are undefined.
- 114637 *LC\_MESSAGES*
- 114638 Determine the locale that should be used to affect the format and contents of  
114639 diagnostic messages written to standard error and informative messages written to  
114640 standard output.
- 114641 XSI *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.
- 114642 **ASYNCHRONOUS EVENTS**
- 114643 If an interrupt signal is received, *write* shall write an appropriate message on the recipient's  
114644 terminal and exit with a status of zero. It shall take the standard action for all other signals.
- 114645 **STDOUT**
- 114646 An informational message shall be written to standard output if a recipient is logged in more  
114647 than once.
- 114648 **STDERR**
- 114649 The standard error shall be used only for diagnostic messages.
- 114650 **OUTPUT FILES**
- 114651 The recipient's terminal is used for output.
- 114652 **EXTENDED DESCRIPTION**
- 114653 None.
- 114654 **EXIT STATUS**
- 114655 The following exit values shall be returned:
- 114656 0 Successful completion.
- 114657 >0 The addressed user is not logged on or the addressed user denies permission.

114658 **CONSEQUENCES OF ERRORS**

114659 Default.

114660 **APPLICATION USAGE**114661 The *talk* utility is considered by some users to be a more usable utility on full-screen terminals.114662 **EXAMPLES**

114663 None.

114664 **RATIONALE**

114665 The *write* utility was included in this volume of POSIX.1-2008 since it can be implemented on all  
114666 terminal types. The standard developers considered the *talk* utility, which cannot be  
114667 implemented on certain terminals, to be a “better” communications interface. Both of these  
114668 programs are in widespread use on historical implementations. Therefore, the standard  
114669 developers decided that both utilities should be specified.

114670 The format of the terminal name is unspecified, but the descriptions of *ps*, *talk*, *who*, and *write*  
114671 require that they all use or accept the same format.

114672 **FUTURE DIRECTIONS**

114673 None.

114674 **SEE ALSO**114675 *mesg*, *talk*, *who*114676 XBD [Chapter 8](#) (on page 173), [Chapter 11](#) (on page 199)114677 **CHANGE HISTORY**

114678 First released in Issue 2.

114679 **Issue 5**

114680 The FUTURE DIRECTIONS section is added.

114681 **Issue 6**

114682 This utility is marked as part of the User Portability Utilities option.

114683 The normative text is reworded to avoid use of the term “must” for application requirements.

114684 **Issue 7**

114685 The *write* utility is moved from the User Portability Utilities option to the Base. User Portability  
114686 Utilities is now an option for interactive utilities.

114687 **NAME**114688 **xargs** — construct argument lists and invoke utility114689 **SYNOPSIS**

```
114690 XSI xargs [-ptx] [-E eofstr] [-I replstr|-L number|-n number]
114691 [-s size] [utility [argument...]]
```

114692 **DESCRIPTION**

114693 The *xargs* utility shall construct a command line consisting of the *utility* and *argument* operands  
 114694 specified followed by as many arguments read in sequence from standard input as fit in length  
 114695 and number constraints specified by the options. The *xargs* utility shall then invoke the  
 114696 constructed command line and wait for its completion. This sequence shall be repeated until one  
 114697 of the following occurs:

- 114698 • An end-of-file condition is detected on standard input.
- 114699 • An argument consisting of just the logical end-of-file string (see the `-E eofstr` option) is  
 114700 found on standard input after double-quote processing, <apostrophe> processing, and  
 114701 <backslash>-escape processing (see next paragraph). All arguments up to but not  
 114702 including the argument consisting of just the logical end-of-file string shall be used as  
 114703 arguments in constructed command lines.
- 114704 • An invocation of a constructed command line returns an exit status of 255.

114705 The application shall ensure that arguments in the standard input are separated by unquoted  
 114706 <blank> characters, unescaped <blank> characters, or <newline> characters. A string of zero or  
 114707 more non-double-quote ( ' ' ) characters and non-<newline> characters can be quoted by  
 114708 enclosing them in double-quotes. A string of zero or more non-<apostrophe> ( ' \ ' ' ) characters  
 114709 and non-<newline> characters can be quoted by enclosing them in <apostrophe> characters.  
 114710 Any unquoted character can be escaped by preceding it with a <backslash>. The utility named  
 114711 by *utility* shall be executed one or more times until the end-of-file is reached or the logical end-of  
 114712 file string is found. The results are unspecified if the utility named by *utility* attempts to read  
 114713 from its standard input.

114714 The generated command line length shall be the sum of the size in bytes of the utility name and  
 114715 each argument treated as strings, including a null byte terminator for each of these strings. The  
 114716 *xargs* utility shall limit the command line length such that when the command line is invoked,  
 114717 the combined argument and environment lists (see the *exec* family of functions in the System  
 114718 Interfaces volume of POSIX.1-2008) shall not exceed {ARG\_MAX}-2048 bytes. Within this  
 114719 constraint, if neither the `-n` nor the `-s` option is specified, the default command line length shall  
 114720 be at least {LINE\_MAX}.

114721 **OPTIONS**114722 The *xargs* utility shall conform to XBD [Section 12.2](#) (on page 215).

114723 The following options shall be supported:

114724 `-E eofstr` Use *eofstr* as the logical end-of-file string. If `-E` is not specified, it is unspecified  
 114725 whether the logical end-of-file string is the <underscore> character ( ' \_ ' ) or the  
 114726 end-of-file string capability is disabled. When *eofstr* is the null string, the logical  
 114727 end-of-file string capability shall be disabled and <underscore> characters shall be  
 114728 taken literally.

114729 XSI `-I replstr` Insert mode: *utility* is executed for each logical line from standard input.  
 114730 Arguments in the standard input shall be separated only by unescaped <newline>  
 114731 characters, not by <blank> characters. Any unquoted unescaped <blank>  
 114732 characters at the beginning of each line shall be ignored. The resulting argument  
 114733 shall be inserted in *arguments* in place of each occurrence of *replstr*. At least five

|        |     |                                                                                                                   |
|--------|-----|-------------------------------------------------------------------------------------------------------------------|
| 114734 |     | arguments in <i>arguments</i> can each contain one or more instances of <i>replstr</i> . Each of                  |
| 114735 |     | these constructed arguments cannot grow larger than an implementation-defined                                     |
| 114736 |     | limit greater than or equal to 255 bytes. Option <b>-x</b> shall be forced on.                                    |
| 114737 | XSI | <b>-L number</b> The <i>utility</i> shall be executed for each non-empty <i>number</i> lines of arguments from    |
| 114738 |     | standard input. The last invocation of <i>utility</i> shall be with fewer lines of arguments                      |
| 114739 |     | if fewer than <i>number</i> remain. A line is considered to end with the first <newline>                          |
| 114740 |     | unless the last character of the line is a <blank>; a trailing <blank> signals                                    |
| 114741 |     | continuation to the next non-empty line, inclusive.                                                               |
| 114742 |     | <b>-n number</b> Invoke <i>utility</i> using as many standard input arguments as possible, up to <i>number</i> (a |
| 114743 |     | positive decimal integer) arguments maximum. Fewer arguments shall be used if:                                    |
| 114744 |     | • The command line length accumulated exceeds the size specified by the <b>-s</b>                                 |
| 114745 |     | option (or {LINE_MAX} if there is no <b>-s</b> option).                                                           |
| 114746 |     | • The last iteration has fewer than <i>number</i> , but not zero, operands remaining.                             |
| 114747 |     | <b>-p</b> Prompt mode: the user is asked whether to execute <i>utility</i> at each invocation. Trace              |
| 114748 |     | mode ( <b>-t</b> ) is turned on to write the command instance to be executed, followed by                         |
| 114749 |     | a prompt to standard error. An affirmative response read from <b>/dev/tty</b> shall                               |
| 114750 |     | execute the command; otherwise, that particular invocation of <i>utility</i> shall be                             |
| 114751 |     | skipped.                                                                                                          |
| 114752 |     | <b>-s size</b> Invoke <i>utility</i> using as many standard input arguments as possible yielding a                |
| 114753 |     | command line length less than <i>size</i> (a positive decimal integer) bytes. Fewer                               |
| 114754 |     | arguments shall be used if:                                                                                       |
| 114755 |     | • The total number of arguments exceeds that specified by the <b>-n</b> option.                                   |
| 114756 | XSI | • The total number of lines exceeds that specified by the <b>-L</b> option.                                       |
| 114757 |     | • End-of-file is encountered on standard input before <i>size</i> bytes are                                       |
| 114758 |     | accumulated.                                                                                                      |
| 114759 |     | Values of <i>size</i> up to at least {LINE_MAX} bytes shall be supported, provided that                           |
| 114760 |     | the constraints specified in the DESCRIPTION are met. It shall not be considered                                  |
| 114761 |     | an error if a value larger than that supported by the implementation or exceeding                                 |
| 114762 |     | the constraints specified in the DESCRIPTION is given; <i>xargs</i> shall use the largest                         |
| 114763 |     | value it supports within the constraints.                                                                         |
| 114764 |     | <b>-t</b> Enable trace mode. Each generated command line shall be written to standard                             |
| 114765 |     | error just prior to invocation.                                                                                   |
| 114766 |     | <b>-x</b> Terminate if a constructed command line will not fit in the implied or specified size                   |
| 114767 |     | (see the <b>-s</b> option above).                                                                                 |

## 114768 OPERANDS

114769 The following operands shall be supported:

|        |                 |                                                                                                       |
|--------|-----------------|-------------------------------------------------------------------------------------------------------|
| 114770 | <i>utility</i>  | The name of the utility to be invoked, found by search path using the <i>PATH</i>                     |
| 114771 |                 | environment variable, described in XBD Chapter 8 (on page 173). If <i>utility</i> is                  |
| 114772 |                 | omitted, the default shall be the <i>echo</i> utility. If the <i>utility</i> operand names any of the |
| 114773 |                 | special built-in utilities in Section 2.14 (on page 2356), the results are undefined.                 |
| 114774 | <i>argument</i> | An initial option or operand for the invocation of <i>utility</i> .                                   |

114775 **STDIN**

114776 The standard input shall be a text file. The results are unspecified if an end-of-file condition is  
 114777 detected immediately following an escaped <newline>.

114778 **INPUT FILES**

114779 The file `/dev/tty` shall be used to read responses required by the `-p` option.

114780 **ENVIRONMENT VARIABLES**

114781 The following environment variables shall affect the execution of *xargs*:

114782 *LANG* Provide a default value for the internationalization variables that are unset or null.  
 114783 (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 114784 variables used to determine the values of locale categories.)

114785 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
 114786 internationalization variables.

114787 *LC\_COLLATE*

114788 Determine the locale for the behavior of ranges, equivalence classes, and multi-  
 114789 character collating elements used in the extended regular expression defined for  
 114790 the **yesexpr** locale keyword in the *LC\_MESSAGES* category.

114791 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
 114792 characters (for example, single-byte as opposed to multi-byte characters in  
 114793 arguments and input files) and the behavior of character classes used in the  
 114794 extended regular expression defined for the **yesexpr** locale keyword in the  
 114795 *LC\_MESSAGES* category.

114796 *LC\_MESSAGES*

114797 Determine the locale used to process affirmative responses, and the locale used to  
 114798 affect the format and contents of diagnostic messages and prompts written to  
 114799 standard error.

114800 XSI *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

114801 *PATH* Determine the location of *utility*, as described in XBD [Chapter 8](#) (on page 173).

114802 **ASYNCHRONOUS EVENTS**

114803 Default.

114804 **STDOUT**

114805 Not used.

114806 **STDERR**

114807 The standard error shall be used for diagnostic messages and the `-t` and `-p` options. If the `-t`  
 114808 option is specified, the *utility* and its constructed argument list shall be written to standard error,  
 114809 as it will be invoked, prior to invocation. If `-p` is specified, a prompt of the following format  
 114810 shall be written (in the POSIX locale):

114811 " ? . . . "

114812 at the end of the line of the output from `-t`.

114813 **OUTPUT FILES**

114814 None.



114815 **EXTENDED DESCRIPTION**

114816 None.

114817 **EXIT STATUS**

114818 The following exit values shall be returned:

114819 0 All invocations of *utility* returned exit status zero.

114820 1-125 A command line meeting the specified requirements could not be assembled, one or  
 114821 more of the invocations of *utility* returned a non-zero exit status, or some other error  
 114822 occurred.

114823 126 The utility specified by *utility* was found but could not be invoked.114824 127 The utility specified by *utility* could not be found.114825 **CONSEQUENCES OF ERRORS**

114826 If a command line meeting the specified requirements cannot be assembled, the utility cannot be  
 114827 invoked, an invocation of the utility is terminated by a signal, or an invocation of the utility exits  
 114828 with exit status 255, the *xargs* utility shall write a diagnostic message and exit without  
 114829 processing any remaining input.

114830 **APPLICATION USAGE**

114831 The 255 exit status allows a utility being used by *xargs* to tell *xargs* to terminate if it knows no  
 114832 further invocations using the current data stream will succeed. Thus, *utility* should explicitly *exit*  
 114833 with an appropriate value to avoid accidentally returning with 255.

114834 Note that since input is parsed as lines, <blank> characters separate arguments, and  
 114835 <backslash>, <apostrophe>, and double-quote characters are used for quoting, if *xargs* is used to  
 114836 bundle the output of commands like *find dir -print* or *ls* into commands to be executed,  
 114837 unexpected results are likely if any filenames contain <blank>, <newline>, or quoting characters.  
 114838 This can be solved by using *find* to call a script that converts each file found into a quoted string  
 114839 that is then piped to *xargs*, but in most cases it is preferable just to have *find* do the argument  
 114840 aggregation itself by using *-exec* with a '+' terminator instead of ';' . Note that the quoting  
 114841 rules used by *xargs* are not the same as in the shell. They were not made consistent here because  
 114842 existing applications depend on the current rules. An easy (but inefficient) method that can be  
 114843 used to transform input consisting of one argument per line into a quoted form that *xargs*  
 114844 interprets correctly is to precede each non-<newline> character with a <backslash>. More  
 114845 efficient alternatives are shown in Example 2 and Example 5 below.

114846 On implementations with a large value for {ARG\_MAX}, *xargs* may produce command lines  
 114847 longer than {LINE\_MAX}. For invocation of utilities, this is not a problem. If *xargs* is being used  
 114848 to create a text file, users should explicitly set the maximum command line length with the *-s*  
 114849 option.

114850 The *command*, *env*, *nice*, *nohup*, *time*, and *xargs* utilities have been specified to use exit code 127 if  
 114851 an error occurs so that applications can distinguish "failure to find a utility" from "invoked  
 114852 utility exited with an error indication". The value 127 was chosen because it is not commonly  
 114853 used for other meanings; most utilities use small values for "normal error conditions" and the  
 114854 values above 128 can be confused with termination due to receipt of a signal. The value 126 was  
 114855 chosen in a similar manner to indicate that the utility could be found, but not invoked. Some  
 114856 scripts produce meaningful error messages differentiating the 126 and 127 cases. The distinction  
 114857 between exit codes 126 and 127 is based on KornShell practice that uses 127 when all attempts to  
 114858 *exec* the utility fail with [ENOENT], and uses 126 when any attempt to *exec* the utility fails for  
 114859 any other reason.

## 114860 EXAMPLES

- 114861 1. The following command combines the output of the parenthesized commands (minus the  
114862 <apostrophe> characters) onto one line, which is then appended to the file log. It assumes  
114863 that the expansion of "\$0 \$" does not include any <apostrophe> or <newline>  
114864 characters.

114865 `(logname; date; printf "'%s'\n" "$0 $*") | xargs -E "" >>log`

- 114866 2. The following command invokes *diff* with successive pairs of arguments originally typed  
114867 as command line arguments. It assumes there are no embedded <newline> characters in  
114868 the elements of the original argument list.

114869 `printf "%s\n" "$@" | sed 's/^[[:alnum:]]/\&/g' |`  
114870 `xargs -E "" -n 2 -x diff`

- 114871 3. In the following commands, the user is asked which files in the current directory  
114872 (excluding dotfiles) are to be archived. The files are archived into **arch**; *a*, one at a time or  
114873 *b*, many at a time. The commands assume that no filenames contain <blank>, <newline>,  
114874 <backslash>, <apostrophe>, or double-quote characters.

114875 `a. ls | xargs -E "" -p -L 1 ar -r arch`

114876 `b. ls | xargs -E "" -p -L 1 | xargs -E "" ar -r arch`

- 114877 4. The following command invokes *command1* one or more times with multiple arguments,  
114878 stopping if an invocation of *command1* has a non-zero exit status.

114879 `xargs -E "" sh -c 'command1 "$@" || exit 255' sh < xargs_input`

- 114880 5. On XSI-conformant systems, the following command moves all files from directory **\$1** to  
114881 directory **\$2**, and echoes each move command just before doing it. It assumes no  
114882 filenames contain <newline> characters and that neither **\$1** nor **\$2** contains the sequence  
114883 "{}".

114884 `ls -A "$1" | sed -e 's/"\/"\/"/g' -e 's/.*/"&"/' |`  
114885 `xargs -E "" -I {} -t mv "$1"/{} "$2"/{}`

## 114886 RATIONALE

114887 The *xargs* utility was usually found only in System V-based systems; BSD systems included an  
114888 *apply* utility that provided functionality similar to *xargs -n number*. The SVID lists *xargs* as a  
114889 software development extension. This volume of POSIX.1-2008 does not share the view that it is  
114890 used only for development, and therefore it is not optional.

114891 The classic application of the *xargs* utility is in conjunction with the *find* utility to reduce the  
114892 number of processes launched by a simplistic use of the *find -exec* combination. The *xargs* utility  
114893 is also used to enforce an upper limit on memory required to launch a process. With this basis in  
114894 mind, this volume of POSIX.1-2008 selected only the minimal features required.

114895 Although the 255 exit status is mostly an accident of historical implementations, it allows a  
114896 utility being used by *xargs* to tell *xargs* to terminate if it knows no further invocations using the  
114897 current data stream shall succeed. Any non-zero exit status from a utility falls into the 1-125  
114898 range when *xargs* exits. There is no statement of how the various non-zero utility exit status  
114899 codes are accumulated by *xargs*. The value could be the addition of all codes, their highest  
114900 value, the last one received, or a single value such as 1. Since no algorithm is arguably better  
114901 than the others, and since many of the standard utilities say little more (portably) than  
114902 "pass/fail", no new algorithm was invented.

114903 Several other *xargs* options were removed because simple alternatives already exist within this

volume of POSIX.1-2008. For example, the `-i replstr` option can be just as efficiently performed using a shell **for** loop. Since *xargs* calls an *exec* function with each input line, the `-i` option does not usually exploit the grouping capabilities of *xargs*.

The requirement that *xargs* never produces command lines such that invocation of *utility* is within 2048 bytes of hitting the POSIX *exec* {ARG\_MAX} limitations is intended to guarantee that the invoked utility has room to modify its environment variables and command line arguments and still be able to invoke another utility. Note that the minimum {ARG\_MAX} allowed by the System Interfaces volume of POSIX.1-2008 is 4096 bytes and the minimum value allowed by this volume of POSIX.1-2008 is 2048 bytes; therefore, the 2048 bytes difference seems reasonable. Note, however, that *xargs* may never be able to invoke a utility if the environment passed in to *xargs* comes close to using {ARG\_MAX} bytes.

The version of *xargs* required by this volume of POSIX.1-2008 is required to wait for the completion of the invoked command before invoking another command. This was done because historical scripts using *xargs* assumed sequential execution. Implementations wanting to provide parallel operation of the invoked utilities are encouraged to add an option enabling parallel invocation, but should still wait for termination of all of the children before *xargs* terminates normally.

The `-e` option was omitted from the ISO POSIX-2:1993 standard in the belief that the *eofstr* option-argument was recognized only when it was on a line by itself and before quote and escape processing were performed, and that the logical end-of-file processing was only enabled if a `-e` option was specified. In that case, a simple *sed* script could be used to duplicate the `-e` functionality. Further investigation revealed that:

- The logical end-of-file string was checked for after quote and escape processing, making a *sed* script that provided equivalent functionality much more difficult to write.
- The default was to perform logical end-of-file processing with an `<underscore>` as the logical end-of-file string.

To correct this misunderstanding, the `-E eofstr` option was adopted from the X/Open Portability Guide. Users should note that the description of the `-E` option matches historical documentation of the `-e` option (which was not adopted because it did not support the Utility Syntax Guidelines), by saying that if *eofstr* is the null string, logical end-of-file processing is disabled. Historical implementations of *xargs* actually did not disable logical end-of-file processing; they treated a null argument found in the input as a logical end-of-file string. (A null *string* argument could be generated using single or double-quotes (' ' or " "). Since this behavior was not documented historically, it is considered to be a bug.

The `-I`, `-L`, and `-n` options are mutually-exclusive. Some implementations use the last one specified if more than one is given on a command line; other implementations treat combinations of the options in different ways.

## FUTURE DIRECTIONS

None.

## SEE ALSO

[Chapter 2](#) (on page 2319), [diff](#), [echo](#), [find](#)

XBD [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 215)

XSH [exec](#)

114947 **CHANGE HISTORY**

114948 First released in Issue 2.

114949 **Issue 5**

114950 A second FUTURE DIRECTION is added.

114951 **Issue 6**114952 The obsolescent `-e`, `-i`, and `-l` options are removed.114953 The following new requirements on POSIX implementations derive from alignment with the  
114954 Single UNIX Specification:

- 114955 • The `-p` option is added.
- 114956 • In the INPUT FILES section, the file `/dev/tty` is used to read responses required by the `-p`  
114957 option.
- 114958 • The STDERR section is updated to describe the `-p` option.

114959 The description of the `-E` option is aligned with the ISO POSIX-2: 1993 standard.

114960 The normative text is reworded to avoid use of the term “must” for application requirements.

114961 **Issue 7**114962 Austin Group Interpretation 1003.1-2001 #123 is applied, changing the description of the *xargs* `-I`  
114963 option.114964 Austin Group Interpretation 1003.1-2001 #126 is applied, changing the description of the  
114965 `LC_MESSAGES` environment variable.

114966 SD5-XCU-ERN-68 is applied.

114967 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

114968 SD5-XCU-ERN-128 is applied, clarifying the DESCRIPTION of the logical end-of-file string.

114969 SD5-XCU-ERN-132 is applied, updating the EXAMPLES section.

114970 POSIX.1-2008, Technical Corrigendum 1, XCU/TC1-2008/0149 [342] is applied.

+

114971 **NAME**114972 yacc — yet another compiler compiler (**DEVELOPMENT**)114973 **SYNOPSIS**114974 CD `yacc [-dltv] [-b file_prefix] [-p sym_prefix] grammar`114975 **DESCRIPTION**

114976 The *yacc* utility shall read a description of a context-free grammar in *grammar* and write C source  
 114977 code, conforming to the ISO C standard, to a code file, and optionally header information into a  
 114978 header file, in the current directory. The generated source code shall not depend on any  
 114979 undefined, unspecified, or implementation-defined behavior, except in cases where it is copied  
 114980 directly from the supplied grammar, or in cases that are documented by the implementation.  
 114981 The C code shall define a function and related routines and macros for an automaton that  
 114982 executes a parsing algorithm meeting the requirements in [Algorithms](#) (on page 3426).

114983 The form and meaning of the grammar are described in the EXTENDED DESCRIPTION section.

114984 The C source code and header file shall be produced in a form suitable as input for the C  
 114985 compiler (see [c99](#)).

114986 **OPTIONS**

114987 The *yacc* utility shall conform to XBD [Section 12.2](#) (on page 215), except for Guideline 9.

114988 The following options shall be supported:

114989 **-b *file\_prefix*** Use *file\_prefix* instead of **y** as the prefix for all output filenames. The code file  
 114990 **y.tab.c**, the header file **y.tab.h** (created when **-d** is specified), and the description  
 114991 file **y.output** (created when **-v** is specified), shall be changed to *file\_prefix.tab.c*,  
 114992 *file\_prefix.tab.h*, and *file\_prefix.output*, respectively.

114993 **-d** Write the header file; by default only the code file is written. The **#define**  
 114994 statements associate the token codes assigned by *yacc* with the user-declared token  
 114995 names. This allows source files other than **y.tab.c** to access the token codes.

114996 **-l** Produce a code file that does not contain any **#line** constructs. If this option is not  
 114997 present, it is unspecified whether the code file or header file contains **#line**  
 114998 directives. This should only be used after the grammar and the associated actions  
 114999 are fully debugged.

115000 **-p *sym\_prefix***  
 115001 Use *sym\_prefix* instead of **yy** as the prefix for all external names produced by *yacc*.  
 115002 The names affected shall include the functions *yyparse()*, *yylex()*, and *yyerror()*,  
 115003 and the variables *yyval*, *yychar*, and *yydebug*. (In the remainder of this section, the  
 115004 six symbols cited are referenced using their default names only as a notational  
 115005 convenience.) Local names may also be affected by the **-p** option; however, the **-p**  
 115006 option shall not affect **#define** symbols generated by *yacc*.

115007 **-t** Modify conditional compilation directives to permit compilation of debugging  
 115008 code in the code file. Runtime debugging statements shall always be contained in  
 115009 the code file, but by default conditional compilation directives prevent their  
 115010 compilation.

115011 **-v** Write a file containing a description of the parser and a report of conflicts  
 115012 generated by ambiguities in the grammar.

**115013 OPERANDS**

115014 The following operand is required:

115015 *grammar* A pathname of a file containing instructions, hereafter called *grammar*, for which a  
 115016 parser is to be created. The format for the grammar is described in the EXTENDED  
 115017 DESCRIPTION section.

**115018 STDIN**

115019 Not used.

**115020 INPUT FILES**

115021 The file *grammar* shall be a text file formatted as specified in the EXTENDED DESCRIPTION  
 115022 section.

**115023 ENVIRONMENT VARIABLES**

115024 The following environment variables shall affect the execution of *yacc*:

115025 *LANG* Provide a default value for the internationalization variables that are unset or null.  
 115026 (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 115027 variables used to determine the values of locale categories.)

115028 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
 115029 internationalization variables.

115030 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
 115031 characters (for example, single-byte as opposed to multi-byte characters in  
 115032 arguments and input files).

115033 *LC\_MESSAGES*

115034 Determine the locale that should be used to affect the format and contents of  
 115035 diagnostic messages written to standard error.

115036 XSI *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

115037 The *LANG* and *LC\_\** variables affect the execution of the *yacc* utility as stated. The *main()*  
 115038 function defined in [Yacc Library](#) (on page 3426) shall call:

115039 `setlocale(LC_ALL, "")`

115040 and thus the program generated by *yacc* shall also be affected by the contents of these variables  
 115041 at runtime.

**115042 ASYNCHRONOUS EVENTS**

115043 Default.

**115044 STDOUT**

115045 Not used.

**115046 STDERR**

115047 If shift/reduce or reduce/reduce conflicts are detected in *grammar*, *yacc* shall write a report of  
 115048 those conflicts to the standard error in an unspecified format.

115049 Standard error shall also be used for diagnostic messages.

**115050 OUTPUT FILES**

115051 The code file, the header file, and the description file shall be text files. All are described in the  
 115052 following sections.

**Code File**

This file shall contain the C source code for the *yyparse()* function. It shall contain code for the various semantic actions with macro substitution performed on them as described in the EXTENDED DESCRIPTION section. It also shall contain a copy of the **#define** statements in the header file. If a **%union** declaration is used, the declaration for YYSTYPE shall also be included in this file.

**Header File**

The header file shall contain **#define** statements that associate the token numbers with the token names. This allows source files other than the code file to access the token codes. If a **%union** declaration is used, the declaration for YYSTYPE and an *extern YYSTYPE yylval* declaration shall also be included in this file.

**Description File**

The description file shall be a text file containing a description of the state machine corresponding to the parser, using an unspecified format. Limits for internal tables (see [Limits](#), on page 3427) shall also be reported, in an implementation-defined manner. (Some implementations may use dynamic allocation techniques and have no specific limit values to report.)

**EXTENDED DESCRIPTION**

The *yacc* command accepts a language that is used to define a grammar for a target language to be parsed by the tables and code generated by *yacc*. The language accepted by *yacc* as a grammar for the target language is described below using the *yacc* input language itself.

The input *grammar* includes rules describing the input structure of the target language and code to be invoked when these rules are recognized to provide the associated semantic action. The code to be executed shall appear as bodies of text that are intended to be C-language code. These bodies of text shall not contain C-language trigraphs. The C-language inclusions are presumed to form a correct function when processed by *yacc* into its output files. The code included in this way shall be executed during the recognition of the target language.

Given a grammar, the *yacc* utility generates the files described in the OUTPUT FILES section. The code file can be compiled and linked using *c99*. If the declaration and programs sections of the grammar file did not include definitions of *main()*, *yylex()*, and *yyerror()*, the compiled output requires linking with externally supplied versions of those functions. Default versions of *main()* and *yyerror()* are supplied in the *yacc* library and can be linked in by using the *-ly* operand to *c99*. The *yacc* library interfaces need not support interfaces with other than the default **yy** symbol prefix. The application provides the lexical analyzer function, *yylex()*; the *lex* utility is specifically designed to generate such a routine.

**Input Language**

The application shall ensure that every specification file consists of three sections in order: *declarations*, *grammar rules*, and *programs*, separated by double <percent-sign> characters ("%"). The declarations and programs sections can be empty. If the latter is empty, the preceding "%" mark separating it from the rules section can be omitted.

The input is free form text following the structure of the grammar defined below.

## Lexical Structure of the Grammar

The <blank>, <newline>, and <form-feed> character shall be ignored, except that the application shall ensure that they do not appear in names or multi-character reserved symbols. Comments shall be enclosed in `"/* . . . */"`, and can appear wherever a name is valid.

Names are of arbitrary length, made up of letters, periods (`'.'`), underscores (`'_'`), and non-initial digits. Uppercase and lowercase letters are distinct. Conforming applications shall not use names beginning in `yy` or `YY` since the *yacc* parser uses such names. Many of the names appear in the final output of *yacc*, and thus they should be chosen to conform with any additional rules created by the C compiler to be used. In particular they appear in `#define` statements.

A literal shall consist of a single character enclosed in single-quote characters. All of the escape sequences supported for character constants by the ISO C standard shall be supported by *yacc*.

The relationship with the lexical analyzer is discussed in detail below.

The application shall ensure that the NUL character is not used in grammar rules or literals.

## Declarations Section

The declarations section is used to define the symbols used to define the target language and their relationship with each other. In particular, much of the additional information required to resolve ambiguities in the context-free grammar for the target language is provided here.

Usually *yacc* assigns the relationship between the symbolic names it generates and their underlying numeric value. The declarations section makes it possible to control the assignment of these values.

It is also possible to keep semantic information associated with the tokens currently on the parse stack in a user-defined C-language **union**, if the members of the union are associated with the various names in the grammar. The declarations section provides for this as well.

The first group of declarators below all take a list of names as arguments. That list can optionally be preceded by the name of a C union member (called a *tag* below) appearing within `'<'` and `'>'`. (As an exception to the typographical conventions of the rest of this volume of POSIX.1-2008, in this case `<tag>` does not represent a metavariable, but the literal angle bracket characters surrounding a symbol.) The use of *tag* specifies that the tokens named on this line shall be of the same C type as the union member referenced by *tag*. This is discussed in more detail below.

For lists used to define tokens, the first appearance of a given token can be followed by a positive integer (as a string of decimal digits). If this is done, the underlying value assigned to it for lexical purposes shall be taken to be that number.

The following declares *name* to be a token:

```
%token [<tag>] name [number] [name [number]]...
```

If *tag* is present, the C type for all tokens on this line shall be declared to be the type referenced by *tag*. If a positive integer, *number*, follows a *name*, that value shall be assigned to the token.

The following declares *name* to be a token, and assigns precedence to it:

```
%left [<tag>] name [number] [name [number]]...
```

```
%right [<tag>] name [number] [name [number]]...
```

One or more lines, each beginning with one of these symbols, can appear in this section. All tokens on the same line have the same precedence level and associativity; the lines are in order



of increasing precedence or binding strength. **%left** denotes that the operators on that line are left associative, and **%right** similarly denotes right associative operators. If *tag* is present, it shall declare a C type for *names* as described for **%token**.

The following declares *name* to be a token, and indicates that this cannot be used associatively:

```
%nonassoc [<tag>] name [number] [name [number]]...
```

If the parser encounters associative use of this token it reports an error. If *tag* is present, it shall declare a C type for *names* as described for **%token**.

The following declares that union member *names* are non-terminals, and thus it is required to have a *tag* field at its beginning:

```
%type <tag> name...
```

Because it deals with non-terminals only, assigning a token number or using a literal is also prohibited. If this construct is present, *yacc* shall perform type checking; if this construct is not present, the parse stack shall hold only the **int** type.

Every name used in *grammar* not defined by a **%token**, **%left**, **%right**, or **%nonassoc** declaration is assumed to represent a non-terminal symbol. The *yacc* utility shall report an error for any non-terminal symbol that does not appear on the left side of at least one grammar rule.

Once the type, precedence, or token number of a name is specified, it shall not be changed. If the first declaration of a token does not assign a token number, *yacc* shall assign a token number. Once this assignment is made, the token number shall not be changed by explicit assignment.

The following declarators do not follow the previous pattern.

The following declares the non-terminal *name* to be the *start symbol*, which represents the largest, most general structure described by the grammar rules:

```
%start name
```

By default, it is the left-hand side of the first grammar rule; this default can be overridden with this declaration.

The following declares the *yacc* value stack to be a union of the various types of values desired.

```
%union { body of union (in C) }
```

The body of the union shall not contain unbalanced curly brace preprocessing tokens.

By default, the values returned by actions (see below) and the lexical analyzer shall be of type **int**. The *yacc* utility keeps track of types, and it shall insert corresponding union member names in order to perform strict type checking of the resulting parser.

Alternatively, given that at least one *<tag>* construct is used, the union can be declared in a header file (which shall be included in the declarations section by using a **#include** construct within **%{** and **%}**), and a **typedef** used to define the symbol **YYSTYPE** to represent this union. The effect of **%union** is to provide the declaration of **YYSTYPE** directly from the *yacc* input.

C-language declarations and definitions can appear in the declarations section, enclosed by the following marks:

```
%{ ... %}
```

These statements shall be copied into the code file, and have global scope within it so that they can be used in the rules and program sections. The statements shall not contain **"%}"** outside a comment, string literal, or multi-character constant.

115178 The application shall ensure that the declarations section is terminated by the token `%%`.

## 115179 Grammar Rules in yacc

115180 The rules section defines the context-free grammar to be accepted by the function *yacc* generates,  
115181 and associates with those rules C-language actions and additional precedence information. The  
115182 grammar is described below, and a formal definition follows.

115183 The rules section is comprised of one or more grammar rules. A grammar rule has the form:

115184 `A : BODY ;`

115185 The symbol **A** represents a non-terminal name, and **BODY** represents a sequence of zero or  
115186 more *names*, *literals*, and *semantic actions* that can then be followed by optional *precedence rules*.  
115187 Only the names and literals participate in the formation of the grammar; the semantic actions  
115188 and precedence rules are used in other ways. The `<colon>` and the `<semicolon>` are *yacc*  
115189 punctuation. If there are several successive grammar rules with the same left-hand side, the  
115190 `<vertical-line>` (`'|'`) can be used to avoid rewriting the left-hand side; in this case the  
115191 `<semicolon>` appears only after the last rule. The BODY part can be empty (or empty of names  
115192 and literals) to indicate that the non-terminal symbol matches the empty string.

115193 The *yacc* utility assigns a unique number to each rule. Rules using the vertical bar notation are  
115194 distinct rules. The number assigned to the rule appears in the description file.

115195 The elements comprising a BODY are:

115196 *name, literal* These form the rules of the grammar: *name* is either a *token* or a *non-terminal*; *literal*  
115197 stands for itself (less the lexically required quotation marks).

115198 *semantic action*

115199 With each grammar rule, the user can associate actions to be performed each time  
115200 the rule is recognized in the input process. (Note that the word “action” can also  
115201 refer to the actions of the parser—shift, reduce, and so on.)

115202 These actions can return values and can obtain the values returned by previous  
115203 actions. These values are kept in objects of type YYSTYPE (see **%union**). The  
115204 result value of the action shall be kept on the parse stack with the left-hand side of  
115205 the rule, to be accessed by other reductions as part of their right-hand side. By  
115206 using the `<tag>` information provided in the declarations section, the code  
115207 generated by *yacc* can be strictly type checked and contain arbitrary information. In  
115208 addition, the lexical analyzer can provide the same kinds of values for tokens, if  
115209 desired.

115210 An action is an arbitrary C statement and as such can do input or output, call  
115211 subprograms, and alter external variables. An action is one or more C statements  
115212 enclosed in curly braces `'{'` and `'}'`. The statements shall not contain  
115213 unbalanced curly brace preprocessing tokens.

115214 Certain pseudo-variables can be used in the action. These are macros for access to  
115215 data structures known internally to *yacc*.

115216 **\$\$** The value of the action can be set by assigning it to **\$\$**. If type  
115217 checking is enabled and the type of the value to be assigned cannot  
115218 be determined, a diagnostic message may be generated.

115219 **\$number** This refers to the value returned by the component specified by the  
115220 token *number* in the right side of a rule, reading from left to right;  
115221 *number* can be zero or negative. If *number* is zero or negative, it refers

115222 to the data associated with the name on the parser's stack preceding  
 115223 the leftmost symbol of the current rule. (That is, "\$0" refers to the  
 115224 name immediately preceding the leftmost name in the current rule to  
 115225 be found on the parser's stack and "\$-1" refers to the symbol to *its*  
 115226 left.) If *number* refers to an element past the current point in the rule,  
 115227 or beyond the bottom of the stack, the result is undefined. If type  
 115228 checking is enabled and the type of the value to be assigned cannot  
 115229 be determined, a diagnostic message may be generated.

115230 \$<*tag*>*number*

115231 These correspond exactly to the corresponding symbols without the  
 115232 *tag* inclusion, but allow for strict type checking (and preclude  
 115233 unwanted type conversions). The effect is that the macro is expanded  
 115234 to use *tag* to select an element from the YYSTYPE union (using  
 115235 *dataname.tag*). This is particularly useful if *number* is not positive.

115236 \$<*tag*>\$

115237 This imposes on the reference the type of the union member  
 115238 referenced by *tag*. This construction is applicable when a reference to  
 115239 a left context value occurs in the grammar, and provides *yacc* with a  
 means for selecting a type.

115240 Actions can occur anywhere in a rule (not just at the end); an action can access  
 115241 values returned by actions to its left, and in turn the value it returns can be  
 115242 accessed by actions to its right. An action appearing in the middle of a rule shall be  
 115243 equivalent to replacing the action with a new non-terminal symbol and adding an  
 115244 empty rule with that non-terminal symbol on the left-hand side. The semantic  
 115245 action associated with the new rule shall be equivalent to the original action. The  
 115246 use of actions within rules might introduce conflicts that would not otherwise  
 115247 exist.

115248 By default, the value of a rule shall be the value of the first element in it. If the first  
 115249 element does not have a type (particularly in the case of a literal) and type  
 115250 checking is turned on by **%type**, an error message shall result.

115251 *precedence* The keyword **%prec** can be used to change the precedence level associated with a  
 115252 particular grammar rule. Examples of this are in cases where a unary and binary  
 115253 operator have the same symbolic representation, but need to be given different  
 115254 precedences, or where the handling of an ambiguous if-else construction is  
 115255 necessary. The reserved symbol **%prec** can appear immediately after the body of  
 115256 the grammar rule and can be followed by a token name or a literal. It shall cause  
 115257 the precedence of the grammar rule to become that of the following token name or  
 115258 literal. The action for the rule as a whole can follow **%prec**.

115259 If a program section follows, the application shall ensure that the grammar rules are terminated  
 115260 by **%%**.

## 115261 Programs Section

115262 The *programs* section can include the definition of the lexical analyzer *yylex()*, and any other  
 115263 functions; for example, those used in the actions specified in the grammar rules. It is unspecified  
 115264 whether the programs section precedes or follows the semantic actions in the output file;  
 115265 therefore, if the application contains any macro definitions and declarations intended to apply to  
 115266 the code in the semantic actions, it shall place them within "%{ . . . %}" in the declarations  
 115267 section.

**Input Grammar**

The following input to *yacc* yields a parser for the input to *yacc*. This formal syntax takes precedence over the preceding text syntax description.

The lexical structure is defined less precisely; [Lexical Structure of the Grammar](#) (on page 3418) defines most terms. The correspondence between the previous terms and the tokens below is as follows.

**IDENTIFIER** This corresponds to the concept of *name*, given previously. It also includes literals as defined previously.

**C\_IDENTIFIER** This is a name, and additionally it is known to be followed by a <colon>. A literal cannot yield this token.

**NUMBER** A string of digits (a non-negative decimal integer).

**TYPE, LEFT, MARK, LCURL, RCURL**

These correspond directly to **%type**, **%left**, **%%**, **%{**, and **%}**.

**{...}** This indicates C-language source code, with the possible inclusion of '\$' macros as discussed previously.

```

/* Grammar for the input to yacc. */
/* Basic entries. */
/* The following are recognized by the lexical analyzer. */

%token IDENTIFIER /* Includes identifiers and literals */
%token C_IDENTIFIER /* identifier (but not literal)
 followed by a :. */
%token NUMBER /* [0-9][0-9]* */

/* Reserved words : %type=>TYPE %left=>LEFT, and so on */
%token LEFT RIGHT NONASSOC TOKEN PREC TYPE START UNION

%token MARK /* The %% mark. */
%token LCURL /* The %{ mark. */
%token RCURL /* The %} mark. */

/* 8-bit character literals stand for themselves; */
/* tokens have to be defined for multi-byte characters. */

%start spec

%%

spec : defs MARK rules tail
 ;
tail : MARK
 {
 /* In this action, set up the rest of the file. */
 }
 | /* Empty; the second MARK is optional. */
 ;
defs : /* Empty. */
 | defs def
 ;
def : START IDENTIFIER
 | UNION

```

```

115312 {
115313 /* Copy union definition to output. */
115314 }
115315 | LCURL
115316 {
115317 /* Copy C code to output file. */
115318 }
115319 RCURL
115320 | rword tag nlist
115321 ;
115322 rword : TOKEN
115323 | LEFT
115324 | RIGHT
115325 | NONASSOC
115326 | TYPE
115327 ;
115328 tag : /* Empty: union tag ID optional. */
115329 | '<' IDENTIFIER '>'
115330 ;
115331 nlist : nmno
115332 | nlist nmno
115333 ;
115334 nmno : IDENTIFIER /* Note: literal invalid with % type. */
115335 | IDENTIFIER NUMBER /* Note: invalid with % type. */
115336 ;
115337 /* Rule section */
115338 rules : C_IDENTIFIER rbody prec
115339 | rules rule
115340 ;
115341 rule : C_IDENTIFIER rbody prec
115342 | '|' rbody prec
115343 ;
115344 rbody : /* empty */
115345 | rbody IDENTIFIER
115346 | rbody act
115347 ;
115348 act : '{'
115349 {
115350 /* Copy action, translate $$, and so on. */
115351 }
115352 '}'
115353 ;
115354 prec : /* Empty */
115355 | PREC IDENTIFIER
115356 | PREC IDENTIFIER act
115357 | prec ';'
115358 ;

```

## Conflicts

The parser produced for an input grammar may contain states in which conflicts occur. The conflicts occur because the grammar is not LALR(1). An ambiguous grammar always contains at least one LALR(1) conflict. The *yacc* utility shall resolve all conflicts, using either default rules or user-specified precedence rules.

Conflicts are either shift/reduce conflicts or reduce/reduce conflicts. A shift/reduce conflict is where, for a given state and lookahead symbol, both a shift action and a reduce action are possible. A reduce/reduce conflict is where, for a given state and lookahead symbol, reductions by two different rules are possible.

The rules below describe how to specify what actions to take when a conflict occurs. Not all shift/reduce conflicts can be successfully resolved this way because the conflict may be due to something other than ambiguity, so incautious use of these facilities can cause the language accepted by the parser to be much different from that which was intended. The description file shall contain sufficient information to understand the cause of the conflict. Where ambiguity is the reason either the default or explicit rules should be adequate to produce a working parser.

The declared precedences and associativities (see [Declarations Section](#), on page 3418) are used to resolve parsing conflicts as follows:

1. A precedence and associativity is associated with each grammar rule; it is the precedence and associativity of the last token or literal in the body of the rule. If the **%prec** keyword is used, it overrides this default. Some grammar rules might not have both precedence and associativity.
2. If there is a shift/reduce conflict, and both the grammar rule and the input symbol have precedence and associativity associated with them, then the conflict is resolved in favor of the action (shift or reduce) associated with the higher precedence. If the precedences are the same, then the associativity is used; left associative implies reduce, right associative implies shift, and non-associative implies an error in the string being parsed.
3. When there is a shift/reduce conflict that cannot be resolved by rule 2, the shift is done. Conflicts resolved this way are counted in the diagnostic output described in [Error Handling](#).
4. When there is a reduce/reduce conflict, a reduction is done by the grammar rule that occurs earlier in the input sequence. Conflicts resolved this way are counted in the diagnostic output described in [Error Handling](#).

Conflicts resolved by precedence or associativity shall not be counted in the shift/reduce and reduce/reduce conflicts reported by *yacc* on either standard error or in the description file.

## Error Handling

The token **error** shall be reserved for error handling. The name **error** can be used in grammar rules. It indicates places where the parser can recover from a syntax error. The default value of **error** shall be 256. Its value can be changed using a **%token** declaration. The lexical analyzer should not return the value of **error**.

The parser shall detect a syntax error when it is in a state where the action associated with the lookahead symbol is **error**. A semantic action can cause the parser to initiate error handling by executing the macro YYERROR. When YYERROR is executed, the semantic action passes control back to the parser. YYERROR cannot be used outside of semantic actions.

When the parser detects a syntax error, it normally calls *yyerror()* with the character string "syntax error" as its argument. The call shall not be made if the parser is still recovering

from a previous error when the error is detected. The parser is considered to be recovering from a previous error until the parser has shifted over at least three normal input symbols since the last error was detected or a semantic action has executed the macro *yyerrok*. The parser shall not call *yyerror()* when YYERROR is executed.

The macro function YYRECOVERING shall return 1 if a syntax error has been detected and the parser has not yet fully recovered from it. Otherwise, zero shall be returned.

When a syntax error is detected by the parser, the parser shall check if a previous syntax error has been detected. If a previous error was detected, and if no normal input symbols have been shifted since the preceding error was detected, the parser checks if the lookahead symbol is an endmarker (see [Interface to the Lexical Analyzer](#)). If it is, the parser shall return with a non-zero value. Otherwise, the lookahead symbol shall be discarded and normal parsing shall resume.

When YYERROR is executed or when the parser detects a syntax error and no previous error has been detected, or at least one normal input symbol has been shifted since the previous error was detected, the parser shall pop back one state at a time until the parse stack is empty or the current state allows a shift over **error**. If the parser empties the parse stack, it shall return with a non-zero value. Otherwise, it shall shift over **error** and then resume normal parsing. If the parser reads a lookahead symbol before the error was detected, that symbol shall still be the lookahead symbol when parsing is resumed.

The macro *yyerrok* in a semantic action shall cause the parser to act as if it has fully recovered from any previous errors. The macro *yyclearin* shall cause the parser to discard the current lookahead token. If the current lookahead token has not yet been read, *yyclearin* shall have no effect.

The macro YYACCEPT shall cause the parser to return with the value zero. The macro YYABORT shall cause the parser to return with a non-zero value.

### Interface to the Lexical Analyzer

The *yylex()* function is an integer-valued function that returns a *token number* representing the kind of token read. If there is a value associated with the token returned by *yylex()* (see the discussion of *tag* above), it shall be assigned to the external variable *yyval*.

If the parser and *yylex()* do not agree on these token numbers, reliable communication between them cannot occur. For (single-byte character) literals, the token is simply the numeric value of the character in the current character set. The numbers for other tokens can either be chosen by *yacc*, or chosen by the user. In either case, the **#define** construct of C is used to allow *yylex()* to return these numbers symbolically. The **#define** statements are put into the code file, and the header file if that file is requested. The set of characters permitted by *yacc* in an identifier is larger than that permitted by C. Token names found to contain such characters shall not be included in the **#define** declarations.

If the token numbers are chosen by *yacc*, the tokens other than literals shall be assigned numbers greater than 256, although no order is implied. A token can be explicitly assigned a number by following its first appearance in the declarations section with a number. Names and literals not defined this way retain their default definition. All token numbers assigned by *yacc* shall be unique and distinct from the token numbers used for literals and user-assigned tokens. If duplicate token numbers cause conflicts in parser generation, *yacc* shall report an error; otherwise, it is unspecified whether the token assignment is accepted or an error is reported.

The end of the input is marked by a special token called the *endmarker*, which has a token number that is zero or negative. (These values are invalid for any other token.) All lexical analyzers shall return zero or negative as a token number upon reaching the end of their input.

If the tokens up to, but excluding, the endmarker form a structure that matches the start symbol, the parser shall accept the input. If the endmarker is seen in any other context, it shall be considered an error.

### Completing the Program

In addition to *yyparse()* and *yylex()*, the functions *yyerror()* and *main()* are required to make a complete program. The application can supply *main()* and *yyerror()*, or those routines can be obtained from the *yacc* library.

### Yacc Library

The following functions shall appear only in the *yacc* library accessible through the **-l y** operand to *c99*; they can therefore be redefined by a conforming application:

#### **int main(void)**

This function shall call *yyparse()* and exit with an unspecified value. Other actions within this function are unspecified.

#### **int yyerror(const char \*s)**

This function shall write the NUL-terminated argument to standard error, followed by a <newline>.

The order of the **-l y** and **-l l** operands given to *c99* is significant; the application shall either provide its own *main()* function or ensure that **-l y** precedes **-l l**.

### Debugging the Parser

The parser generated by *yacc* shall have diagnostic facilities in it that can be optionally enabled at either compile time or at runtime (if enabled at compile time). The compilation of the runtime debugging code is under the control of *YYDEBUG*, a preprocessor symbol. If *YYDEBUG* has a non-zero value, the debugging code shall be included. If its value is zero, the code shall not be included.

In parsers where the debugging code has been included, the external **int yydebug** can be used to turn debugging on (with a non-zero value) and off (zero value) at runtime. The initial value of *yydebug* shall be zero.

When **-t** is specified, the code file shall be built such that, if *YYDEBUG* is not already defined at compilation time (using the *c99* **-D YYDEBUG** option, for example), *YYDEBUG* shall be set explicitly to 1. When **-t** is not specified, the code file shall be built such that, if *YYDEBUG* is not already defined, it shall be set explicitly to zero.

The format of the debugging output is unspecified but includes at least enough information to determine the shift and reduce actions, and the input symbols. It also provides information about error recovery.

### Algorithms

The parser constructed by *yacc* implements an LALR(1) parsing algorithm as documented in the literature. It is unspecified whether the parser is table-driven or direct-coded.

A parser generated by *yacc* shall never request an input symbol from *yylex()* while in a state where the only actions other than the error action are reductions by a single rule.

The literature of parsing theory defines these concepts.



## Limits

The *yacc* utility may have several internal tables. The minimum maximums for these tables are shown in the following table. The exact meaning of these values is implementation-defined. The implementation shall define the relationship between these values and between them and any error messages that the implementation may generate should it run out of space for any internal structure. An implementation may combine groups of these resources into a single pool as long as the total available to the user does not fall below the sum of the sizes specified by this section.

**Table 4-23** Internal Limits in *yacc*

| Limit      | Minimum<br>Maximum | Description                                                                                                                                                                                                                                                        |
|------------|--------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| {NTERMS}   | 126                | Number of tokens.                                                                                                                                                                                                                                                  |
| {NNONTERM} | 200                | Number of non-terminals.                                                                                                                                                                                                                                           |
| {NPROD}    | 300                | Number of rules.                                                                                                                                                                                                                                                   |
| {NSTATES}  | 600                | Number of states.                                                                                                                                                                                                                                                  |
| {MEMSIZE}  | 5 200              | Length of rules. The total length, in names (tokens and non-terminals), of all the rules of the grammar. The left-hand side is counted for each rule, even if it is not explicitly repeated, as specified in <a href="#">Grammar Rules in yacc</a> (on page 3420). |
| {ACTSIZE}  | 4 000              | Number of actions. “Actions” here (and in the description file) refer to parser actions (shift, reduce, and so on) not to semantic actions defined in <a href="#">Grammar Rules in yacc</a> (on page 3420).                                                        |

## EXIT STATUS

The following exit values shall be returned:

0 Successful completion.

>0 An error occurred.

## CONSEQUENCES OF ERRORS

If any errors are encountered, the run is aborted and *yacc* exits with a non-zero status. Partial code files and header files may be produced. The summary information in the description file shall always be produced if the *-v* flag is present.

## APPLICATION USAGE

Historical implementations experience name conflicts on the names **yacc.tmp**, **yacc.acts**, **yacc.debug**, **y.tab.c**, **y.tab.h**, and **y.output** if more than one copy of *yacc* is running in a single directory at one time. The *-b* option was added to overcome this problem. The related problem of allowing multiple *yacc* parsers to be placed in the same file was addressed by adding a *-p* option to override the previously hard-coded **yy** variable prefix.

The description of the *-p* option specifies the minimal set of function and variable names that cause conflict when multiple parsers are linked together. **YYSTYPE** does not need to be changed. Instead, the programmer can use *-b* to give the header files for different parsers different names, and then the file with the *yylex()* for a given parser can include the header for that parser. Names such as *yyclearerr* do not need to be changed because they are used only in the actions; they do not have linkage. It is possible that an implementation has other names, either internal ones for implementing things such as *yyclearerr*, or providing non-standard features that it wants

to change with `-p`.

Unary operators that are the same token as a binary operator in general need their precedence adjusted. This is handled by the `%prec` advisory symbol associated with the particular grammar rule defining that unary operator. (See [Grammar Rules in yacc](#) (on page 3420).) Applications are not required to use this operator for unary operators, but the grammars that do not require it are rare.

## EXAMPLES

Access to the *yacc* library is obtained with library search operands to *c99*. To use the *yacc* library *main()*:

```
c99 y.tab.c -l y
```

Both the *lex* library and the *yacc* library contain *main()*. To access the *yacc main()*:

```
c99 y.tab.c lex.yy.c -l y -l l
```

This ensures that the *yacc* library is searched first, so that its *main()* is used.

The historical *yacc* libraries have contained two simple functions that are normally coded by the application programmer. These functions are similar to the following code:

```
#include <locale.h>
int main(void)
{
 extern int yyparse();
 setlocale(LC_ALL, "");

 /* If the following parser is one created by lex, the
 application must be careful to ensure that LC_CTYPE
 and LC_COLLATE are set to the POSIX locale. */
 (void) yyparse();
 return (0);
}

#include <stdio.h>

int yyerror(const char *msg)
{
 (void) fprintf(stderr, "%s\n", msg);
 return (0);
}
```

## RATIONALE

The references in **Referenced Documents** may be helpful in constructing the parser generator. The referenced DeRemer and Pennello article (along with the works it references) describes a technique to generate parsers that conform to this volume of POSIX.1-2008. Work in this area continues to be done, so implementors should consult current literature before doing any new implementations. The original Knuth article is the theoretical basis for this kind of parser, but the tables it generates are impractically large for reasonable grammars and should not be used. The “equivalent to” wording is intentional to assure that the best tables that are LALR(1) can be generated.

There has been confusion between the class of grammars, the algorithms needed to generate parsers, and the algorithms needed to parse the languages. They are all reasonably orthogonal. In particular, a parser generator that accepts the full range of LR(1) grammars need not generate a table any more complex than one that accepts SLR(1) (a relatively weak class of LR grammars)

for a grammar that happens to be SLR(1). Such an implementation need not recognize the case, either; table compression can yield the SLR(1) table (or one even smaller than that) without recognizing that the grammar is SLR(1). The speed of an LR(1) parser for any class is dependent more upon the table representation and compression (or the code generation if a direct parser is generated) than upon the class of grammar that the table generator handles.

The speed of the parser generator is somewhat dependent upon the class of grammar it handles. However, the original Knuth article algorithms for constructing LR parsers were judged by its author to be impractically slow at that time. Although full LR is more complex than LALR(1), as computer speeds and algorithms improve, the difference (in terms of acceptable wall-clock execution time) is becoming less significant.

Potential authors are cautioned that the referenced DeRemer and Pennello article previously cited identifies a bug (an over-simplification of the computation of LALR(1) lookahead sets) in some of the LALR(1) algorithm statements that preceded it to publication. They should take the time to seek out that paper, as well as current relevant work, particularly Aho's.

The **-b** option was added to provide a portable method for permitting *yacc* to work on multiple separate parsers in the same directory. If a directory contains more than one *yacc* grammar, and both grammars are constructed at the same time (by, for example, a parallel *make* program), conflict results. While the solution is not historical practice, it corrects a known deficiency in historical implementations. Corresponding changes were made to all sections that referenced the filenames **y.tab.c** (now "the code file"), **y.tab.h** (now "the header file"), and **y.output** (now "the description file").

The grammar for *yacc* input is based on System V documentation. The textual description shows there that the **' ; '** is required at the end of the rule. The grammar and the implementation do not require this. (The use of **C\_IDENTIFIER** causes a reduce to occur in the right place.)

Also, in that implementation, the constructs such as **%token** can be terminated by a **<semicolon>**, but this is not permitted by the grammar. The keywords such as **%token** can also appear in uppercase, which is again not discussed. In most places where **' % '** is used, **<backslash>** can be substituted, and there are alternate spellings for some of the symbols (for example, **%LEFT** can be **"%<"** or even **"\<"**).

Historically, **<tag>** can contain any characters except **' > '**, including white space, in the implementation. However, since the *tag* must reference an ISO C standard union member, in practice conforming implementations need to support only the set of characters for ISO C standard identifiers in this context.

Some historical implementations are known to accept actions that are terminated by a period. Historical implementations often allow **' \$ '** in names. A conforming implementation does not need to support either of these behaviors.

Deciding when to use **%prec** illustrates the difficulty in specifying the behavior of *yacc*. There may be situations in which the *grammar* is not, strictly speaking, in error, and yet *yacc* cannot interpret it unambiguously. The resolution of ambiguities in the grammar can in many instances be resolved by providing additional information, such as using **%type** or **%union** declarations. It is often easier and it usually yields a smaller parser to take this alternative when it is appropriate.

The size and execution time of a program produced without the runtime debugging code is usually smaller and slightly faster in historical implementations.

Statistics messages from several historical implementations include the following types of information:

115627 *n*/512 terminals, *n*/300 non-terminals  
 115628 *n*/600 grammar rules, *n*/1500 states  
 115629 *n* shift/reduce, *n* reduce/reduce conflicts reported  
 115630 *n*/350 working sets used  
 115631 Memory: states, etc. *n*/15 000, parser *n*/15 000  
 115632 *n*/600 distinct lookahead sets  
 115633 *n* extra closures  
 115634 *n* shift entries, *n* exceptions  
 115635 *n* goto entries  
 115636 *n* entries saved by goto default  
 115637 Optimizer space used: input *n*/15 000, output *n*/15 000  
 115638 *n* table entries, *n* zero  
 115639 Maximum spread: *n*, Maximum offset: *n*

115640 The report of internal tables in the description file is left implementation-defined because all  
 115641 aspects of these limits are also implementation-defined. Some implementations may use  
 115642 dynamic allocation techniques and have no specific limit values to report.

115643 The format of the **y.output** file is not given because specification of the format was not seen to  
 115644 enhance applications portability. The listing is primarily intended to help human users  
 115645 understand and debug the parser; use of **y.output** by a conforming application script would be  
 115646 unusual. Furthermore, implementations have not produced consistent output and no popular  
 115647 format was apparent. The format selected by the implementation should be human-readable, in  
 115648 addition to the requirement that it be a text file.

115649 Standard error reports are not specifically described because they are seldom of use to  
 115650 conforming applications and there was no reason to restrict implementations.

115651 Some implementations recognize "`={"`" as equivalent to '`{`' because it appears in historical  
 115652 documentation. This construction was recognized and documented as obsolete as long ago as  
 115653 1978, in the referenced *Yacc: Yet Another Compiler-Compiler*. This volume of POSIX.1-2008 chose to  
 115654 leave it as obsolete and omit it.

115655 Multi-byte characters should be recognized by the lexical analyzer and returned as tokens. They  
 115656 should not be returned as multi-byte character literals. The token **error** that is used for error  
 115657 recovery is normally assigned the value 256 in the historical implementation. Thus, the token  
 115658 value 256, which is used in many multi-byte character sets, is not available for use as the value  
 115659 of a user-defined token.

#### 115660 FUTURE DIRECTIONS

115661 None.

#### 115662 SEE ALSO

115663 *c99*, *lex*

115664 XBD [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 215)

#### 115665 CHANGE HISTORY

115666 First released in Issue 2.

#### 115667 Issue 5

115668 The FUTURE DIRECTIONS section is added.

#### 115669 Issue 6

115670 This utility is marked as part of the C-Language Development Utilities option.

115671 Minor changes have been added to align with the IEEE P1003.2b draft standard.

- 115672 The normative text is reworded to avoid use of the term “must” for application requirements.
- 115673 IEEE PASC Interpretation 1003.2 #177 is applied, changing the comment on **RCURL** from the }%  
115674 token to the %}.
- 115675 **Issue 7**
- 115676 Austin Group Interpretation 1003.1-2001 #190 is applied, clarifying the requirements for  
115677 generated code to conform to the ISO C standard.
- 115678 Austin Group Interpretation 1003.1-2001 #191 is applied, clarifying the handling of C-language  
115679 trigraphs and curly brace preprocessing tokens.
- 115680 SD5-XCU-ERN-6 is applied, clarifying that Guideline 9 of the Utility Syntax Guidelines does not  
115681 apply.
- 115682 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

115683 **NAME**

115684           zcat — expand and concatenate data

115685 **SYNOPSIS**115686 XSI        zcat [*file...*]115687 **DESCRIPTION**

115688       The *zcat* utility shall write to standard output the uncompressed form of files that have been  
 115689       compressed using the *compress* utility. It is the equivalent of *uncompress -c*. Input files are not  
 115690       affected.

115691 **OPTIONS**

115692       None.

115693 **OPERANDS**

115694       The following operand shall be supported:

115695       *file*        The pathname of a file previously processed by the *compress* utility. If *file* already  
 115696       has the *.Z* suffix specified, it is used as submitted. Otherwise, the *.Z* suffix is  
 115697       appended to the filename prior to processing.

115698 **STDIN**115699       The standard input shall be used only if no *file* operands are specified, or if a *file* operand is *'-'*.115700 **INPUT FILES**115701       Input files shall be compressed files that are in the format produced by the *compress* utility.115702 **ENVIRONMENT VARIABLES**115703       The following environment variables shall affect the execution of *zcat*:

115704       *LANG*        Provide a default value for the internationalization variables that are unset or null.  
 115705       (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 115706       variables used to determine the values of locale categories.)

115707       *LC\_ALL*      If set to a non-empty string value, override the values of all the other  
 115708       internationalization variables.

115709       *LC\_CTYPE*    Determine the locale for the interpretation of sequences of bytes of text data as  
 115710       characters (for example, single-byte as opposed to multi-byte characters in  
 115711       arguments).

115712       *LC\_MESSAGES* Determine the locale that should be used to affect the format and contents of  
 115713       diagnostic messages written to standard error.  
 115714

115715       *NLSPATH*    Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

115716 **ASYNCHRONOUS EVENTS**

115717       Default.

115718 **STDOUT**

115719       The compressed files given as input shall be written on standard output in their uncompressed  
 115720       form.

115721 **STDERR**

115722       The standard error shall be used only for diagnostic messages.

115723 **OUTPUT FILES**

115724       None.

115725 **EXTENDED DESCRIPTION**

115726       None.

115727 **EXIT STATUS**

115728       The following exit values shall be returned:

115729       0   Successful completion.

115730       &gt;0   An error occurred.

115731 **CONSEQUENCES OF ERRORS**

115732       Default.

115733 **APPLICATION USAGE**

115734       None.

115735 **EXAMPLES**

115736       None.

115737 **RATIONALE**

115738       None.

115739 **FUTURE DIRECTIONS**

115740       None.

115741 **SEE ALSO**115742       *compress, uncompress*115743       XBD [Chapter 8](#) (on page 173)115744 **CHANGE HISTORY**

115745       First released in Issue 4.





115746

# *Technical Standard*

115747

**Vol. 4:**

115748

**Rationale (Informative), Issue 7**

115749

*The Open Group*

115750

*The Institute of Electrical and Electronics Engineers, Inc.*



115751

# *Rationale (Informative)*

115752

## **Part A:**

115753

## **Base Definitions**

115754

*The Open Group*

115755

*The Institute of Electrical and Electronics Engineers, Inc.*



# Rationale for Base Definitions

## 115758 A.1 Introduction

### 115759 A.1.1 Scope

115760 POSIX.1-2008 is one of a family of standards known as POSIX. The family of standards extends  
 115761 to many topics; POSIX.1 consists of both operating system interfaces and shell and utilities.  
 115762 POSIX.1-2008 is technically identical to The Open Group Base Specifications, Issue 7.

#### 115763 Scope of POSIX.1-2008

115764 The (paraphrased) goals of this development were to revise the single document that is ISO/IEC  
 115765 9945:2003 Parts 1 through 4, IEEE Std 1003.1, 2004 Edition, and the appropriate parts of The  
 115766 Open Group Single UNIX Specification, Version 3. This work has been undertaken by the  
 115767 Austin Group, a joint working group of IEEE, The Open Group, and ISO/IEC JTC 1/SC 22.

115768 The following are the base documents in this version:

- 115769 • IEEE Std 1003.1, 2004 Edition
- 115770 • ISO/IEC 9899:1999, Programming Languages — C, including ISO/IEC  
 115771 9899:1999/Cor.1:2001(E), ISO/IEC 9899:1999/Cor.2:2004(E), and ISO/IEC  
 115772 9899:1999/Cor.3
- 115773 • The Open Group Extended API Sets, Parts 1 through 4

115774 This version has addressed the following areas:

- 115775 • Issues raised by Austin Group defect reports, IEEE Interpretations against IEEE Std 1003.1,  
 115776 and ISO/IEC defect reports against ISO/IEC 9945

115777 The repository of interpretations can be accessed at [www.opengroup.org/austin/interps](http://www.opengroup.org/austin/interps).

- 115778 • Issues raised in corrigenda for The Open Group Technical Standards and working group  
 115779 resolutions from The Open Group
- 115780 • Issues arising from ISO TR 24715:2006, Conflicts between POSIX and the LSB

115781 This is a Type 3 informative technical report highlighting differences between the LSB 3.1  
 115782 and the 2004 Edition of this standard.

- 115783 • Changes to make the text self-consistent with the additional material merged

115784 The new material merged has come from the The Open Group Extended API Sets, Parts 1  
 115785 through 4. A list of the new interfaces is included in [Section B.1.1](#) (on page 3521).

- 115786 • Features, marked legacy or obsolescent in the base documents, have been considered for  
 115787 removal in this version

115788 See [Section B.1.1](#) (on page 3521) and [Section C.1.1](#) (on page 3665).

- A review and reorganization of the options within the standard

This has included marking the following options obsolescent:

- Batch Environment Services and Utilities
- Tracing
- XSI STREAMS

The UUCP Utilities option is a new option for this version.

Functionality from the following former options is now mandatory in this version:

|     |                                                        |
|-----|--------------------------------------------------------|
| AIO | _POSIX_ASYNCHRONOUS_IO (Asynchronous Input and Output) |
| BAR | _POSIX_BARRIERS (Barriers)                             |
| CS  | _POSIX_CLOCK_SELECTION (Clock Selection)               |
| MF  | _POSIX_MAPPED_FILES (Memory Mapped Files)              |
| MPR | _POSIX_MEMORY_PROTECTION (Memory Protection)           |
| RTS | _POSIX_REALTIME_SIGNALS (Realtime Signals Extension)   |
| RWL | _POSIX_READER_WRITER_LOCKS (Read-Write Locks)          |
| SEM | _POSIX_SEMAPHORES (Semaphores)                         |
| SPI | _POSIX_SPIN_LOCKS (Spin Locks)                         |
| THR | _POSIX_THREADS (Threads)                               |
| TMO | _POSIX_TIMEOUTS (Timeouts)                             |
| TMR | _POSIX_TIMERS (Timers)                                 |
| TSF | _POSIX_THREAD_SAFE_FUNCTIONS (Thread-Safe Functions)   |

- Alignment with the ISO/IEC 9899:1999 standard, including ISO/IEC 9899:1999/Cor.2:2004(E)

- A review of the use of fixed path filenames within the standard

For example, the *at*, *batch*, and *crontab* utilities previously had a requirement for use of the directory **/usr/lib/cron**.

The following were requirements on POSIX.1-2008:

- Backward-compatibility

For interfaces carried forward, it was agreed that there should be no breakage of functionality in the existing base documents. All strictly conforming applications will be conforming but not necessarily strictly conforming to the revised standard. The goal is for system implementations to be able to support the existing and revised standards simultaneously.

- Architecture and *n*-bit-neutral

The common standard should not make any implicit assumptions about the system architecture or size of data types; for example, previously some 32-bit implicit assumptions had crept into the standards.

- Extensibility

It should be possible to extend the common standard without breaking backwards-compatibility; for example, the name space should be reserved and structured to avoid duplication of names between the standard and extensions to it.

**POSIX.1 and the ISO C Standard**

The standard developers believed it essential for a programmer to have a single complete reference place, but recognized that deference to the formal standard has to be addressed for the duplicate interface definitions between the ISO C standard and POSIX.1-2008.

Where an interface has a version in the ISO C standard, the DESCRIPTION section describes the relationship to the ISO C standard and markings are included as appropriate to show where the ISO C standard has been extended in the text.

A block of text is included at the start of each affected reference page stating whether the page is aligned with the ISO C standard or extended. Each page has been parsed for additions beyond the ISO C standard (that is, including both POSIX and UNIX extensions), and these extensions are marked as CX extensions (for C extensions).

**FIPS Requirements**

The Federal Information Processing Standards (FIPS) are a series of US government procurement standards managed and maintained on behalf of the US Department of Commerce by the National Institute of Standards and Technology (NIST).

The following restrictions were integrated into IEEE Std 1003.1-2001. They originally came from FIPS 151-2 which was withdrawn by NIST on February 25 2000.

- The implementation supports `_POSIX_CHOWN_RESTRICTED`.
- The limit `{NGROUPS_MAX}` is greater than or equal to 8.
- The implementation supports the setting of the group ID of a file (when it is created) to that of the parent directory.
- The implementation supports `_POSIX_SAVED_IDS`.
- The implementation supports `_POSIX_VDISABLE`.
- The implementation supports `_POSIX_JOB_CONTROL`.
- The implementation supports `_POSIX_NO_TRUNC`.
- The `read()` function returns the number of bytes read when interrupted by a signal and does not return `-1`.
- The `write()` function returns the number of bytes written when interrupted by a signal and does not return `-1`.
- In the environment for the login shell, the environment variables `LOGNAME` and `HOME` are defined and have the properties described in POSIX.1-2008.
- The value of `{CHILD_MAX}` is greater than or equal to 25.
- The value of `{OPEN_MAX}` is greater than or equal to 20.
- The implementation supports the functionality associated with the symbols `CS7`, `CS8`, `CSTOPB`, `PARODD`, and `PARENB` defined in **<termios.h>**.

115864 **A.1.2 Conformance**115865 See [Section A.2](#) (on page 3445).115866 **A.1.3 Normative References**

115867 There is no additional rationale provided for this section.

115868 **A.1.4 Change History**

115869 For Issue 7 onwards, in references to Technical Corrigenda, the original Austin Group defect  
 115870 report numbers that gave rise to the change are included in square brackets after the change  
 115871 number from the Technical Corrigendum. For more information on Austin Group defect reports  
 115872 see [www.opengroup.org/austin/defectform.html](http://www.opengroup.org/austin/defectform.html).

115873 **A.1.5 Terminology**

115874 The meanings specified in POSIX.1-2008 for the words *shall*, *should*, and *may* are mandated by  
 115875 ISO/IEC directives.

115876 In the Rationale (Informative) volume of POSIX.1-2008, the words *shall*, *should*, and *may* are  
 115877 sometimes used to illustrate similar usages in POSIX.1-2008. However, the rationale itself does  
 115878 not specify anything regarding implementations or applications.

115879 **conformance document**

115880 As a practical matter, the conformance document is effectively part of the system  
 115881 documentation. Conformance documents are distinguished by POSIX.1-2008 so that they  
 115882 can be referred to distinctly.

115883 **implementation-defined**

115884 This definition is analogous to that of the ISO C standard and, together with “undefined”  
 115885 and “unspecified”, provides a range of specification of freedom allowed to the interface  
 115886 implementor.

115887 **may**

115888 The use of *may* has been limited as much as possible, due both to confusion stemming from  
 115889 its ordinary English meaning and to objections regarding the desirability of having as few  
 115890 options as possible and those as clearly specified as possible.

115891 The usage of *can* and *may* were selected to contrast optional application behavior (can)  
 115892 against optional implementation behavior (may).

115893 **shall**

115894 Declarative sentences are sometimes used in POSIX.1-2008 as if they included the word  
 115895 *shall*, and facilities thus specified are no less required. For example, the two statements:

- 115896 1. The *foo()* function shall return zero.
- 115897 2. The *foo()* function returns zero.

115898 are meant to be exactly equivalent.

115899 **should**

115900 In POSIX.1-2008, the word *should* does not usually apply to the implementation, but rather  
 115901 to the application. Thus, the important words regarding implementations are *shall*, which  
 115902 indicates requirements, and *may*, which indicates options.



**obsolescent**

The term “obsolescent” means “do not use this feature in new applications”. A feature noted as obsolescent is supported by all implementations, but may be removed in a future version; new applications should not use these features. The obsolescence concept is not an ideal solution, but was used as a method of increasing consensus: many more objections would be heard from the user community if some of these historical features were suddenly removed without the grace period obsolescence implies. The phrase “may be removed in a future version” implies that the result of that consideration might in fact keep those features indefinitely if the predominance of applications do not migrate away from them quickly.

**legacy**

The term “legacy” was included in earlier versions of this standard but is no longer used in the current version.

**system documentation**

The system documentation should normally describe the whole of the implementation, including any extensions provided by the implementation. Such documents normally contain information at least as detailed as the specifications in POSIX.1-2008. Few requirements are made on the system documentation, but the term is needed to avoid a dangling pointer where the conformance document is permitted to point to the system documentation.

**undefined**

See *implementation-defined*.

**unspecified**

See *implementation-defined*.

The definitions for “unspecified” and “undefined” appear nearly identical at first examination, but are not. The term “unspecified” means that a conforming application may deal with the unspecified behavior, and it should not care what the outcome is. The term “undefined” says that a conforming application should not do it because no definition is provided for what it does (and implicitly it would care what the outcome was if it tried it). It is important to remember, however, that if the syntax permits the statement at all, it must have some outcome in a real implementation.

Thus, the terms “undefined” and “unspecified” apply to the way the application should think about the feature. In terms of the implementation, it is always “defined”—there is always some result, even if it is an error. The implementation is free to choose the behavior it prefers.

This also implies that an implementation, or another standard, could specify or define the result in a useful fashion. The terms apply to POSIX.1-2008 specifically.

The term “implementation-defined” implies requirements for documentation that are not required for “undefined” (or “unspecified”). Where there is no need for a conforming program to know the definition, the term “undefined” is used, even though “implementation-defined” could also have been used in this context. There could be a fourth term, specifying “this standard does not say what this does; it is acceptable to define it in an implementation, but it does not need to be documented”, and undefined would then be used very rarely for the few things for which any definition is not useful. In particular, implementation-defined is used where it is believed that certain classes of application will need to know such details to determine whether the application can be successfully ported to the implementation. Such applications are not always strictly portable, but nevertheless are common and useful; often the requirements met by the application cannot be met without dealing with the issues implied by “implementation-defined”. In some places the text refers to facilities supplied by the implementation that are

115952 outside the standard as implementation-supplied or implementation-provided. This is not  
 115953 intended to imply a requirement for documentation. If it were, the term “implementation-  
 115954 defined” would have been used.

115955 In many places POSIX.1-2008 is silent about the behavior of some possible construct. For  
 115956 example, a variable may be defined for a specified range of values and behaviors are  
 115957 described for those values; nothing is said about what happens if the variable has any other  
 115958 value. That kind of silence can imply an error in the standard, but it may also imply that the  
 115959 standard was intentionally silent and that any behavior is permitted. There is a natural  
 115960 tendency to infer that if the standard is silent, a behavior is prohibited. That is not the intent.  
 115961 Silence is intended to be equivalent to the term “unspecified”.

115962 The term “application” is not defined in POSIX.1-2008; it is assumed to be a part of general  
 115963 computer science terminology.

115964 Three terms used within POSIX.1-2008 overlap in meaning: “macro”, “symbolic name”, and  
 115965 “symbolic constant”.

#### 115966 **macro**

115967 This usually describes a C preprocessor symbol, the result of the **#define** operator, with or  
 115968 without an argument. It may also be used to describe similar mechanisms in editors and  
 115969 text processors.

#### 115970 **symbolic name**

115971 In earlier versions of this standard this was also sometimes used to refer to a C preprocessor  
 115972 symbol (without arguments), but the intention is for all such uses to have been removed. It  
 115973 is now mainly used to refer to the names for characters in character sets, but is sometimes  
 115974 used to refer to host names and even filenames.

#### 115975 **symbolic constant**

115976 This also refers to a C preprocessor symbol, with specific associated requirements. See the  
 115977 definition in [Section 3.374](#) (on page 93).

### 115978 **A.1.6 Definitions and Concepts**

115979 There is no additional rationale provided for this section.

### 115980 **A.1.7 Portability**

115981 To aid the identification of options within POSIX.1-2008, a notation consisting of margin codes  
 115982 and shading is used. This is based on the notation used in earlier versions of The Open Group  
 115983 Base specifications.

115984 The benefit of this approach is a reduction in the number of *if* statements within the running  
 115985 text, that makes the text easier to read, and also an identification to the programmer that they  
 115986 need to ensure that their target platforms support the underlying options. For example, if  
 115987 functionality is marked with RPP in the margin, it will be available on all systems supporting  
 115988 the Robust Mutex Priority Protection option, but may not be available on some others.

## 115989 A.1.7.1 Codes

115990 This section includes codes for options defined in XBD [Section 2.1.6](#) (on page 26), and the  
 115991 following additional codes for other purposes:

115992 CX This margin code is used to denote extensions beyond the ISO C standard. For  
 115993 interfaces that are duplicated between POSIX.1-2008 and the ISO C standard, a CX  
 115994 introduction block describes the nature of the duplication, with any extensions  
 115995 appropriately CX marked and shaded.

115996 Where an interface is added to an ISO C standard header, within the header the  
 115997 interface has an appropriate margin marker and shading (for example, CX, XSI, TSE,  
 115998 and so on) and the same marking appears on the reference page in the SYNOPSIS  
 115999 section. This enables a programmer to easily identify that the interface is extending an  
 116000 ISO C standard header.

## 116001 MX and MXX

116002 These two margin codes both relate to the IEC 60559 Floating-Point option. The MX  
 116003 code denotes functionality that is mandated by the ISO C standard for IEC 60559  
 116004 implementations; the MXX code denotes IEC 60559 functionality that is an extension to  
 116005 the ISO C standard.

116006 OB This margin code is used to denote obsolescent behavior and thus flag a possible future  
 116007 applications portability warning.

116008 OH The Single UNIX Specification has historically tried to reduce the number of headers an  
 116009 application has had to include when using a particular interface. Sometimes this was  
 116010 fewer than the base standard, and hence a notation is used to flag which headers are  
 116011 optional if you are using a system supporting the XSI option.

## 116012 A.1.7.2 Margin Code Notation

116013 Since some features may depend on one or more options, or require more than one option, a  
 116014 notation is used. Where a feature requires support of a single option, a single margin code will  
 116015 occur in the margin. If it depends on two options and both are required, then the codes will  
 116016 appear with a <space> separator. If either of two options are required, then a logical OR is  
 116017 denoted using the ' | ' symbol. If more than two codes are used, a special notation is used.

## 116018 A.2 Conformance

116019 The terms “profile” and “profiling” are used throughout this section.

116020 A profile of a standard or standards is a codified set of option selections, such that by being  
 116021 conformant to a profile, particular classes of users are specifically supported.

## 116022 A.2.1 Implementation Conformance

116023 These definitions allow application developers to know what to depend on in an  
 116024 implementation.

116025 There is no definition of a “strictly conforming implementation”; that would be an  
 116026 implementation that provides *only* those facilities specified by POSIX.1 with no extensions  
 116027 whatsoever. This is because no actual operating system implementation can exist without  
 116028 system administration and initialization facilities that are beyond the scope of POSIX.1.

## 116029 A.2.1.1 Requirements

116030 The word “support” is used in certain instances, rather than “provide”, in order to allow an  
116031 implementation that has no resident software development facilities, but that supports the  
116032 execution of a *Strictly Conforming POSIX.1 Application*, to be a *conforming implementation*.

## 116033 A.2.1.2 Documentation

116034 The conformance documentation is required to use the same numbering scheme as POSIX.1 for  
116035 purposes of cross-referencing. All options that an implementation chooses are reflected in  
116036 **<limits.h>** and **<unistd.h>**.

116037 Note that the use of “may” in terms of where conformance documents record where  
116038 implementations may vary, implies that it is not required to describe those features identified as  
116039 undefined or unspecified.

116040 Other aspects of systems must be evaluated by purchasers for suitability. Many systems  
116041 incorporate buffering facilities, maintaining updated data in volatile storage and transferring  
116042 such updates to non-volatile storage asynchronously. Various exception conditions, such as a  
116043 power failure or a system crash, can cause this data to be lost. The data may be associated with a  
116044 file that is still open, with one that has been closed, with a directory, or with any other internal  
116045 system data structures associated with permanent storage. This data can be lost, in whole or  
116046 part, so that only careful inspection of file contents could determine that an update did not  
116047 occur.

116048 Also, interrelated file activities, where multiple files and/or directories are updated, or where  
116049 space is allocated or released in the file system structures, can leave inconsistencies in the  
116050 relationship between data in the various files and directories, or in the file system itself. Such  
116051 inconsistencies can break applications that expect updates to occur in a specific sequence, so that  
116052 updates in one place correspond with related updates in another place.

116053 For example, if a user creates a file, places information in the file, and then records this action in  
116054 another file, a system or power failure at this point followed by restart may result in a state in  
116055 which the record of the action is permanently recorded, but the file created (or some of its  
116056 information) has been lost. The consequences of this to the user may be undesirable. For a user  
116057 on such a system, the only safe action may be to require the system administrator to have a  
116058 policy that requires, after any system or power failure, that the entire file system must be  
116059 restored from the most recent backup copy (causing all intervening work to be lost).

116060 The characteristics of each implementation will vary in this respect and may or may not meet  
116061 the requirements of a given application or user. Enforcement of such requirements is beyond the  
116062 scope of POSIX.1. It is up to the purchaser to determine what facilities are provided in an  
116063 implementation that affect the exposure to possible data or sequence loss, and also what  
116064 underlying implementation techniques and/or facilities are provided that reduce or limit such  
116065 loss or its consequences.

## 116066 A.2.1.3 POSIX Conformance

116067 This really means conformance to the base standard; however, since this document includes the  
116068 core material of the Single UNIX Specification, the standard developers decided that it was  
116069 appropriate to segment the conformance requirements into two, the former for the base  
116070 standard, and the latter for the Single UNIX Specification (denoted XSI Conformance).

116071 Within POSIX.1 there are some symbolic constants that, if defined to a certain value or range of  
116072 values, indicate that a certain option is enabled. Other symbolic constants exist in POSIX.1 for  
116073 other reasons.

116074 In this version, some features that were previously optional have been made mandatory. For  
 116075 backwards compatibility, the symbolic constants associated with the option are still required  
 116076 now with fixed allowable ranges or values. The following options from the previous version of  
 116077 this standard are now mandatory:

116078        \_POSIX\_ASYNCHRONOUS\_IO  
 116079        \_POSIX\_BARRIERS  
 116080        \_POSIX\_CLOCK\_SELECTION  
 116081        \_POSIX\_MAPPED\_FILES  
 116082        \_POSIX\_MEMORY\_PROTECTION  
 116083        \_POSIX\_READER\_WRITER\_LOCKS  
 116084        \_POSIX\_REALTIME\_SIGNALS  
 116085        \_POSIX\_SEMAPHORES  
 116086        \_POSIX\_SPIN\_LOCKS  
 116087        \_POSIX\_THREAD\_SAFE\_FUNCTIONS  
 116088        \_POSIX\_THREADS  
 116089        \_POSIX\_TIMEOUTS  
 116090        \_POSIX\_TIMERS

116091 A POSIX-conformant system may support the XSI option required by the Single UNIX  
 116092 Specification. This was intentional since the standard developers intend them to be upwards-  
 116093 compatible, so that a system conforming to the Single UNIX Specification can also conform to  
 116094 the base standard at the same time.

#### 116095 A.2.1.4 XSI Conformance

116096 This section is included to describe the conformance requirements for the base volumes of the  
 116097 Single UNIX Specification.

116098 XSI conformance can be thought of as a profile, selecting certain options from POSIX.1-2008.

#### 116099 A.2.1.5 Option Groups

116100 The concept of “Option Groups” is included to allow collections of related functions or options  
 116101 to be grouped together. This has been used as follows: the “XSI Option Groups” have been  
 116102 created to allow super-options, collections of underlying options and related functions, to be  
 116103 collectively supported by XSI-conforming systems.

116104 The standard developers considered the matter of subprofiling and decided it was better to  
 116105 include an enabling mechanism rather than detailed normative requirements. A set of  
 116106 subprofiling options was developed and included later in this volume of POSIX.1-2008 as an  
 116107 informative illustration.

#### 116108 Subprofiling Considerations

116109 The goal of not simultaneously fixing maximums and minimums was to allow implementations  
 116110 of the base standard or standards to support multiple profiles without conflict.

The following summarizes the rules for the limit types:

| Limit Type       | Fixed Value                          | Minimum Acceptable Value                            | Maximum Acceptable Value                            |
|------------------|--------------------------------------|-----------------------------------------------------|-----------------------------------------------------|
| Standard Profile | $X_s$<br>$X_p == X_s$<br>(No change) | $Y_s$<br>$Y_p \geq Y_s$<br>(May increase the limit) | $Z_s$<br>$Z_p \leq Z_s$<br>(May decrease the limit) |

The intent is that ranges specified by limits in profiles be entirely contained within the corresponding ranges of the base standard or standards being profiled, and that the unlimited end of a range in a base standard must remain unlimited in any profile of that standard.

Thus, the fixed `_POSIX_*` limits are constants and must not be changed by a profile. The variable counterparts (typically without the leading `_POSIX_`) can be changed but still remain semantically the same; that is, they still allow implementation values to vary as long as they meet the requirements for that value (be it a minimum or maximum).

Where a profile does not provide a feature upon which a limit is based, the limit is not relevant. Applications written to that profile should be written to operate independently of the value of the limit.

An example which has previously allowed implementations to support both the base standard and two other profiles in a compatible manner follows:

```
Base standard (POSIX.1-1996): _POSIX_CHILD_MAX 6
Base standard: CHILD_MAX minimum maximum _POSIX_CHILD_MAX
FIPS profile/SUSv2 CHILD_MAX 25 (minimum maximum)
```

Another example:

```
Base standard (POSIX.1-1996): _POSIX_NGROUPS_MAX 0
Base standard: NGROUPS_MAX minimum maximum _POSIX_NGROUP_MAX
FIPS profile/SUSv2 NGROUPS_MAX 8
```

A profile may lower a minimum maximum below the equivalent `_POSIX` value:

```
Base standard: _POSIX_foo_MAX Z
Base standard: foo_MAX _POSIX_foo_MAX
profile standard : foo_MAX X (X can be less than, equal to,
 or greater than _POSIX_foo_MAX)
```

In this case an implementation conforming to the profile may not conform to the base standard, but an implementation to the base standard will conform to the profile.

#### A.2.1.6 Options

The final subsections within *Implementation Conformance* list the core options within POSIX.1-2008. This includes both options for the System Interfaces volume of POSIX.1-2008 and the Shell and Utilities volume of POSIX.1-2008.

## 116147 A.2.2 Application Conformance

116148 These definitions guide users or adapters of applications in determining on which  
116149 implementations an application will run and how much adaptation would be required to make  
116150 it run on others. These definitions are modeled after related ones in the ISO C standard.

116151 POSIX.1 occasionally uses the expressions “portable application” or “conforming application”.  
116152 As they are used, these are synonyms for any of these terms. The differences between the classes  
116153 of application conformance relate to the requirements for other standards, the options supported  
116154 (such as the XSI option) or, in the case of the Conforming POSIX.1 Application Using Extensions,  
116155 to implementation extensions. When one of the less explicit expressions is used, it should be  
116156 apparent from the context of the discussion which of the more explicit names is appropriate

### 116157 A.2.2.1 Strictly Conforming POSIX Application

116158 This definition is analogous to that of an ISO C standard “conforming program”.

116159 The major difference between a Strictly Conforming POSIX Application and an ISO C standard  
116160 strictly conforming program is that the latter is not allowed to use features of POSIX that are not  
116161 in the ISO C standard.

### 116162 A.2.2.2 Conforming POSIX Application

116163 Examples of <National Bodies> include ANSI, BSI, and AFNOR.

### 116164 A.2.2.3 Conforming POSIX Application Using Extensions

116165 Due to possible requirements for configuration or implementation characteristics in excess of the  
116166 specifications in <limits.h> or related to the hardware (such as array size or file space), not  
116167 every Conforming POSIX Application Using Extensions will run on every conforming  
116168 implementation.

### 116169 A.2.2.4 Strictly Conforming XSI Application

116170 This is intended to be upwards-compatible with the definition of a Strictly Conforming POSIX  
116171 Application, with the addition of the facilities and functionality included in the XSI option.

### 116172 A.2.2.5 Conforming XSI Application Using Extensions

116173 Such applications may use extensions beyond the facilities defined by POSIX.1-2008 including  
116174 the XSI option, but need to document the additional requirements.

## 116175 A.2.3 Language-Dependent Services for the C Programming Language

116176 POSIX.1 is, for historical reasons, both a specification of an operating system interface, shell and  
116177 utilities, and a C binding for that specification. Efforts had been previously undertaken to  
116178 generate a language-independent specification; however, that had failed, and the fact that the  
116179 ISO C standard is the *de facto* primary language on POSIX and the UNIX system makes this a  
116180 necessary and workable situation.

#### 116181 **A.2.4 Other Language-Related Specifications**

116182 There is no additional rationale provided for this section.

### 116183 **A.3 Definitions**

116184 The definitions in this section are stated so that they can be used as exact substitutes for the  
116185 terms in text. They should not contain requirements or cross-references to sections within  
116186 POSIX.1-2008; that is accomplished by using an informative note. In addition, the term should  
116187 not be included in its own definition. Where requirements or descriptions need to be addressed  
116188 but cannot be included in the definitions, due to not meeting the above criteria, these occur in  
116189 the General Concepts chapter.

116190 In this version, the definitions have been reworked extensively to meet style requirements and to  
116191 include terms from the base documents (see the Scope).

116192 Many of these definitions are necessarily circular, and some of the terms (such as “process”) are  
116193 variants of basic computing science terms that are inherently hard to define. Where some  
116194 definitions are more conceptual and contain requirements, these appear in the General Concepts  
116195 chapter. Those listed in this section appear in an alphabetical glossary format of terms.

116196 Some definitions must allow extension to cover terms or facilities that are not explicitly  
116197 mentioned in POSIX.1-2008. For example, the definition of “Extended Security Controls”  
116198 permits implementations beyond those defined in POSIX.1-2008.

116199 Some terms in the following list of notes do not appear in POSIX.1-2008; these are marked  
116200 suffixed with an asterisk (\*). Many of them have been specifically excluded from POSIX.1-2008  
116201 because they concern system administration, implementation, or other issues that are not  
116202 specific to the programming interface. Those are marked with a reason, such as  
116203 “implementation-defined”.

#### 116204 **Appropriate Privileges**

116205 One of the fundamental security problems with many historical UNIX systems has been that the  
116206 privilege mechanism is monolithic—a user has either no privileges or *all* privileges. Thus, a  
116207 successful “trojan horse” attack on a privileged process defeats all security provisions.  
116208 Therefore, POSIX.1 allows more granular privilege mechanisms to be defined. For many  
116209 historical implementations of the UNIX system, the presence of the term “appropriate  
116210 privileges” in POSIX.1 may be understood as a synonym for “superuser” (UID 0). However,  
116211 other systems have emerged where this is not the case and each discrete controllable action has  
116212 *appropriate privileges* associated with it. Because this mechanism is implementation-defined, it  
116213 must be described in the conformance document. Although that description affects several parts  
116214 of POSIX.1 where the term “appropriate privilege” is used, because the term “implementation-  
116215 defined” only appears here, the description of the entire mechanism and its effects on these  
116216 other sections belongs in this equivalent section of the conformance document. This is especially  
116217 convenient for implementations with a single mechanism that applies in all areas, since it only  
116218 needs to be described once.



**Base Character\***

The term “Base Character” has been removed, as it was felt that the use of this term within POSIX.1-2008 was common usage English.

**Byte**

The restriction that a byte is now exactly eight bits was a conscious decision by the standard developers. It came about due to a combination of factors, primarily the use of the type `int8_t` within the networking functions and the alignment with the ISO/IEC 9899:1999 standard, where the `intN_t` types are now defined.

According to the ISO/IEC 9899:1999 standard:

- The `[u]intN_t` types must be two’s complement with no padding bits and no illegal values.
- All types (apart from bit fields, which are not relevant here) must occupy an integral number of bytes.
- If a type with width  $W$  occupies  $B$  bytes with  $C$  bits per byte ( $C$  is the value of `{CHAR_BIT}`), then it has  $P$  padding bits where  $P+W=B*C$ .
- Therefore, for `int8_t`  $P=0$ ,  $W=8$ . Since  $B \geq 1$ ,  $C \geq 8$ , the only solution is  $B=1$ ,  $C=8$ .

The standard developers also felt that this was not an undue restriction for the current state-of-the-art for this version of the standard, but recognize that if industry trends continue, a wider character type may be required in the future.

**Character**

The term “character” is used to mean a sequence of one or more bytes representing a single graphic symbol. The deviation in the exact text of the ISO C standard definition for “byte” meets the intent of the rationale of the ISO C standard also clears up the ambiguity raised by the term “basic execution character set”. The octet-minimum requirement is a reflection of the `{CHAR_BIT}` value.

**Child Process**

IEEE Std 1003.1-2001/Cor 2-2004, item XBD/TC2/D6/3 is applied, adding the `vfork()` function to those listed.

**Clock Tick**

The ISO C standard defines a similar interval for use by the `clock()` function. There is no requirement that these intervals be the same. In historical implementations these intervals are different.

**Command**

The terms “command” and “utility” are related but have distinct meanings. Command is defined as “a directive to a shell to perform a specific task”. The directive can be in the form of a single utility name (for example, `ls`), or the directive can take the form of a compound command (for example, `"ls | grep name | pr"`). A utility is a program that can be called by name from a shell. Issuing only the name of the utility to a shell is the equivalent of a one-word command. A utility may be invoked as a separate program that executes in a different process than the command language interpreter, or it may be implemented as a part of the command language interpreter. For example, the `echo` command (the directive to perform a specific task) may be implemented such that the `echo` utility (the logic that performs the task of echoing) is in a separate program; therefore, it is executed in a process that is different from the command

language interpreter. Conversely, the logic that performs the *echo* utility could be built into the command language interpreter; therefore, it could execute in the same process as the command language interpreter.

The terms “tool” and “application” can be thought of as being synonymous with “utility” from the perspective of the operating system kernel. Tools, applications, and utilities historically have run, typically, in processes above the kernel level. Tools and utilities historically have been a part of the operating system non-kernel code and have performed system-related functions, such as listing directory contents, checking file systems, repairing file systems, or extracting system status information. Applications have not generally been a part of the operating system, and they perform non-system-related functions, such as word processing, architectural design, mechanical design, workstation publishing, or financial analysis. Utilities have most frequently been provided by the operating system distributor, applications by third-party software distributors, or by the users themselves. Nevertheless, POSIX.1-2008 does not differentiate between tools, utilities, and applications when it comes to receiving services from the system, a shell, or the standard utilities. (For example, the *xargs* utility invokes another utility; it would be of fairly limited usefulness if the users could not run their own applications in place of the standard utilities.) Utilities are not applications in the sense that they are not themselves subject to the restrictions of POSIX.1-2008 or any other standard—there is no requirement for *grep*, *stty*, or any of the utilities defined here to be any of the classes of conforming applications.

### Column Positions

In most 1-byte character sets, such as ASCII, the concept of column positions is identical to character positions and to bytes. Therefore, it has been historically acceptable for some implementations to describe line folding or tab stops or table column alignment in terms of bytes or character positions. Other character sets pose complications, as they can have internal representations longer than one octet and they can have display characters that have different widths on the terminal screen or printer.

In POSIX.1-2008 the term “column positions” has been defined to mean character—not byte—positions in input files (such as “column position 7 of the FORTRAN input”). Output files describe the column position in terms of the display width of the narrowest printable character in the character set, adjusted to fit the characteristics of the output device. It is very possible that *n* column positions will not be able to hold *n* characters in some character sets, unless all of those characters are of the narrowest width. It is assumed that the implementation is aware of the width of the various characters, deriving this information from the value of *LC\_CTYPE*, and thus can determine how many column positions to allot for each character in those utilities where it is important.

The term “column position” was used instead of the more natural “column” because the latter is frequently used in the different contexts of columns of figures, columns of table values, and so on. Wherever confusion might result, these latter types of columns are referred to as “text columns”.

### Controlling Terminal

The question of which of possibly several special files referring to the terminal is meant is not addressed in POSIX.1. The pathname */dev/tty* is a synonym for the controlling terminal associated with a process.

**116304 Device Number\***

116305 The concept is handled in *stat()* as *ID of device*.

**116306 Direct I/O**

116307 Historically, direct I/O refers to the system bypassing intermediate buffering, but may be  
116308 extended to cover implementation-defined optimizations.

**116309 Directory**

116310 The format of the directory file is implementation-defined and differs radically between  
116311 System V and 4.3 BSD. However, routines (derived from 4.3 BSD) for accessing directories and  
116312 certain constraints on the format of the information returned by those routines are described in  
116313 the **<dirent.h>** header.

**116314 Directory Entry**

116315 Throughout POSIX.1-2008, the term “link” is used (about the *link()* function, for example) in  
116316 describing the objects that point to files from directories.

**116317 Display**

116318 The Shell and Utilities volume of POSIX.1-2008 assigns precise requirements for the terms  
116319 “display” and “write”. Some historical systems have chosen to implement certain utilities  
116320 without using the traditional file descriptor model. For example, the *vi* editor might employ  
116321 direct screen memory updates on a personal computer, rather than a *write()* system call. An  
116322 instance of user prompting might appear in a dialog box, rather than with standard error. When  
116323 the Shell and Utilities volume of POSIX.1-2008 uses the term “display”, the method of  
116324 outputting to the terminal is unspecified; many historical implementations use *termcap* or  
116325 *terminfo*, but this is not a requirement. The term “write” is used when the Shell and Utilities  
116326 volume of POSIX.1-2008 mandates that a file descriptor be used and that the output can be  
116327 redirected. However, it is assumed that when the writing is directly to the terminal (it has not  
116328 been redirected elsewhere), there is no practical way for a user or test suite to determine whether  
116329 a file descriptor is being used. Therefore, the use of a file descriptor is mandated only for the  
116330 redirection case and the implementation is free to use any method when the output is not  
116331 redirected. The verb *write* is used almost exclusively, with the very few exceptions of those  
116332 utilities where output redirection need not be supported: *tabs*, *talk*, *tput*, and *vi*.

**116333 Dot**

116334 The symbolic name *dot* is carefully used in POSIX.1 to distinguish the working directory  
116335 filename from a period or a decimal point.

**116336 Dot-Dot**

116337 Historical implementations permit the use of these filenames without their special meanings.  
116338 Such use precludes any meaningful use of these filenames by a Conforming POSIX.1  
116339 Application. Therefore, such use is considered an extension, the use of which makes an  
116340 implementation non-conforming; see also [Section A.4.12](#) (on page 3478).

**Epoch**

Historically, the origin of UNIX system time was referred to as “00:00:00 GMT, January 1, 1970”. Greenwich Mean Time is actually not a term acknowledged by the international standards community; therefore, this term, “Epoch”, is used to abbreviate the reference to the actual standard, Coordinated Universal Time.

**FIFO Special File**

See [Pipe](#) (on page 3462).

**File**

It is permissible for an implementation-defined file type to be non-readable or non-writable.

**File Classes**

These classes correspond to the historical sets of permission bits. The classes are general to allow implementations flexibility in expanding the access mechanism for more stringent security environments. Note that a process is in one and only one class, so there is no ambiguity.

**Filename**

Filenames are sequences of bytes, not sequences of characters. The only bytes that this standard says cannot appear in any filename are the slash byte and the null byte. This is a side-effect of the fact that no conforming implementations of the standard currently provide a way to pass information specifying the locale associated with strings passed between user-level applications and the kernel. This decision could be revisited if implementations develop a way to associate a locale with the strings passed between kernel space and user space.

Implementations may add other restrictions to the byte sequences allowed in filenames except that any filename consisting of no more than {NAME\_MAX} bytes from the set of characters in the portable filename character set must be allowed.

See [Pathname](#) (on page 3461).

**File System**

Historically, the meaning of this term has been overloaded with two meanings: that of the complete file hierarchy, and that of a mountable subset of that hierarchy; that is, a mounted file system. POSIX.1 uses the term “file system” in the second sense, except that it is limited to the scope of a process (and root directory of a process). This usage also clarifies the domain in which a file serial number is unique.

**Graphic Character**

This definition is made available for those definitions (in particular, *TZ*) which must exclude control characters.

**116374 Group Database**

116375 IEEE Std 1003.1-2001/Cor 2-2004, item XBD/TC2/D6/4 is applied, removing the words “of  
116376 implementation-defined format”. See [User Database](#) (on page 3471).

**116377 Group File\***

116378 Implementation-defined; see [User Database](#) (on page 3471).

**116379 Historical Implementations\***

116380 This refers to previously existing implementations of programming interfaces and operating  
116381 systems that are related to the interface specified by POSIX.1.

**116382 Hosted Implementation\***

116383 This refers to a POSIX.1 implementation that is accomplished through interfaces from the  
116384 POSIX.1 services to some alternate form of operating system kernel services. Note that the line  
116385 between a hosted implementation and a native implementation is blurred, since most  
116386 implementations will provide some services directly from the kernel and others through some  
116387 indirect path. (For example, *fopen()* might use *open()*; or *mkfifo()* might use *mknod()*.) There is  
116388 no necessary relationship between the type of implementation and its correctness, performance,  
116389 and/or reliability.

**116390 Implementation\***

116391 This term is generally used instead of its synonym, “system”, to emphasize the consequences of  
116392 decisions to be made by system implementors. Perhaps if no options or extensions to POSIX.1  
116393 were allowed, this usage would not have occurred.

116394 The term “specific implementation” is sometimes used as a synonym for “implementation”.  
116395 This should not be interpreted too narrowly; both terms can represent a relatively broad group  
116396 of systems. For example, a hardware vendor could market a very wide selection of systems that  
116397 all used the same instruction set, with some systems desktop models and others large multi-user  
116398 minicomputers. This wide range would probably share a common POSIX.1 operating system,  
116399 allowing an application compiled for one to be used on any of the others; this is a [*specific*]  
116400 *implementation*. However, such a wide range of machines probably has some differences  
116401 between the models. Some may have different clock rates, different file systems, different  
116402 resource limits, different network connections, and so on, depending on their sizes or intended  
116403 usages. Even on two identical machines, the system administrators may configure them  
116404 differently. Each of these different systems is known by the term “a specific instance of a specific  
116405 implementation”. This term is only used in the portions of POSIX.1 dealing with runtime  
116406 queries: *sysconf()* and *pathconf()*.

**116407 Incomplete Pathname\***

116408 Absolute pathname has been adequately defined.

**Job Control**

In order to understand the job control facilities in POSIX.1 it is useful to understand how they are used by a job control-cognizant shell to create the user interface effect of job control.

While the job control facilities supplied by POSIX.1 can, in theory, support different types of interactive job control interfaces supplied by different types of shells, there was historically one particular interface that was most common when the standard was originally developed (provided by BSD C Shell).

This discussion describes that interface as a means of illustrating how the POSIX.1 job control facilities can be used.

Job control allows users to selectively stop (suspend) the execution of processes and continue (resume) their execution at a later point. The user typically employs this facility via the interactive interface jointly supplied by the terminal I/O driver and a command interpreter (shell).

The user can launch jobs (command pipelines) in either the foreground or background. When launched in the foreground, the shell waits for the job to complete before prompting for additional commands. When launched in the background, the shell does not wait, but immediately prompts for new commands.

If the user launches a job in the foreground and subsequently regrets this, the user can type the suspend character (typically set to <control>-Z), which causes the foreground job to stop and the shell to begin prompting for new commands. The stopped job can be continued by the user (via special shell commands) either as a foreground job or as a background job. Background jobs can also be moved into the foreground via shell commands.

If a background job attempts to access the login terminal (controlling terminal), it is stopped by the terminal driver and the shell is notified, which, in turn, notifies the user. (Terminal access includes *read()* and certain terminal control functions, and conditionally includes *write()*.) The user can continue the stopped job in the foreground, thus allowing the terminal access to succeed in an orderly fashion. After the terminal access succeeds, the user can optionally move the job into the background via the suspend character and shell commands.

*Implementing Job Control Shells*

The interactive interface described previously can be accomplished using the POSIX.1 job control facilities in the following way.

The key feature necessary to provide job control is a way to group processes into jobs. This grouping is necessary in order to direct signals to a single job and also to identify which job is in the foreground. (There is at most one job that is in the foreground on any controlling terminal at a time.)

The concept of process groups is used to provide this grouping. The shell places each job in a separate process group via the *setpgid()* function. To do this, the *setpgid()* function is invoked by the shell for each process in the job. It is actually useful to invoke *setpgid()* twice for each process: once in the child process, after calling *fork()* to create the process, but before calling one of the *exec* family of functions to begin execution of the program, and once in the parent shell process, after calling *fork()* to create the child. The redundant invocation avoids a race condition by ensuring that the child process is placed into the new process group before either the parent or the child relies on this being the case. The process group ID for the job is selected by the shell to be equal to the process ID of one of the processes in the job. Some shells choose to make one process in the job be the parent of the other processes in the job (if any). Other shells (for example, the C Shell) choose to make themselves the parent of all processes in the pipeline (job). In order to support this latter case, the *setpgid()* function accepts a process group ID parameter

116456 since the correct process group ID cannot be inherited from the shell. The shell itself is  
116457 considered to be a job and is the sole process in its own process group.

116458 The shell also controls which job is currently in the foreground. A foreground and background  
116459 job differ in two ways: the shell waits for a foreground command to complete (or stop) before  
116460 continuing to read new commands, and the terminal I/O driver inhibits terminal access by  
116461 background jobs (causing the processes to stop). Thus, the shell must work cooperatively with  
116462 the terminal I/O driver and have a common understanding of which job is currently in the  
116463 foreground. It is the user who decides which command should be currently in the foreground,  
116464 and the user informs the shell via shell commands. The shell, in turn, informs the terminal I/O  
116465 driver via the *tcsetpgrp()* function. This indicates to the terminal I/O driver the process group ID  
116466 of the foreground process group (job). When the current foreground job either stops or  
116467 terminates, the shell places itself in the foreground via *tcsetpgrp()* before prompting for  
116468 additional commands. Note that when a job is created the new process group begins as a  
116469 background process group. It requires an explicit act of the shell via *tcsetpgrp()* to move a  
116470 process group (job) into the foreground.

116471 When a process in a job stops or terminates, its parent (for example, the shell) receives  
116472 synchronous notification by calling the *waitpid()* function with the WUNTRACED flag set.  
116473 Asynchronous notification is also provided when the parent establishes a signal handler for  
116474 SIGCHLD and does not specify the SA\_NOCLDSTOP flag. Usually all processes in a job stop as  
116475 a unit since the terminal I/O driver always sends job control stop signals to all processes in the  
116476 process group.

116477 To continue a stopped job, the shell sends the SIGCONT signal to the process group of the job. In  
116478 addition, if the job is being continued in the foreground, the shell invokes *tcsetpgrp()* to place the  
116479 job in the foreground before sending SIGCONT. Otherwise, the shell leaves itself in the  
116480 foreground and reads additional commands.

116481 There is additional flexibility in the POSIX.1 job control facilities that allows deviations from the  
116482 typical interface. Clearing the TOSTOP terminal flag allows background jobs to perform *write()*  
116483 functions without stopping. The same effect can be achieved on a per-process basis by having a  
116484 process set the signal action for SIGTTOU to SIG\_IGN.

116485 Note that the terms “job” and “process group” can be used interchangeably. A login session that  
116486 is not using the job control facilities can be thought of as a large collection of processes that are  
116487 all in the same job (process group). Such a login session may have a partial distinction between  
116488 foreground and background processes; that is, the shell may choose to wait for some processes  
116489 before continuing to read new commands and may not wait for other processes. However, the  
116490 terminal I/O driver will consider all these processes to be in the foreground since they are all  
116491 members of the same process group.

116492 In addition to the basic job control operations already mentioned, a job control-cognizant shell  
116493 needs to perform the following actions.

116494 When a foreground (not background) job stops, the shell must sample and remember the current  
116495 terminal settings so that it can restore them later when it continues the stopped job in the  
116496 foreground (via the *tcgetattr()* and *tcsetattr()* functions).

116497 Because a shell itself can be spawned from a shell, it must take special action to ensure that  
116498 subshells interact well with their parent shells.

116499 A subshell can be spawned to perform an interactive function (prompting the terminal for  
116500 commands) or a non-interactive function (reading commands from a file). When operating non-  
116501 interactively, the job control shell will refrain from performing the job control-specific actions  
116502 described above. It will behave as a shell that does not support job control. For example, all jobs  
116503 will be left in the same process group as the shell, which itself remains in the process group

established for it by its parent. This allows the shell and its children to be treated as a single job by a parent shell, and they can be affected as a unit by terminal keyboard signals.

An interactive subshell can be spawned from another job control-cognizant shell in either the foreground or background. (For example, from the C Shell, the user can execute the command, `csh &`.) Before the subshell activates job control by calling `setpgid()` to place itself in its own process group and `tcsetpgrp()` to place its new process group in the foreground, it needs to ensure that it has already been placed in the foreground by its parent. (Otherwise, there could be multiple job control shells that simultaneously attempt to control mediation of the terminal.) To determine this, the shell retrieves its own process group via `getpgrp()` and the process group of the current foreground job via `tcgetpgrp()`. If these are not equal, the shell sends SIGTTIN to its own process group, causing itself to stop. When continued later by its parent, the shell repeats the process group check. When the process groups finally match, the shell is in the foreground and it can proceed to take control. After this point, the shell ignores all the job control stop signals so that it does not inadvertently stop itself.

#### *Implementing Job Control Applications*

Most applications do not need to be aware of job control signals and operations; the intuitively correct behavior happens by default. However, sometimes an application can inadvertently interfere with normal job control processing, or an application may choose to overtly effect job control in cooperation with normal shell procedures.

An application can inadvertently subvert job control processing by “blindly” altering the handling of signals. A common application error is to learn how many signals the system supports and to ignore or catch them all. Such an application makes the assumption that it does not know what this signal is, but knows the right handling action for it. The system may initialize the handling of job control stop signals so that they are being ignored. This allows shells that do not support job control to inherit and propagate these settings and hence to be immune to stop signals. A job control shell will set the handling to the default action and propagate this, allowing processes to stop. In doing so, the job control shell is taking responsibility for restarting the stopped applications. If an application wishes to catch the stop signals itself, it should first determine their inherited handling states. If a stop signal is being ignored, the application should continue to ignore it. This is directly analogous to the recommended handling of SIGINT described in the referenced UNIX Programmer’s Manual.

If an application is reading the terminal and has disabled the interpretation of special characters (by clearing the ISIG flag), the terminal I/O driver will not send SIGTSTP when the suspend character is typed. Such an application can simulate the effect of the suspend character by recognizing it and sending SIGTSTP to its process group as the terminal driver would have done. Note that the signal is sent to the process group, not just to the application itself; this ensures that other processes in the job also stop. (Note also that other processes in the job could be children, siblings, or even ancestors.) Applications should not assume that the suspend character is `<control>-Z` (or any particular value); they should retrieve the current setting at startup.

#### *Implementing Job Control Systems*

The intent in adding 4.2 BSD-style job control functionality was to adopt the necessary 4.2 BSD programmatic interface with only minimal changes to resolve syntactic or semantic conflicts with System V or to close recognized security holes. The goal was to maximize the ease of providing both conforming implementations and Conforming POSIX.1 Applications.

It is only useful for a process to be affected by job control signals if it is the descendant of a job control shell. Otherwise, there will be nothing that continues the stopped process.

POSIX.1 does not specify how controlling terminal access is affected by a user logging out (that



116552 is, by a controlling process terminating). 4.2 BSD uses the *vhangup()* function to prevent any  
116553 access to the controlling terminal through file descriptors opened prior to logout. System V does  
116554 not prevent controlling terminal access through file descriptors opened prior to logout (except  
116555 for the case of the special file, */dev/tty*). Some implementations choose to make processes  
116556 immune from job control after logout (that is, such processes are always treated as if in the  
116557 foreground); other implementations continue to enforce foreground/background checks after  
116558 logout. Therefore, a Conforming POSIX.1 Application should not attempt to access the  
116559 controlling terminal after logout since such access is unreliable. If an implementation chooses to  
116560 deny access to a controlling terminal after its controlling process exits, POSIX.1 requires a certain  
116561 type of behavior (see [Controlling Terminal](#), on page 3452).

#### 116562 **Kernel\***

116563 See [System Call\\*](#) (on page 3469).

#### 116564 **Library Routine\***

116565 See [System Call\\*](#) (on page 3469).

#### 116566 **Logical Device\***

116567 Implementation-defined.

#### 116568 **Map**

116569 The definition of map is included to clarify the usage of mapped pages in the description of the  
116570 behavior of process memory locking.

#### 116571 **Memory-Resident**

116572 The term “memory-resident” is historically understood to mean that the so-called resident pages  
116573 are actually present in the physical memory of the computer system and are immune from  
116574 swapping, paging, copy-on-write faults, and so on. This is the actual intent of POSIX.1-2008 in  
116575 the process memory locking section for implementations where this is logical. But for some  
116576 implementations—primarily mainframes—actually locking pages into primary storage is not  
116577 advantageous to other system objectives, such as maximizing throughput. For such  
116578 implementations, memory locking is a “hint” to the implementation that the application wishes  
116579 to avoid situations that would cause long latencies in accessing memory. Furthermore, there are  
116580 other implementation-defined issues with minimizing memory access latencies that “memory  
116581 residency” does not address—such as MMU reload faults. The definition attempts to  
116582 accommodate various implementations while allowing conforming applications to specify to the  
116583 implementation that they want or need the best memory access times that the implementation  
116584 can provide.

#### 116585 **Memory Object\***

116586 The term “memory object” usually implies shared memory. If the object is the same as a  
116587 filename in the file system name space of the implementation, it is expected that the data written  
116588 into the memory object be preserved on disk. A memory object may also apply to a physical  
116589 device on an implementation. In this case, writes to the memory object are sent to the controller  
116590 for the device and reads result in control registers being returned.

**Mount Point\***

The directory on which a “mounted file system” is mounted. This term, like *mount()* and *umount()*, was not included because it was implementation-defined.

**Mounted File System\***

See [File System](#) (on page 3454).

**Name**

There are no explicit limits in POSIX.1-2008 on the sizes of names, words (see the definition of word in the Base Definitions volume of POSIX.1-2008), lines, or other objects. However, other implicit limits do apply: shell script lines produced by many of the standard utilities cannot exceed {LINE\_MAX} and the sum of exported variables comes under the {ARG\_MAX} limit. Historical shells dynamically allocate memory for names and words and parse incoming lines a character at a time. Lines cannot have an arbitrary {LINE\_MAX} limit because of historical practice, such as *makefiles*, where *make* removes the <newline> characters associated with the commands for a target and presents the shell with one very long line. The text on INPUT FILES in XCU [Section 1.4](#) (on page 2310) does allow a shell to run out of memory, but it cannot have arbitrary programming limits.

**Native Implementation\***

This refers to an implementation of POSIX.1 that interfaces directly to an operating system kernel; see also *hosted implementation*. A similar concept is a native UNIX system, which would be a kernel derived from one of the original UNIX system products.

**Nice Value**

This definition is not intended to suggest that all processes in a system have priorities that are comparable. Scheduling policy extensions, such as adding realtime priorities, make the notion of a single underlying priority for all scheduling policies problematic. Some implementations may implement the features related to *nice* to affect all processes on the system, others to affect just the general time-sharing activities implied by POSIX.1-2008, and others may have no effect at all. Because of the use of “implementation-defined” in *nice* and *renice*, a wide range of implementation strategies is possible.

**Open File Description**

An “open file description”, as it is currently named, describes how a file is being accessed. What is currently called a “file descriptor” is actually just an identifier or “handle”; it does not actually describe anything.

The following alternate names were discussed:

- For “open file description”:  
“open instance”, “file access description”, “open file information”, and “file access information”.
- For “file descriptor”:  
“file handle”, “file number” (cf., *fileno()*). Some historical implementations use the term “file table entry”.

## 116630 Orphaned Process Group

116631 Historical implementations have a concept of an orphaned process, which is a process whose  
 116632 parent process has exited. When job control is in use, it is necessary to prevent processes from  
 116633 being stopped in response to interactions with the terminal after they no longer are controlled by  
 116634 a job control-cognizant program. Because signals generated by the terminal are sent to a process  
 116635 group and not to individual processes, and because a signal may be provoked by a process that  
 116636 is not orphaned, but sent to another process that is orphaned, it is necessary to define an  
 116637 orphaned process group. The definition assumes that a process group will be manipulated as a  
 116638 group and that the job control-cognizant process controlling the group is outside of the group  
 116639 and is the parent of at least one process in the group (so that state changes may be reported via  
 116640 *waitpid()*). Therefore, a group is considered to be controlled as long as at least one process in the  
 116641 group has a parent that is outside of the process group, but within the session.

116642 This definition of orphaned process groups ensures that a session leader's process group is  
 116643 always considered to be orphaned, and thus it is prevented from stopping in response to  
 116644 terminal signals.

## 116645 Page

116646 The term "page" is defined to support the description of the behavior of memory mapping for  
 116647 shared memory and memory mapped files, and the description of the behavior of process  
 116648 memory locking. It is not intended to imply that shared memory/file mapping and memory  
 116649 locking are applicable only to "paged" architectures. For the purposes of POSIX.1-2008,  
 116650 whatever the granularity on which an architecture supports mapping or locking, this is  
 116651 considered to be a "page". If an architecture cannot support the memory mapping or locking  
 116652 functions specified by POSIX.1-2008 on any granularity, then these options will not be  
 116653 implemented on the architecture.

## 116654 Pathname

116655 Pathnames historically allowed all bytes except for the <slash> and <NUL> characters. For +  
 116656 compatibility with existing file systems, this usage is maintained throughout the standard by +  
 116657 noting that a pathname need not be a valid character string in all locales. However, the +  
 116658 properties of the portable filename character set are such that a pathname using only those +  
 116659 characters and the <slash> is portable in all locales as a character string.

## 116660 Passwd File\*

116661 Implementation-defined; see [User Database](#) (on page 3471).

## 116662 Parent Directory

116663 There may be more than one directory entry pointing to a given directory in some  
 116664 implementations. The wording here identifies that exactly one of those is the parent directory. In  
 116665 pathname resolution, dot-dot is identified as the way that the unique directory is identified.  
 116666 (That is, the parent directory is the one to which dot-dot points.) In the case of a remote file  
 116667 system, if the same file system is mounted several times, it would appear as if they were distinct  
 116668 file systems (with interesting synchronization properties).

**116669 Pipe**

116670 It proved convenient to define a pipe as a special case of a FIFO, even though historically the  
116671 latter was not introduced until System III and does not exist at all in 4.3 BSD.

**116672 Portable Filename Character Set**

116673 The encoding of this character set is not specified—specifically, ASCII is not required. But the  
116674 implementation must provide a unique character code for each of the printable graphics  
116675 specified by POSIX.1; see also [Section A.4.6](#) (on page 3474).

116676 Situations where characters beyond the portable filename character set (or historically ASCII or  
116677 the ISO/IEC 646:1991 standard) would be used (in a context where the portable filename  
116678 character set or the ISO/IEC 646:1991 standard is required by POSIX.1) are expected to be  
116679 common. Although such a situation renders the use technically non-compliant, mutual  
116680 agreement among the users of an extended character set will make such use portable between  
116681 those users. Such a mutual agreement could be formalized as an optional extension to POSIX.1.  
116682 (Making it required would eliminate too many possible systems, as even those systems using the  
116683 ISO/IEC 646:1991 standard as a base character set extend their character sets for Western  
116684 Europe and the rest of the world in different ways.)

116685 Nothing in POSIX.1 is intended to preclude the use of extended characters where interchange is  
116686 not required or where mutual agreement is obtained. It has been suggested that in several places  
116687 “should” be used instead of “shall”. Because (in the worst case) use of any character beyond the  
116688 portable filename character set would render the program or data not portable to all possible  
116689 systems, no extensions are permitted in this context.

**116690 Process Lifetime**

116691 IEEE Std 1003.1-2001/Cor 2-2004, item XBD/TC2/D6/5 is applied, adding *fork()*, *posix\_spawn()*,  
116692 *posix\_spawnp()*, and *vfork()* to the list of functions.

**116693 Process Termination**

116694 IEEE Std 1003.1-2001/Cor 2-2004, item XBD/TC2/D6/6 is applied, rewording the definition to  
116695 address the “passive exit” on termination of the last thread or the *\_Exit()* function.

**116696 Regular File**

116697 POSIX.1 does not intend to preclude the addition of structuring data (for example, record  
116698 lengths) in the file, as long as such data is not visible to an application that uses the features  
116699 described in POSIX.1.

**116700 Root Directory**

116701 This definition permits the operation of *chroot()*, even though that function is not in POSIX.1; see  
116702 also [Section A.4.5](#) (on page 3473).

**116703 Root File System\***

116704 Implementation-defined.

**Root of a File System\***

Implementation-defined; see [Mount Point\\*](#) (on page 3460).

**Signal**

The definition implies a double meaning for the term. Although a signal is an event, common usage implies that a signal is an identifier of the class of event.

**Superuser\***

This concept, with great historical significance to UNIX system users, has been replaced with the notion of appropriate privileges.

**Supplementary Group ID**

The POSIX.1-1990 standard is inconsistent in its treatment of supplementary groups. The definition of supplementary group ID explicitly permits the effective group ID to be included in the set, but wording in the description of the *setuid()* and *setgid()* functions states: “Any supplementary group IDs of the calling process remain unchanged by these function calls”. In the case of *setgid()* this contradicts that definition. In addition, some felt that the unspecified behavior in the definition of supplementary group IDs adds unnecessary portability problems. The standard developers considered several solutions to this problem:

1. Reword the description of *setgid()* to permit it to change the supplementary group IDs to reflect the new effective group ID. A problem with this is that it adds more “may”s to the wording and does not address the portability problems of this optional behavior.
2. Mandate the inclusion of the effective group ID in the supplementary set (giving {NGROUPS\_MAX} a minimum value of 1). This is the behavior of 4.4 BSD. In that system, the effective group ID is the first element of the array of supplementary group IDs (there is no separate copy stored, and changes to the effective group ID are made only in the supplementary group set). By convention, the initial value of the effective group ID is duplicated elsewhere in the array so that the initial value is not lost when executing a set-group-ID program.
3. Change the definition of supplementary group ID to exclude the effective group ID and specify that the effective group ID does not change the set of supplementary group IDs. This is the behavior of 4.2 BSD, 4.3 BSD, and System V Release 4.
4. Change the definition of supplementary group ID to exclude the effective group ID, and require that *getgroups()* return the union of the effective group ID and the supplementary group IDs.
5. Change the definition of {NGROUPS\_MAX} to be one more than the number of supplementary group IDs, so it continues to be the number of values returned by *getgroups()* and existing applications continue to work. This alternative is effectively the same as the second (and might actually have the same implementation).

The standard developers decided to permit either 2 or 3. The effective group ID is orthogonal to the set of supplementary group IDs, and it is implementation-defined whether *getgroups()* returns this. If the effective group ID is returned with the set of supplementary group IDs, then all changes to the effective group ID affect the supplementary group set returned by *getgroups()*. It is permissible to eliminate duplicates from the list returned by *getgroups()*. However, if a group ID is contained in the set of supplementary group IDs, setting the group ID to that value and then to a different value should not remove that value from the supplementary group IDs.

The definition of supplementary group IDs has been changed to not include the effective group

ID. This simplifies permanent rationale and makes the relevant functions easier to understand. The `getgroups()` function has been modified so that it can, on an implementation-defined basis, return the effective group ID. By making this change, functions that modify the effective group ID do not need to discuss adding to the supplementary group list; the only view into the supplementary group list that the application developer has is through the `getgroups()` function.

## Symbolic Constant

Earlier versions of this standard used a variety of terms other than “macro” for many of the constants defined in headers, and it was not clear in which of these cases they were required to be macros or not, or to be pre-processor constants (i.e., usable in `#if`) or not. In cases where the symbols had a reserved prefix or suffix, there was often inconsistency between whether the prefix/suffix was reserved only for macros or for any use, and whether the term “macro” or a different term was used in the descriptions of the symbols. There were also some unintentional differences from the ISO C standard.

One of the most commonly used terms was “symbolic constant”. This has now been designated as the default term to be used wherever appropriate, and a formal definition of the term has been added giving the exact requirements for symbols that are described as symbolic constants.

The standard developers have performed a major rationalization of the header descriptions of symbols with constant values according to the following policy:

- Where symbols are from the ISO C standard, the wording from the ISO C standard (or equivalent, in cases where the exact wording is not appropriate) is used to describe them.
- For all other constants, the first choice is to use “symbolic constant” when the requirements for the symbol are a reasonably close fit with those of the term.

The description of the symbol can override individual requirements for symbolic constants; e.g., to specify a non-integer type, or to add a requirement that the symbol is usable in `#if` preprocessor directives.

- When neither of the above apply, the exact requirements are stated in the description. (Note that macros are not required to be usable in `#if`, or even to expand to constant expressions, unless explicitly stated.)
- In cases where there is a reserved prefix or suffix, if the symbol(s) with that prefix/suffix are from the ISO C standard and are required to be macros, or if the symbol is required to be usable in `#if`, then the prefix/suffix is reserved for use only as macros. If the symbol(s) are “symbolic constants” and not required to be usable in `#if`, the prefix/suffix is reserved for any use except in a few special cases.

Where a constant is required to be a macro but is also allowed to be another type of constant such as an enumeration constant, on implementations which do define it as another type of constant the macro is typically defined as follows:

```
#define macro_name macro_name
```

This allows applications to use `#ifdef`, etc. to determine whether the macro is defined, but the macro is not usable in `#if` preprocessor directives because the preprocessor will treat the unexpanded word `macro_name` as having the value zero.

## Symbolic Link

Earlier versions of this standard did not require symbolic links to have attributes such as ownership and a file serial number. This was because the 4.4 BSD implementation did not have them, and it was expected that other implementations may wish to do the same. However, experience with 4.4 BSD has shown that symbolic links implemented in this way cause problems for users and application developers, and later BSD systems have reverted to using *inodes* to implement symbolic links. Allowing *no-inode* symbolic links also caused problems in the standard. For example, leaving the *st\_ino* value for symbolic links unspecified meant that the common technique of comparing the *st\_dev* and *st\_ino* values for two pathnames to see if they refer to the same file could only be used with *stat()* in conforming applications and not with *lstat()*. The standard now requires symbolic links to have meaningful values for the same **struct stat** fields as regular files, except for the file mode bits in *st\_mode*. Historically, the file mode bits were unused (the contents of a symbolic link could always be read), but implementations differed as to whether the file mode bits (as returned in *st\_mode* or reported by *ls -l*) were set according to the *umask* or just to a fixed value such as 0777. Accordingly, the standard requires the file mode bits to be ignored by *readlink()* and when a symbolic link is followed during pathname resolution, but leaves the corresponding part of the value returned in *st\_mode* unspecified.

Historical implementations were followed when determining which interfaces should apply to symbolic links. Interfaces that historically followed symbolic links include *chmod()*, *stat()*, and *utime()*. Interfaces that historically did not follow symbolic links include *lstat()*, *rename()*, *remove()*, *rmdir()*, and *unlink()*. For *chown()* and *link()*, historical implementations differed. POSIX.1-2008 inherited the *lchown()* function from the Single UNIX Specification, Version 2, and therefore requires *chown()* to follow symbolic links. Earlier versions of this standard required *link()* to follow symbolic links, but with the addition of the *linkat()* function (which has a flag to indicate whether to follow symbolic links), both behaviors are now allowed for *link()*.

When the final component of a pathname is a symbolic link, the standard requires that a trailing *<slash>* causes the link to be followed. This is the behavior of historical implementations. For example, for */a/b* and */a/b/*, if */a/b* is a symbolic link to a directory, then */a/b* refers to the symbolic link, and */a/b/* refers to the directory to which the symbolic link points.

Because a symbolic link and its referenced object coexist in the file system name space, confusion can arise in distinguishing between the link itself and the referenced object. Historically, utilities and system calls have adopted their own link following conventions in a somewhat *ad hoc* fashion. Rules for a uniform approach are outlined here, although historical practice has been adhered to as much as was possible. To promote consistent system use, user-written utilities are encouraged to follow these same rules.

Symbolic links are handled either by operating on the link itself, or by operating on the object referenced by the link. In the latter case, an application or system call is said to “follow” the link. Symbolic links may reference other symbolic links, in which case links are dereferenced until an object that is not a symbolic link is found, a symbolic link that references a file that does not exist is found, or a loop is detected. (Current implementations do not detect loops, but have a limit on the number of symbolic links that they will dereference before declaring it an error.)

There are four domains for which default symbolic link policy is established in a system. In almost all cases, there are utility options that override this default behavior. The four domains are as follows:

1. Symbolic links specified to system calls that take pathname arguments |
2. Symbolic links specified as command line pathname arguments to utilities that are not performing a traversal of a file hierarchy |

3. Symbolic links referencing files not of type directory, specified to utilities that are performing a traversal of a file hierarchy

4. Symbolic links referencing files of type directory, specified to utilities that are performing a traversal of a file hierarchy

#### First Domain

The first domain is considered in earlier rationale.

#### Second Domain

The reason this category is restricted to utilities that are not traversing the file hierarchy is that some standard utilities take an option that specifies a hierarchical traversal, but by default operate on the arguments themselves. Generally, users specifying the option for a file hierarchy traversal wish to operate on a single, physical hierarchy, and therefore symbolic links, which may reference files outside of the hierarchy, are ignored. For example, *chown owner file* is a different operation from the same command with the **-R** option specified. In this example, the behavior of the command *chown owner file* is described here, while the behavior of the command *chown -R owner file* is described in the third and fourth domains.

The general rule is that the utilities in this category follow symbolic links named as arguments.

Exceptions in the second domain are:

- The *mv* and *rm* utilities do not follow symbolic links named as arguments, but respectively attempt to rename or delete them.
- The *ls* utility is also an exception to this rule. For compatibility with historical systems, when the **-R** option is not specified, the *ls* utility follows symbolic links named as arguments if the **-L** option is specified or if the **-F**, **-d**, or **-l** options are not specified. (If the **-L** option is specified, *ls* always follows symbolic links; it is the only utility where the **-L** option affects its behavior even though a tree walk is not being performed.)

All other standard utilities, when not traversing a file hierarchy, always follow symbolic links named as arguments.

Historical practice is that the **-h** option is specified if standard utilities are to act upon symbolic links instead of upon their targets. Examples of commands that have historically had a **-h** option for this purpose are the *chgrp*, *chown*, *file*, and *test* utilities.

#### Third Domain

The third domain is symbolic links, referencing files not of type directory, specified to utilities that are performing a traversal of a file hierarchy. (This includes symbolic links specified as command line pathname arguments or encountered during the traversal.)

The intention of the Shell and Utilities volume of POSIX.1-2008 is that the operation that the utility is performing is applied to the symbolic link itself, if that operation is applicable to symbolic links. If the operation is not applicable to symbolic links, the symbolic link should be ignored. Specifically, by default, no change should be made to the file referenced by the symbolic link.

#### Fourth Domain

The fourth domain is symbolic links referencing files of type directory, specified to utilities that are performing a traversal of a file hierarchy. (This includes symbolic links specified as command line pathname arguments or encountered during the traversal.)

Most standard utilities do not, by default, indirect into the file hierarchy referenced by the symbolic link. (The Shell and Utilities volume of POSIX.1-2008 uses the informal term “physical



walk” to describe this case. The case where the utility does indirect through the symbolic link is termed a “logical walk”.)

There are three reasons for the default to be a physical walk:

1. With very few exceptions, a physical walk has been the historical default on UNIX systems supporting symbolic links. Because some utilities (that is, *rm*) must default to a physical walk, regardless, changing historical practice in this regard would be confusing to users and needlessly incompatible.
2. For systems where symbolic links have the historical file attributes (that is, *owner*, *group*, *mode*), defaulting to a logical traversal would require the addition of a new option to the commands to modify the attributes of the link itself. This is painful and more complex than the alternatives.
3. There is a security issue with defaulting to a logical walk. Historically, the command *chown -R user file* has been safe for the superuser because *setuid* and *setgid* bits were lost when the ownership of the file was changed. If the walk were logical, changing ownership would no longer be safe because a user might have inserted a symbolic link pointing to any file in the tree. Again, this would necessitate the addition of an option to the commands doing hierarchy traversal to not indirect through the symbolic links, and historical scripts doing recursive walks would instantly become security problems. While this is mostly an issue for system administrators, it is preferable to not have different defaults for different classes of users.

However, the standard developers agreed to leave it unspecified to achieve consensus.

As consistently as possible, users may cause standard utilities performing a file hierarchy traversal to follow any symbolic links named on the command line, regardless of the type of file they reference, by specifying the **-H** (for half logical) option. This option is intended to make the command line name space look like the logical name space.

As consistently as possible, users may cause standard utilities performing a file hierarchy traversal to follow any symbolic links named on the command line as well as any symbolic links encountered during the traversal, regardless of the type of file they reference, by specifying the **-L** (for logical) option. This option is intended to make the entire name space look like the logical name space.

For consistency, implementors are encouraged to use the **-P** (for “physical”) flag to specify the physical walk in utilities that do logical walks by default for whatever reason.

When one or more of the **-H**, **-L**, and **-P** flags can be specified, the last one specified determines the behavior of the utility. This permits users to alias commands so that the default behavior is a logical walk and then override that behavior on the command line.

#### *Exceptions in the Third and Fourth Domains*

The *ls* and *rm* utilities are exceptions to these rules. The *rm* utility never follows symbolic links and does not support the **-H**, **-L**, or **-P** options. Some historical versions of *ls* always followed symbolic links given on the command line whether the **-L** option was specified or not. Historical versions of *ls* did not support the **-H** option. In POSIX.1-2008, unless one of the **-H** or **-L** options is specified, the *ls* utility only follows symbolic links to directories that are given as operands. The *ls* utility does not support the **-P** option.

The Shell and Utilities volume of POSIX.1-2008 requires that the standard utilities *ls*, *find*, and *pax* detect infinite loops when doing logical walks; that is, a directory, or more commonly a symbolic link, that refers to an ancestor in the current file hierarchy. If the file system itself is corrupted, causing the infinite loop, it may be impossible to recover. Because *find* and *ls* are often used in system administration and security applications, they should attempt to recover and

116928 continue as best as they can. The *pax* utility should terminate because the archive it was creating  
116929 is by definition corrupted. Other, less vital, utilities should probably simply terminate as well.  
116930 Implementations are strongly encouraged to detect infinite loops in all utilities.

116931 Historical practice is shown in [Table A-1](#) (on page 3469). The heading **SVID3** stands for the  
116932 Third Edition of the System V Interface Definition.

116933 Historically, several shells have had built-in versions of the *pwd* utility. In some of these shells,  
116934 *pwd* reported the physical path, and in others, the logical path. Implementations of the shell  
116935 corresponding to POSIX.1-2008 must report the logical path by default.

116936 The *cd* command is required, by default, to treat the filename dot-dot logically. Implementors are  
116937 required to support the **-P** flag in *cd* so that users can have their current environment handled  
116938 physically. In 4.3 BSD, *chgrp* during tree traversal changed the group of the symbolic link, not  
116939 the target. Symbolic links in 4.4 BSD did not have *owner*, *group*, *mode*, or other standard UNIX  
116940 system file attributes.

116941

**Table A-1** Historical Practice for Symbolic Links

116942

116943

116944

116945

116946

116947

116948

116949

116950

116951

116952

116953

116954

116955

116956

116957

116958

116959

116960

116961

116962

116963

116964

116965

116966

116967

116968

116969

116970

116971

116972

116973

116974

| Utility      | SVID3   | 4.3 BSD | 4.4 BSD | POSIX | Comments                                |
|--------------|---------|---------|---------|-------|-----------------------------------------|
| <i>cd</i>    |         |         |         | –L    | Treat " . . " logically.                |
| <i>cd</i>    |         |         |         | –P    | Treat " . . " physically.               |
| <i>chgrp</i> |         |         | –H      | –H    | Follow command line symlinks.           |
| <i>chgrp</i> |         |         | –h      | –L    | Follow symlinks.                        |
| <i>chgrp</i> | –h      |         |         | –h    | Affect the symlink.                     |
| <i>chmod</i> |         |         |         |       | Affect the symlink.                     |
| <i>chmod</i> |         |         | –H      |       | Follow command line symlinks.           |
| <i>chmod</i> |         |         | –h      |       | Follow symlinks.                        |
| <i>chown</i> |         |         | –H      | –H    | Follow command line symlinks.           |
| <i>chown</i> |         |         | –h      | –L    | Follow symlinks.                        |
| <i>chown</i> | –h      |         |         | –h    | Affect the symlink.                     |
| <i>cp</i>    |         |         | –H      | –H    | Follow command line symlinks.           |
| <i>cp</i>    |         |         | –h      | –L    | Follow symlinks.                        |
| <i>cpio</i>  | –L      |         | –L      |       | Follow symlinks.                        |
| <i>du</i>    |         |         | –H      | –H    | Follow command line symlinks.           |
| <i>du</i>    |         |         | –h      | –L    | Follow symlinks.                        |
| <i>file</i>  | –h      |         |         | –h    | Affect the symlink.                     |
| <i>find</i>  |         |         | –H      | –H    | Follow command line symlinks.           |
| <i>find</i>  |         |         | –h      | –L    | Follow symlinks.                        |
| <i>find</i>  | –follow |         | –follow |       | Follow symlinks.                        |
| <i>ln</i>    | –s      | –s      | –s      | –s    | Create a symbolic link.                 |
| <i>ls</i>    | –L      | –L      | –L      | –L    | Follow symlinks.                        |
| <i>ls</i>    |         |         |         | –H    | Follow command line symlinks.           |
| <i>mv</i>    |         |         |         |       | Operates on the symlink.                |
| <i>pax</i>   |         |         | –H      | –H    | Follow command line symlinks.           |
| <i>pax</i>   |         |         | –h      | –L    | Follow symlinks.                        |
| <i>pwd</i>   |         |         |         | –L    | Printed path may contain symlinks.      |
| <i>pwd</i>   |         |         |         | –P    | Printed path will not contain symlinks. |
| <i>rm</i>    |         |         |         |       | Operates on the symlink.                |
| <i>tar</i>   |         |         | –H      |       | Follow command line symlinks.           |
| <i>tar</i>   |         | –h      | –h      |       | Follow symlinks.                        |
| <i>test</i>  | –h      |         | –h      | –h    | Affect the symlink.                     |

116975

**Synchronously-Generated Signal**

116976

116977

Those signals that may be generated synchronously include SIGABRT, SIGBUS, SIGILL, SIGFPE, SIGPIPE, and SIGSEGV.

116978

116979

Any signal sent via the *raise()* function or a *kill()* function targeting the current process is also considered synchronous.

116980

**System Call\***

116981

116982

The distinction between a “system call” and a “library routine” is an implementation detail that may differ between implementations and has thus been excluded from POSIX.1.

116983

See “Interface, Not Implementation” in the Preface.

**116984 System Console**

116985 IEEE Std 1003.1-2001/Cor 2-2004, item XBD/TC2/D6/7 is applied, changing from “An  
116986 implementation-defined device” to “A device”.

**116987 System Databases**

116988 IEEE Std 1003.1-2001/Cor 2-2004, item XBD/TC2/D6/9 is applied, rewording the definition to  
116989 reference the existing definitions for “group database” and “user database”.

**116990 System Process**

116991 IEEE Std 1003.1-2001/Cor 2-2004, item XBD/TC2/D6/8 is applied, rewording the definition to  
116992 remove the requirement for an implementation to define the object.

**116993 System Reboot**

116994 A “system reboot” is an event initiated by an unspecified circumstance that causes all processes  
116995 (other than special system processes) to be terminated in an implementation-defined manner,  
116996 after which any changes to the state and contents of files created or written to by a Conforming  
116997 POSIX.1 Application prior to the event are implementation-defined.

116998 IEEE Std 1003.1-2001/Cor 2-2004, item XBD/TC2/D6/10 is applied, changing “An  
116999 implementation-defined sequence of events” to “An unspecified sequence of events”.

**117000 Synchronized I/O Data (and File) Integrity Completion**

117001 These terms specify that for synchronized read operations, pending writes must be successfully  
117002 completed before the read operation can complete. This is motivated by two circumstances.  
117003 Firstly, when synchronizing processes can access the same file, but not share common buffers  
117004 (such as for a remote file system), this requirement permits the reading process to guarantee that  
117005 it can read data written remotely. Secondly, having data written synchronously is insufficient to  
117006 guarantee the order with respect to a subsequent write by a reading process, and thus this extra  
117007 read semantic is necessary.

**117008 Text File**

117009 The term “text file” does not prevent the inclusion of control or other non-printable characters  
117010 (other than NUL). Therefore, standard utilities that list text files as inputs or outputs are either  
117011 able to process the special characters or they explicitly describe their limitations within their  
117012 individual descriptions. The definition of “text file” has caused controversy. The only difference  
117013 between text and binary files is that text files have lines of less than {LINE\_MAX} bytes, with no  
117014 NUL characters, each terminated by a <newline>. The definition allows a file with a single  
117015 <newline>, or a totally empty file, to be called a text file. If a file ends with an incomplete line it  
117016 is not strictly a text file by this definition. The <newline> referred to in POSIX.1-2008 is not some  
117017 generic line separator, but a single character; files created on systems where they use multiple  
117018 characters for ends of lines are not portable to all conforming systems without some translation  
117019 process unspecified by POSIX.1-2008.

**Thread**

POSIX.1-2008 defines a thread to be a flow of control within a process. Each thread has a minimal amount of private state; most of the state associated with a process is shared among all of the threads in the process. While most multi-thread extensions to POSIX have taken this approach, others have made different decisions.

**Note:** The choice to put threads within a process does not constrain implementations to implement threads in that manner. However, all functions have to behave as though threads share the indicated state information with the process from which they were created.

Threads need to share resources in order to cooperate. Memory has to be widely shared between threads in order for the threads to cooperate at a fine level of granularity. Threads keep data structures and the locks protecting those data structures in shared memory. For a data structure to be usefully shared between threads, such structures should not refer to any data that can only be interpreted meaningfully by a single thread. Thus, any system resources that might be referred to in data structures need to be shared between all threads. File descriptors, pathnames, and pointers to stack variables are all things that programmers want to share between their threads. Thus, the file descriptor table, the root directory, the current working directory, and the address space have to be shared.

Library implementations are possible as long as the effective behavior is as if system services invoked by one thread do not suspend other threads. This may be difficult for some library implementations on systems that do not provide asynchronous facilities.

See [Section B.2.9](#) (on page 3592) for additional rationale.

**Thread ID**

See [Section B.2.9.2](#) (on page 3609) for additional rationale.

**Thread-Safe Function**

All functions required by POSIX.1-2008 need to be thread-safe; see [Section A.4.17](#) (on page 3481) and [Section B.2.9.1](#) (on page 3606) for additional rationale.

**User Database**

There are no references in POSIX.1-2008 to a “passwd file” or a “group file”, and there is no requirement that the *group* or *passwd* databases be kept in files containing editable text. Many large timesharing systems use *passwd* databases that are hashed for speed. Certain security classifications prohibit certain information in the *passwd* database from being publicly readable.

The term “encoded” is used instead of “encrypted” in order to avoid the implementation connotations (such as reversibility or use of a particular algorithm) of the latter term.

The *getgrent()*, *setgrent()*, *endgrent()*, *getpwent()*, *setpwent()*, and *endpwent()* functions are not included as part of the base standard because they provide a linear database search capability that is not generally useful (the *getpwuid()*, *getpwnam()*, *getgrgid()*, and *getgrnam()* functions are provided for keyed lookup) and because in certain distributed systems, especially those with different authentication domains, it may not be possible or desirable to provide an application with the ability to browse the system databases indiscriminately. They are provided on XSI-conformant systems due to their historical usage by many existing applications.

A change from historical implementations is that the structures used by these functions have fields of the types **gid\_t** and **uid\_t**, which are required to be defined in the **<sys/types.h>** header. POSIX.1-2008 requires implementations to ensure that these types are defined by inclusion of **<grp.h>** and **<pwd.h>**, respectively, without imposing any name space pollution or errors from redefinition of types.

117065 POSIX.1-2008 is silent about the content of the strings containing user or group names. These  
 117066 could be digit strings. POSIX.1-2008 is also silent as to whether such digit strings bear any  
 117067 relationship to the corresponding (numeric) user or group ID.

117068 *Database Access*

117069 The thread-safe versions of the user and group database access functions return values in user-  
 117070 supplied buffers instead of possibly using static data areas that may be overwritten by each call.

117071 IEEE Std 1003.1-2001/Cor 2-2004, item XBD/TC2/D6/11 is applied, removing the words “of  
 117072 implementation-defined format”.

117073 **Virtual Processor\***

117074 The term “virtual processor” was chosen as a neutral term describing all kernel-level  
 117075 schedulable entities, such as processes, Mach tasks, or lightweight processes. Implementing  
 117076 threads using multiple processes as virtual processors, or implementing multiplexed threads  
 117077 above a virtual processor layer, should be possible, provided some mechanism has also been  
 117078 implemented for sharing state between processes or virtual processors. Many systems may also  
 117079 wish to provide implementations of threads on systems providing “shared processes” or  
 117080 “variable-weight processes”. It was felt that exposing such implementation details would  
 117081 severely limit the type of systems upon which the threads interface could be supported and  
 117082 prevent certain types of valid implementations. It was also determined that a virtual processor  
 117083 interface was out of the scope of the Rationale (Informative) volume of POSIX.1-2008.

117084 **XSI**

117085 This is included to allow POSIX.1-2008 to be adopted as an IEEE standard and an Open Group  
 117086 Technical Standard, serving both the POSIX and the Single UNIX Specification in a core set of  
 117087 volumes.

117088 The term “XSI” has been used for 10 years in connection with the XPG series and the first and  
 117089 second versions of the base volumes of the Single UNIX Specification. The XSI margin code was  
 117090 introduced to denote the extended or more restrictive semantics beyond POSIX that are  
 117091 applicable to UNIX systems.

## 117092 **A.4 General Concepts**

117093 The general concepts are similar in nature to the definitions section, with the exception that a  
 117094 term defined in general concepts can contain normative requirements.

### 117095 **A.4.1 Concurrent Execution**

117096 There is no additional rationale provided for this section.

**117097 A.4.2 Directory Protection**

117098 There is no additional rationale provided for this section.

**117099 A.4.3 Extended Security Controls**

117100 Allowing an implementation to define extended security controls enables the use of  
117101 POSIX.1-2008 in environments that require different or more rigorous security than that  
117102 provided in POSIX.1. Extensions are allowed in two areas: privilege and file access permissions.  
117103 The semantics of these areas have been defined to permit extensions with reasonable, but not  
117104 exact, compatibility with all existing practices. For example, the elimination of the superuser  
117105 definition precludes identifying a process as privileged or not by virtue of its effective user ID.

**117106 A.4.4 File Access Permissions**

117107 A process should not try to anticipate the result of an attempt to access data by *a priori* use of  
117108 these rules. Rather, it should make the attempt to access data and examine the return value (and  
117109 possibly *errno* as well), or use *access()*. An implementation may include other security  
117110 mechanisms in addition to those specified in POSIX.1, and an access attempt may fail because of  
117111 those additional mechanisms, even though it would succeed according to the rules given in this  
117112 section. (For example, the user's security level might be lower than that of the object of the  
117113 access attempt.) The supplementary group IDs provide another reason for a process to not  
117114 attempt to anticipate the result of an access attempt.

117115 Since the current standard does not specify a method for opening a directory for searching, it is  
117116 unspecified whether search permission on the *fd* argument to *openat()* and related functions is  
117117 based on whether the directory was opened with search mode or on the current permissions  
117118 allowed by the directory at the time a search is performed. When there is existing practice that  
117119 supports opening directories for searching, it is expected that these functions will be modified to  
117120 specify that the search permissions will be granted based on the file access modes of the  
117121 directory's file descriptor identified by *fd*, and not on the mode of the directory at the time the  
117122 directory is searched.

**117123 A.4.5 File Hierarchy**

117124 Though the file hierarchy is commonly regarded to be a tree, POSIX.1 does not define it as such  
117125 for three reasons:

- 117126 1. Links may join branches.
- 117127 2. In some network implementations, there may be no single absolute root directory; see  
117128 *pathname resolution*.
- 117129 3. With symbolic links, the file system need not be a tree or even a directed acyclic graph.

117130 **A.4.6 Filenames**

117131 Historically, certain filenames and pathnames have been reserved. This list includes **core**,  
 117132 **/etc/passwd**, and so on. Conforming applications should avoid these.

117133 Most historical implementations prohibit case folding in filenames; that is, treating uppercase  
 117134 and lowercase alphabetic characters as identical. However, some consider case folding desirable:

- 117135 • For user convenience
- 117136 • For ease-of-implementation of the POSIX.1 interface as a hosted system on some popular  
 117137 operating systems

117138 Variants, such as maintaining case distinctions in filenames, but ignoring them in comparisons,  
 117139 have been suggested. Methods of allowing escaped characters of the case opposite the default  
 117140 have been proposed.

117141 Many reasons have been expressed for not allowing case folding, including:

- 117142 • No solid evidence has been produced as to whether case-sensitivity or case-insensitivity is  
 117143 more convenient for users.
- 117144 • Making case-insensitivity a POSIX.1 implementation option would be worse than either  
 117145 having it or not having it, because:
  - 117146 — More confusion would be caused among users.
  - 117147 — Application developers would have to account for both cases in their code.
  - 117148 — POSIX.1 implementors would still have other problems with native file systems, such  
 117149 as short or otherwise constrained filenames or pathnames, and the lack of  
 117150 hierarchical directory structure.
- 117151 • Case folding is not easily defined in many European languages, both because many of  
 117152 them use characters outside the US ASCII alphabetic set, and because:
  - 117153 — In Spanish, the digraph "ll" is considered to be a single letter, the capitalized form  
 117154 of which may be either "Ll" or "LL", depending on context.
  - 117155 — In French, the capitalized form of a letter with an accent may or may not retain the  
 117156 accent, depending on the country in which it is written.
  - 117157 — In German, the sharp ess may be represented as a single character resembling a  
 117158 Greek beta ( $\beta$ ) in lowercase, but as the digraph "SS" in uppercase.
  - 117159 — In Greek, there are several lowercase forms of some letters; the one to use depends on  
 117160 its position in the word. Arabic has similar rules.
- 117161 • Many East Asian languages, including Japanese, Chinese, and Korean, do not distinguish  
 117162 case and are sometimes encoded in character sets that use more than one byte per  
 117163 character.
- 117164 • Multiple character codes may be used on the same machine simultaneously. There are  
 117165 several ISO character sets for European alphabets. In Japan, several Japanese character  
 117166 codes are commonly used together, sometimes even in filenames; this is evidently also the  
 117167 case in China. To handle case insensitivity, the kernel would have to at least be able to  
 117168 distinguish for which character sets the concept made sense.
- 117169 • The file system implementation historically deals only with bytes, not with characters.  
 117170 Limitations on valid encodings ensure that the byte sequences for the <slash> character,  
 117171 <period> character, and <NUL> character will not be confused with any other character in  
 117172 any locale. However, there exist common single-shift encodings where other single-byte



- 117173 characters from the portable filename character set can also occur as a subset of a multi- |  
 117174 byte character, making case folding of portable filename bytes dependent on the context of |  
 117175 whether a shift-state is active.
- 117176 • The purpose of POSIX.1 is to standardize the common, existing definition, not to change it. |  
 117177 Mandating case-insensitivity would make all historical implementations non-standard.
  - 117178 • Not only the interface, but also application programs would need to change, counter to the |  
 117179 purpose of having minimal changes to existing application code.
  - 117180 • At least one of the original developers of the UNIX system has expressed objection in the |  
 117181 strongest terms to either requiring case-insensitivity or making it an option, mostly on the |  
 117182 basis that POSIX.1 should not hinder portability of application programs across related |  
 117183 implementations in order to allow compatibility with unrelated operating systems.
- 117184 Two proposals were entertained regarding case folding in filenames:
- 117185 1. Remove all wording that previously permitted case folding.
- 117186 Rationale Case folding is inconsistent with the portable filename character set and |  
 117187 filename definitions (all bytes except <slash> and null). No known |  
 117188 implementations allowing all bytes except <slash> and null also do case |  
 117189 folding.
- 117190 2. Change “though this practice is not recommended:” to “although this practice is strongly |  
 117191 discouraged.”
- 117192 Rationale If case folding must be included in POSIX.1, the wording should be stronger |  
 117193 to discourage the practice.
- 117194 The consensus selected the first proposal. Otherwise, a conforming application would have to |  
 117195 assume that case folding would occur when it was not wanted, but that it would not occur when |  
 117196 it was wanted.

#### 117197 **A.4.7 Filename Portability**

- 117198 Filenames should be constructed from the portable filename character set because the use of |  
 117199 other characters can be confusing or ambiguous in certain contexts. (For example, the use of a |  
 117200 <colon> ( ' : ' ) in a pathname could cause ambiguity if that pathname were included in a *PATH* |  
 117201 definition.)
- 117202 The constraint on use of the <hyphen> character as the first character of a portable filename is a |  
 117203 constraint on application behavior and not on implementations, since applications might not |  
 117204 work as expected when such a filename is passed as a command line argument.

#### 117205 **A.4.8 File Times Update**

- 117206 This section reflects the actions of historical implementations. The times are not updated |  
 117207 immediately, but are only marked for update by the functions. An implementation may update |  
 117208 these times immediately.
- 117209 The accuracy of the time update values is intentionally left unspecified so that systems can |  
 117210 control the bandwidth of a possible covert channel.
- 117211 The wording was carefully chosen to make it clear that there is no requirement that the |  
 117212 conformance document contain information that might incidentally affect file timestamps. Any |  
 117213 function that performs pathname resolution might update several last data access timestamps.

117214 Functions such as *getpwnam()* and *getgrnam()* might update the last data access timestamp of  
 117215 some specific file or files. It is intended that these are not required to be documented in the  
 117216 conformance document, but they should appear in the system documentation.

#### 117217 **A.4.9 Host and Network Byte Order**

117218 There is no additional rationale provided for this section.

#### 117219 **A.4.10 Measurement of Execution Time**

117220 The methods used to measure the execution time of processes and threads, and the precision of  
 117221 these measurements, may vary considerably depending on the software architecture of the  
 117222 implementation, and on the underlying hardware. Implementations can also make tradeoffs  
 117223 between the scheduling overhead and the precision of the execution time measurements.  
 117224 POSIX.1-2008 does not impose any requirement on the accuracy of the execution time; it instead  
 117225 specifies that the measurement mechanism and its precision are implementation-defined.

#### 117226 **A.4.11 Memory Synchronization**

117227 In older multi-processors, access to memory by the processors was strictly multiplexed. This  
 117228 meant that a processor executing program code interrogates or modifies memory in the order  
 117229 specified by the code and that all the memory operation of all the processors in the system  
 117230 appear to happen in some global order, though the operation histories of different processors are  
 117231 interleaved arbitrarily. The memory operations of such machines are said to be sequentially  
 117232 consistent. In this environment, threads can synchronize using ordinary memory operations. For  
 117233 example, a producer thread and a consumer thread can synchronize access to a circular data  
 117234 buffer as follows:

```
117235 int rdptr = 0;
117236 int wrptr = 0;
117237 data_t buf[BUFSIZE];

117238 Thread 1:
117239 while (work_to_do) {
117240 int next;

117241 buf[wrptr] = produce();
117242 next = (wrptr + 1) % BUFSIZE;
117243 while (rdptr == next)
117244 ;
117245 wrptr = next;
117246 }

117247 Thread 2:
117248 while (work_to_do) {
117249 while (rdptr == wrptr)
117250 ;
117251 consume(buf[rdptr]);
117252 rdptr = (rdptr + 1) % BUFSIZE;
117253 }
```

117254 In modern multi-processors, these conditions are relaxed to achieve greater performance. If one  
 117255 processor stores values in location A and then location B, then other processors loading data

from location B and then location A may see the new value of B but the old value of A. The memory operations of such machines are said to be weakly ordered. On these machines, the circular buffer technique shown in the example will fail because the consumer may see the new value of *wrptr* but the old value of the data in the buffer. In such machines, synchronization can only be achieved through the use of special instructions that enforce an order on memory operations. Most high-level language compilers only generate ordinary memory operations to take advantage of the increased performance. They usually cannot determine when memory operation order is important and generate the special ordering instructions. Instead, they rely on the programmer to use synchronization primitives correctly to ensure that modifications to a location in memory are ordered with respect to modifications and/or access to the same location in other threads. Access to read-only data need not be synchronized. The resulting program is said to be data race-free.

Synchronization is still important even when accessing a single primitive variable (for example, an integer). On machines where the integer may not be aligned to the bus data width or be larger than the data width, a single memory load may require multiple memory cycles. This means that it may be possible for some parts of the integer to have an old value while other parts have a newer value. On some processor architectures this cannot happen, but portable programs cannot rely on this.

In summary, a portable multi-threaded program, or a multi-process program that shares writable memory between processes, has to use the synchronization primitives to synchronize data access. It cannot rely on modifications to memory being observed by other threads in the order written in the application or even on modification of a single variable being seen atomically.

Conforming applications may only use the functions listed to synchronize threads of control with respect to memory access. There are many other candidates for functions that might also be used. Examples are: signal sending and reception, or pipe writing and reading. In general, any function that allows one thread of control to wait for an action caused by another thread of control is a candidate. POSIX.1-2008 does not require these additional functions to synchronize memory access since this would imply the following:

- All these functions would have to be recognized by advanced compilation systems so that memory operations and calls to these functions are not reordered by optimization.
- All these functions would potentially have to have memory synchronization instructions added, depending on the particular machine.
- The additional functions complicate the model of how memory is synchronized and make automatic data race detection techniques impractical.

Formal definitions of the memory model were rejected as unreadable by the vast majority of programmers. In addition, most of the formal work in the literature has concentrated on the memory as provided by the hardware as opposed to the application programmer through the compiler and runtime system. It was believed that a simple statement intuitive to most programmers would be most effective. POSIX.1-2008 defines functions that can be used to synchronize access to memory, but it leaves open exactly how one relates those functions to the semantics of each function as specified elsewhere in POSIX.1-2008. POSIX.1-2008 also does not make a formal specification of the partial ordering in time that the functions can impose, as that is implied in the description of the semantics of each function. It simply states that the programmer has to ensure that modifications do not occur “simultaneously” with other access to a memory location.

IEEE Std 1003.1-2001/Cor 1-2002, item XBD/TC1/D6/4 is applied, adding a new paragraph beneath the table of functions: “The *pthread\_once()* function shall synchronize memory for the first call in each thread for a given *pthread\_once\_t* object.”.

117305 **A.4.12 Pathname Resolution**

117306 It is necessary to differentiate between the definition of pathname and the concept of pathname  
 117307 resolution with respect to the handling of trailing <slash> characters. By specifying the behavior  
 117308 here, it is not possible to provide an implementation that is conforming but extends all interfaces  
 117309 that handle pathnames to also handle strings that are not legal pathnames (because they have  
 117310 trailing <slash> characters).

117311 Pathnames that end with one or more trailing <slash> characters must refer to directory paths.  
 117312 Earlier versions of this standard were not specific about the distinction between trailing <slash>  
 117313 characters on files and directories, and both were permitted.

117314 Two types of implementation have been prevalent; those that ignored trailing <slash> characters  
 117315 on all pathnames regardless, and those that permitted them only on existing directories.

117316 An earlier version of this standard required that a pathname with a trailing <slash> character be  
 117317 treated as if it had a trailing " / . " everywhere. This specification was ambiguous. In situations  
 117318 where the intent was that the application wanted to require the implementation to accept the  
 117319 pathname only if it named a directory (existing or to be created as a result of the call performing  
 117320 pathname resolution), literally adding a ' . ' after the trailing <slash> could be interpreted to  
 117321 require use of that pathname to fail. Some of the uses that created ambiguous requirements  
 117322 included `mkdir("newdir /")` and `rmdir("existing-dir /")`. POSIX.1-2008 requires that a pathname  
 117323 with a trailing <slash> be rejected unless it refers to a file that is a directory or to a file that is to  
 117324 be created as a directory. The `rename()` function and the `mv` utility further specify that a trailing  
 117325 <slash> cannot be used on a pathname naming a file that does not exist when used as the last  
 117326 argument to `rename()` or `renameat()`, or as the last operand to `mv`.

117327 Note that this change does not break any conforming applications; since there were two different  
 117328 types of implementation, no application could have portably depended on either behavior. This  
 117329 change does however require some implementations to be altered to remain compliant.  
 117330 Substantial discussion over a three-year period has shown that the benefits to application  
 117331 developers outweighs the disadvantages for some vendors.

117332 On a historical note, some early applications automatically appended a ' / ' to every path.  
 117333 Rather than fix the applications, the system implementation was modified to accept this  
 117334 behavior by ignoring any trailing <slash>.

117335 Each directory has exactly one parent directory which is represented by the name **dot-dot** in the  
 117336 first directory. No other directory, regardless of linkages established by symbolic links, is  
 117337 considered the parent directory by POSIX.1-2008.

117338 There are two general categories of interfaces involving pathname resolution: those that follow  
 117339 the symbolic link, and those that do not. There are several exceptions to this rule; for example,  
 117340 `open(path, O_CREAT | O_EXCL)` will fail when `path` names a symbolic link. However, in all other  
 117341 situations, the `open()` function will follow the link.

117342 What the filename **dot-dot** refers to relative to the root directory is implementation-defined. In  
 117343 Version 7 it refers to the root directory itself; this is the behavior mentioned in POSIX.1-2008. In  
 117344 some networked systems the construction `././hostname/` is used to refer to the root directory of  
 117345 another host, and POSIX.1 permits this behavior.

117346 Other networked systems use the construct `//hostname` for the same purpose; that is, a double  
 117347 initial <slash> is used. There is a potential problem with existing applications that create full  
 117348 pathnames by taking a trunk and a relative pathname and making them into a single string  
 117349 separated by ' / ', because they can accidentally create networked pathnames when the trunk is  
 117350 ' / '. This practice is not prohibited because such applications can be made to conform by  
 117351 simply changing to use " / / " as a separator instead of ' / ':

- If the trunk is `'/'`, the full pathname will begin with `"/"` (the initial `'/'` and the separator `"/"`). This is the same as `'/'`, which is what is desired. (This is the general case of making a relative pathname into an absolute one by prefixing with `"/"` instead of `'/'`.)
- If the trunk is `"/A"`, the result is `"/A/. . ."`; since non-leading sequences of two or more `<slash>` characters are treated as a single `<slash>`, this is equivalent to the desired `"/A/. . ."`.
- If the trunk is `"//A"`, the implementation-defined semantics will apply. (The multiple `<slash>` rule would apply.)

Application developers should avoid generating pathnames that start with `"/"`. Implementations are strongly encouraged to avoid using this special interpretation since a number of applications currently do not follow this practice and may inadvertently generate `"/. . ."`.

The term “root directory” is only defined in POSIX.1 relative to the process. In some implementations, there may be no absolute root directory. The initialization of the root directory of a process is implementation-defined.

When the standard says: “Pathname resolution for a given pathname shall yield the same results when used by any interface in POSIX.1-2008 as long as there are no changes to any files evaluated during pathname resolution for the given pathname between resolutions”, this applies to absolute pathnames or to relative pathnames from the same current working directory. Using the same relative pathname from two different working directories may yield different results.

Earlier versions of this standard were unclear as to whether a pathname was required to be a character string or just a string. This standard is now clear that filenames are just strings, and that pathname processing is locale-independent.

#### A.4.13 Process ID Reuse

There is no additional rationale provided for this section.

#### A.4.14 Scheduling Policy

There is no additional rationale provided for this section.

#### A.4.15 Seconds Since the Epoch

Coordinated Universal Time (UTC) includes leap seconds. However, in POSIX time (seconds since the Epoch), leap seconds are ignored (not applied) to provide an easy and compatible method of computing time differences. Broken-down POSIX time is therefore not necessarily UTC, despite its appearance.

As of December 2007, 23 leap seconds had been added to UTC since the Epoch, 1 January, 1970. Historically, one leap second is added every 15 months on average, so this offset can be expected to grow with time.

Most systems’ notion of “time” is that of a continuously increasing value, so this value should increase even during leap seconds. However, not only do most systems not keep track of leap seconds, but most systems are probably not synchronized to any standard time reference. Therefore, it is inappropriate to require that a time represented as seconds since the Epoch

precisely represent the number of seconds between the referenced time and the Epoch.

It is sufficient to require that applications be allowed to treat this time as if it represented the number of seconds between the referenced time and the Epoch. It is the responsibility of the vendor of the system, and the administrator of the system, to ensure that this value represents the number of seconds between the referenced time and the Epoch as closely as necessary for the application being run on that system.

It is important that the interpretation of time names and seconds since the Epoch values be consistent across conforming systems; that is, it is important that all conforming systems interpret “536 457 599 seconds since the Epoch” as 59 seconds, 59 minutes, 23 hours 31 December 1986, regardless of the accuracy of the system’s idea of the current time. The expression is given to ensure a consistent interpretation, not to attempt to specify the calendar. The relationship between *tm\_yday* and the day of week, day of month, and month is in accordance with the Gregorian calendar, and so is not specified in POSIX.1.

Consistent interpretation of seconds since the Epoch can be critical to certain types of distributed applications that rely on such timestamps to synchronize events. The accrual of leap seconds in a time standard is not predictable. The number of leap seconds since the Epoch will likely increase. POSIX.1 is more concerned about the synchronization of time between applications of astronomically short duration.

Note that *tm\_yday* is zero-based, not one-based, so the day number in the example above is 364. Note also that the division is an integer division (discarding remainder) as in the C language.

Note also that the meaning of *gmtime()*, *localtime()*, and *mktime()* is specified in terms of this expression. However, the ISO C standard computes *tm\_yday* from *tm\_mday*, *tm\_mon*, and *tm\_year* in *mktime()*. Because it is stated as a (bidirectional) relationship, not a function, and because the conversion between month-day-year and day-of-year dates is presumed well known and is also a relationship, this is not a problem.

Implementations that implement **time\_t** as a signed 32-bit integer will overflow in 2038. This standard requires that **time\_t** be an integer type with implementation-defined size, but does not mandate a particular size. The requirement that **time\_t** be integral is an additional constraint beyond the ISO C standard, which allows a real-floating **time\_t**. Implementation practice has shown that much existing code is unprepared to deal with a floating-point **time\_t**, and that use of **struct timespec** is a more uniform way to provide sub-second time manipulation within applications.

See also [Epoch](#) (on page 3454).

The topic of whether seconds since the Epoch should account for leap seconds has been debated on a number of occasions, and each time consensus was reached (with acknowledged dissent each time) that the majority of users are best served by treating all days identically. (That is, the majority of applications were judged to assume a single length—as measured in seconds since the Epoch—for all days. Thus, leap seconds are not applied to seconds since the Epoch.) Those applications which do care about leap seconds can determine how to handle them in whatever way those applications feel is best. This was particularly emphasized because there was disagreement about what the best way of handling leap seconds might be. It is a practical impossibility to mandate that a conforming implementation must have a fixed relationship to any particular official clock (consider isolated systems, or systems performing “reruns” by setting the clock to some arbitrary time).

Note that as a practical consequence of this, the length of a second as measured by some external standard is not specified. This unspecified second is nominally equal to an International System (SI) second in duration. Applications must be matched to a system that provides the particular handling of external time in the way required by the application.

117441 IEEE Std 1003.1-2001/Cor 2-2004, item XBD/TC2/D6/12 is applied, making an editorial  
117442 correction to the paragraph commencing “How any changes to the value of seconds ...”.

117443 **A.4.16 Semaphore**

117444 There is no additional rationale provided for this section.

117445 **A.4.17 Thread-Safety**

117446 Where the interface of a function required by POSIX.1-2008 precludes thread-safety, an alternate  
117447 thread-safe form is provided. The names of these thread-safe forms are the same as the non-  
117448 thread-safe forms with the addition of the suffix “\_r”. The suffix “\_r” is historical, where the  
117449 ‘r’ stood for “reentrant”.

117450 In some cases, thread-safety is provided by restricting the arguments to an existing function.

117451 See also [Section B.2.9.1](#) (on page 3606).

117452 **A.4.18 Tracing**

117453 Refer to [Section B.2.11](#) (on page 3622).

117454 **A.4.19 Treatment of Error Conditions for Mathematical Functions**

117455 There is no additional rationale provided for this section.

117456 **A.4.20 Treatment of NaN Arguments for Mathematical Functions**

117457 There is no additional rationale provided for this section.

117458 **A.4.21 Utility**

117459 There is no additional rationale provided for this section.

117460 **A.4.22 Variable Assignment**

117461 There is no additional rationale provided for this section.

## A.5 File Format Notation

The notation for spaces allows some flexibility for application output. Note that an empty character position in *format* represents one or more <blank> characters on the output (not *white space*, which can include <newline> characters). Therefore, another utility that reads that output as its input must be prepared to parse the data using *scanf()*, *awk*, and so on. The 'Δ' character is used when exactly one <space> is output.

The treatment of integers and spaces is different from the *printf()* function in that they can be surrounded with <blank> characters. This was done so that, given a format such as:

```
"%d\n", <foo>
```

the implementation could use a *printf()* call such as:

```
printf("%6d\n", foo);
```

and still conform. This notation is thus somewhat like *scanf()* in addition to *printf()*.

The *printf()* function was chosen as a model because most of the standard developers were familiar with it. One difference from the C function *printf()* is that the *l* and *h* conversion specifier characters are not used. As expressed by the Shell and Utilities volume of POSIX.1-2008, there is no differentiation between decimal values for type **int**, type **long**, or type **short**. The conversion specifications *%d* or *%i* should be interpreted as an arbitrary length sequence of digits. Also, no distinction is made between single precision and double precision numbers (**float** or **double** in C). These are simply referred to as floating-point numbers.

Many of the output descriptions in the Shell and Utilities volume of POSIX.1-2008 use the term "line", such as:

```
"%s", <input line>
```

Since the definition of *line* includes the trailing <newline> already, there is no need to include a '\n' in the format; a double <newline> would otherwise result.

## A.6 Character Set

### A.6.1 Portable Character Set

The portable character set is listed in full so there is no dependency on the ISO/IEC 646:1991 standard (or historically ASCII) encoded character set, although the set is identical to the characters defined in the International Reference version of the ISO/IEC 646:1991 standard.

POSIX.1-2008 poses no requirement that multiple character sets or codesets be supported, leaving this as a marketing differentiation for implementors. Although multiple charmap files are supported, it is the responsibility of the implementation to provide the file(s); if only one is provided, only that one will be accessible using the *localedef -f* option.

The statement about invariance in codesets for the portable character set is worded to avoid precluding implementations where multiple incompatible codesets are available (for instance, ASCII and EBCDIC). The standard utilities cannot be expected to produce predictable results if they access portable characters that vary on the same implementation.

Not all character sets need include the portable character set, but each locale must include it. For



117500 example, a Japanese-based locale might be supported by a mixture of character sets: JIS X 0201  
 117501 Roman (a Japanese version of the ISO/IEC 646:1991 standard), JIS X 0208, and JIS X 0201  
 117502 Katakana. Not all of these character sets include the portable characters, but at least one does  
 117503 (JIS X 0201 Roman).

## 117504 **A.6.2 Character Encoding**

117505 Encoding mechanisms based on single shifts, such as the EUC encoding used in some Asian and  
 117506 other countries, can be supported via the current charmap mechanism. With single-shift  
 117507 encoding, each character is preceded by a shift code (SS2 or SS3). A complete EUC code,  
 117508 consisting of the portable character set (G0) and up to three additional character sets (G1, G2,  
 117509 G3), can be described using the current charmap mechanism; the encoding for each character in  
 117510 additional character sets G2 and G3 must then include their single-shift code. Other mechanisms  
 117511 to support locales based on encoding mechanisms such as locking shift are not addressed by this  
 117512 volume of POSIX.1-2008.

117513 The encodings for <slash> and <period> are required to be the same across all locales, in part +  
 117514 because pathname resolution requires recognition of these bytes. It is a fortunate accident that all +  
 117515 common shift-based encodings did not use either <slash> or <period> as a valid second byte in +  
 117516 a multi-byte character.

## 117517 **A.6.3 C Language Wide-Character Codes**

117518 The standard does not specify how wide characters are encoded or provide a method for  
 117519 defining wide characters in a charmap. It specifies ways of translating between wide characters  
 117520 and multi-byte characters. The standard does not prevent an extension from providing a method  
 117521 to define wide characters.

117522 IEEE Std 1003.1-2001/Cor 2-2004, item XBD/TC2/D6/13 is applied, adding a statement that the  
 117523 standard has no means of defining a wide-character codeset.

## 117524 **A.6.4 Character Set Description File**

117525 IEEE PASC Interpretation 1003.2 #196 is applied, removing three lines of text dealing with  
 117526 ranges of symbolic names using position constant values which had been erroneously included  
 117527 in the final IEEE P1003.2b draft standard.

117528 IEEE Std 1003.1-2001/Cor 2-2004, item XBD/TC2/D6/14 is applied, correcting the example and  
 117529 adding a statement that the standard provides no means of defining a wide-character codeset.

117530 IEEE Std 1003.1-2001/Cor 2-2004, item XBD/TC2/D6/15 is applied, allowing the value zero for  
 117531 the width value of **WIDTH** and **WIDTH\_DEFAULT**. This is required to cover some existing  
 117532 locales.

### 117533 *A.6.4.1 State-Dependent Character Encodings*

117534 A requirement was considered that would force utilities to eliminate any redundant locking  
 117535 shifts, but this was left as a quality of implementation issue.

117536 This change satisfies the following requirement from the ISO POSIX-2:1993 standard, Annex  
 117537 H.1:

117538 The support of state-dependent (shift encoding) character sets should be addressed fully. See  
 117539 descriptions of these in XBD [Section 6.2](#) (on page 128). If such character encodings are supported,  
 117540 it is expected that this will impact XBD [Section 6.2](#) (on page 128), [Chapter 7](#) (on page 135),  
 117541 [Chapter 9](#) (on page 181), and the *comm*, *cut*, *diff*, *grep*, *head*, *join*, *paste*, and *tail* utilities.

117542 The character set description file provides:

- 117543 • The capability to describe character set attributes (such as collation order or character  
 117544 classes) independent of character set encoding, and using only the characters in the  
 117545 portable character set. This makes it possible to create generic *localedef* source files for all  
 117546 codesets that share the portable character set (such as the ISO 8859 family or IBM Extended  
 117547 ASCII).
- 117548 • Standardized symbolic names for all characters in the portable character set, making it  
 117549 possible to refer to any such character regardless of encoding.

117550 Implementations are free to choose their own symbolic names, as long as the names identified  
 117551 by the Base Definitions volume of POSIX.1-2008 are also defined; this provides support for  
 117552 already existing “character names”.

117553 The names selected for the members of the portable character set follow the  
 117554 ISO/IEC 8859-1:1998 standard and the ISO/IEC 10646-1:2000 standard. However, several  
 117555 commonly used UNIX system names occur as synonyms in the list:

- 117556 • The historical UNIX system names are used for control characters.
- 117557 • The word “slash” is given in addition to “solidus”.
- 117558 • The word “backslash” is given in addition to “reverse-solidus”.
- 117559 • The word “hyphen” is given in addition to “hyphen-minus”.
- 117560 • The word “period” is given in addition to “full-stop”.
- 117561 • For digits, the word “digit” is eliminated.
- 117562 • For letters, the words “Latin Capital Letter” and “Latin Small Letter” are eliminated.
- 117563 • The words “left brace” and “right brace” are given in addition to “left-curly-bracket” and  
 117564 “right-curly-bracket”.
- 117565 • The names of the digits are preferred over the numbers to avoid possible confusion  
 117566 between ‘0’ and ‘o’, and between ‘1’ and ‘l’ (one and the letter ell).

117567 The names for the control characters in XBD [Chapter 6](#) (on page 125) were taken from the  
 117568 ISO/IEC 4873: 1991 standard.

117569 The charmap file was introduced to resolve problems with the portability of, especially, *localedef*  
 117570 sources. POSIX.1-2008 assumes that the portable character set is constant across all locales, but  
 117571 does not prohibit implementations from supporting two incompatible codings, such as both  
 117572 ASCII and EBCDIC. Such dual-support implementations should have all charmaps and *localedef*  
 117573 sources encoded using one portable character set, in effect cross-compiling for the other  
 117574 environment. Naturally, charmaps (and *localedef* sources) are only portable without  
 117575 transformation between systems using the same encodings for the portable character set. They  
 117576 can, however, be transformed between two sets using only a subset of the actual characters (the  
 117577 portable character set). However, the particular coded character set used for an application or an  
 117578 implementation does not necessarily imply different characteristics or collation; on the contrary,  
 117579 these attributes should in many cases be identical, regardless of codeset. The charmap provides  
 117580 the capability to define a common locale definition for multiple codesets (the same *localedef*  
 117581 source can be used for codesets with different extended characters; the ability in the charmap to

- 117582 define empty names allows for characters missing in certain codesets).
- 117583 The **<escape\_char>** declaration was added at the request of the international community to ease  
 117584 the creation of portable charmap files on terminals not implementing the default  
 117585 **<backslash>-escape**. The **<comment\_char>** declaration was added at the request of the  
 117586 international community to eliminate the potential confusion between the **<number-sign>** and  
 117587 the hash sign.
- 117588 The octal number notation with no leading zero required was selected to match those of *awk* and  
 117589 *tr* and is consistent with that used by *localedef*. To avoid confusion between an octal constant and  
 117590 the back-references used in *localedef* source, the octal, hexadecimal, and decimal constants must  
 117591 contain at least two digits. As single-digit constants are relatively rare, this should not impose  
 117592 any significant hardship. Provision is made for more digits to account for systems in which the  
 117593 byte size is larger than 8 bits. For example, a Unicode (ISO/IEC 10646-1:2000 standard) system  
 117594 that has defined 16-bit bytes may require six octal, four hexadecimal, and five decimal digits.
- 117595 The decimal notation is supported because some newer international standards define character  
 117596 values in decimal, rather than in the old column/row notation.
- 117597 The charmap identifies the coded character sets supported by an implementation. At least one  
 117598 charmap must be provided, but no implementation is required to provide more than one.  
 117599 Likewise, implementations can allow users to generate new charmaps (for instance, for a new  
 117600 version of the ISO 8859 family of coded character sets), but does not have to do so. If users are  
 117601 allowed to create new charmaps, the system documentation describes the rules that apply (for  
 117602 instance, “only coded character sets that are supersets of the ISO/IEC 646:1991 standard IRV, no  
 117603 multi-byte characters”).
- 117604 This addition of the **WIDTH** specification satisfies the following requirement from the  
 117605 ISO POSIX-2:1993 standard, Annex H.1:
- 117606 (9) *The definition of column position relies on the implementation’s knowledge of the integral*  
 117607 *width of the characters. The charmap or LC\_CTYPE locale definitions should be enhanced to*  
 117608 *allow application specification of these widths.*
- 117609 The character “width” information was first considered for inclusion under *LC\_CTYPE* but was  
 117610 moved because it is more closely associated with the information in the charmap than  
 117611 information in the locale source (cultural conventions information). Concerns were raised that  
 117612 formalizing this type of information is moving the locale source definition from the codeset-  
 117613 independent entity that it was designed to be to a repository of codeset-specific information. A  
 117614 similar issue occurred with the **<code\_set\_name>**, **<mb\_cur\_max>**, and **<mb\_cur\_min>**  
 117615 information, which was resolved to reside in the charmap definition.
- 117616 The width definition was added to the IEEE P1003.2b draft standard with the intent that the  
 117617 *wcswidth()* and/or *wcwidth()* functions (currently specified in the System Interfaces volume of  
 117618 POSIX.1-2008) be the mechanism to retrieve the character width information.

## 117619 A.7 Locale

### 117620 A.7.1 General

117621 The description of locales is based on work performed in the UniForum Technical Committee,  
117622 Subcommittee on Internationalization. Wherever appropriate, keywords are taken from the  
117623 ISO C standard or the X/Open Portability Guide.

117624 The value used to specify a locale with environment variables is the name specified as the *name*  
117625 operand to the *localedef* utility when the locale was created. This provides a verifiable method to  
117626 create and invoke a locale.

117627 The “object” definitions need not be portable, as long as “source” definitions are. Strictly  
117628 speaking, source definitions are portable only between implementations using the same  
117629 character set(s). Such source definitions, if they use symbolic names only, easily can be ported  
117630 between systems using different codesets, as long as the characters in the portable character set  
117631 (see XBD [Section 6.1](#), on page 125) have common values between the codesets; this is frequently  
117632 the case in historical implementations. Of source, this requires that the symbolic names used for  
117633 characters outside the portable character set be identical between character sets. The definition  
117634 of symbolic names for characters is outside the scope of POSIX.1-2008, but is certainly within the  
117635 scope of other standards organizations.

117636 Applications can select the desired locale by invoking the *setlocale()* function (or equivalent)  
117637 with the appropriate value. If the function is invoked with an empty string, the value of the  
117638 corresponding environment variable is used. If the environment variable is not set or is set to the  
117639 empty string, the implementation sets the appropriate environment as defined in XBD [Chapter 8](#)  
117640 (on page 173).

### 117641 A.7.2 POSIX Locale

117642 The POSIX locale is equal to the C locale. To avoid being classified as a C-language function, the  
117643 name has been changed to the POSIX locale; the environment variable value can be either  
117644 "POSIX" or, for historical reasons, "C".

117645 The POSIX definitions mirror the historical UNIX system behavior.

117646 The use of symbolic names for characters in the tables does not imply that the POSIX locale must  
117647 be described using symbolic character names, but merely that it may be advantageous to do so.

### 117648 A.7.3 Locale Definition

117649 The decision to separate the file format from the *localedef* utility description was only partially  
117650 editorial. Implementations may provide other interfaces than *localedef*. Requirements on “the  
117651 utility”, mostly concerning error messages, are described in this way because they are meant to  
117652 affect the other interfaces implementations may provide as well as *localedef*.

117653 The text about POSIX2\_LOCALEDEF does not mean that internationalization is optional; only  
117654 that the functionality of the *localedef* utility is. REs, for instance, must still be able to recognize,  
117655 for example, character class expressions such as "[[:alpha:]]". A possible analogy is with  
117656 an applications development environment; while all conforming implementations must be  
117657 capable of executing applications, not all need to have the development environment installed.  
117658 The assumption is that the capability to modify the behavior of utilities (and applications) via

locale settings must be supported. If the *localedef* utility is not present, then the only choice is to select an existing (presumably implementation-documented) locale. An implementation could, for example, choose to support only the POSIX locale, which would in effect limit the amount of changes from historical implementations quite drastically. The *localedef* utility is still required, but would always terminate with an exit code indicating that no locale could be created. Supported locales must be documented using the syntax defined in this chapter. (This ensures that users can accurately determine what capabilities are provided. If the implementation decides to provide additional capabilities to the ones in this chapter, that is already provided for.)

If the option is present (that is, locales can be created), then the *localedef* utility must be capable of creating locales based on the syntax and rules defined in this chapter. This does not mean that the implementation cannot also provide alternate means for creating locales.

The octal, decimal, and hexadecimal notations are the same employed by the charmap facility (see XBD Section 6.4, on page 129). To avoid confusion between an octal constant and a back-reference, the octal, hexadecimal, and decimal constants must contain at least two digits. As single-digit constants are relatively rare, this should not impose any significant hardship. Provision is made for more digits to account for systems in which the byte size is larger than 8 bits. For example, a Unicode (see the ISO/IEC 10646-1:2000 standard) system that has defined 16-bit bytes may require six octal, four hexadecimal, and five decimal digits. As with the charmap file, multi-byte characters are described in the locale definition file using “big-endian” notation for reasons of portability. There is no requirement that the internal representation in the computer memory be in this same order.

One of the guidelines used for the development of this volume of POSIX.1-2008 is that characters outside the invariant part of the ISO/IEC 646:1991 standard should not be used in portable specifications. The <backslash> character is not in the invariant part; the <number-sign> is, but with multiple representations: as a <number-sign>, and as a hash sign. As far as general usage of these symbols, they are covered by the “grandfather clause”, but for newly defined interfaces, the WG15 POSIX working group has requested that POSIX provide alternate representations. Consequently, while the default escape character remains the <backslash> and the default comment character is the <number-sign>, implementations are required to recognize alternative representations, identified in the applicable source file via the <escape\_char> and <comment\_char> keywords.

#### A.7.3.1 LC\_CTYPE

The *LC\_CTYPE* category is primarily used to define the encoding-independent aspects of a character set, such as character classification. In addition, certain encoding-dependent characteristics are also defined for an application via the *LC\_CTYPE* category. POSIX.1-2008 does not mandate that the encoding used in the locale is the same as the one used by the application because an implementation may decide that it is advantageous to define locales in a system-wide encoding rather than having multiple, logically identical locales in different encodings, and to convert from the application encoding to the system-wide encoding on usage. Other implementations could require encoding-dependent locales.

In either case, the *LC\_CTYPE* attributes that are directly dependent on the encoding, such as <mb\_cur\_max> and the display width of characters, are not user-specifiable in a locale source and are consequently not defined as keywords.

Implementations may define additional keywords or extend the *LC\_CTYPE* mechanism to allow application-defined keywords.

The text “The ellipsis specification shall only be valid within a single encoded character set” is present because it is possible to have a locale supported by multiple character encodings, as

117707 explained in the rationale for XBD [Section 6.1](#) (on page 125). An example given there is of a  
 117708 possible Japanese-based locale supported by a mixture of the character sets JIS X 0201 Roman,  
 117709 JIS X 0208, and JIS X 0201 Katakana. Attempting to express a range of characters across these sets  
 117710 is not logical and the implementation is free to reject such attempts.

117711 As the `LC_CTYPE` character classes are based on the ISO C standard character class definition,  
 117712 the category does not support multi-character elements. For instance, the German character  
 117713 <sharp-s> is traditionally classified as a lowercase letter. There is no corresponding uppercase  
 117714 letter; in proper capitalization of German text, the <sharp-s> will be replaced by "SS"; that is, by  
 117715 two characters. This kind of conversion is outside the scope of the **toupper** and **tolower**  
 117716 keywords.

117717 Where POSIX.1-2008 specifies that only certain characters can be specified, as for the keywords  
 117718 **digit** and **xdigit**, the specified characters must be from the portable character set, as shown. As  
 117719 an example, only the Arabic digits 0 through 9 are acceptable as digits.

117720 The character classes **digit**, **xdigit**, **lower**, **upper**, and **space** have a set of automatically included  
 117721 characters. These only need to be specified if the character values (that is, encoding) differs from  
 117722 the implementation default values. It is not possible to define a locale without these  
 117723 automatically included characters unless some implementation extension is used to prevent  
 117724 their inclusion. Such a definition would not be a proper superset of the C locale, and thus, it  
 117725 might not be possible for the standard utilities to be implemented as programs conforming to  
 117726 the ISO C standard.

117727 The definition of character class **digit** requires that only ten characters—the ones defining  
 117728 digits—can be specified; alternate digits (for example, Hindi or Kanji) cannot be specified here.  
 117729 However, the encoding may vary if an implementation supports more than one encoding.

117730 The definition of character class **xdigit** requires that the characters included in character class  
 117731 **digit** are included here also and allows for different symbols for the hexadecimal digits 10  
 117732 through 15.

117733 The inclusion of the **charclass** keyword satisfies the following requirement from the  
 117734 ISO POSIX-2: 1993 standard, Annex H.1:

117735 (3) *The `LC_CTYPE` (2.5.2.1) locale definition should be enhanced to allow user-specified additional*  
 117736 *character classes, similar in concept to the ISO C standard Multibyte Support Extension (MSE)*  
 117737 *`iswctype()` function.*

117738 This keyword was previously included in The Open Group specifications and is now mandated  
 117739 in the Shell and Utilities volume of POSIX.1-2008.

117740 The symbolic constant `{CHARCLASS_NAME_MAX}` was also adopted from The Open Group  
 117741 specifications. Applications portability is enhanced by the use of symbolic constants.

#### 117742 A.7.3.2 `LC_COLLATE`

117743 The rules governing collation depend to some extent on the use. At least five different levels of  
 117744 increasingly complex collation rules can be distinguished:

- 117745 1. *Byte/machine code order*: This is the historical collation order in the UNIX system and many  
 117746 proprietary operating systems. Collation is here performed character by character,  
 117747 without any regard to context. The primary virtue is that it usually is quite fast and also  
 117748 completely deterministic; it works well when the native machine collation sequence  
 117749 matches the user expectations.

2. *Character order*: On this level, collation is also performed character by character, without regard to context. The order between characters is, however, not determined by the code values, but on the expectations by the user of the “correct” order between characters. In addition, such a (simple) collation order can specify that certain characters collate equally (for example, uppercase and lowercase letters).
3. *String ordering*: On this level, entire strings are compared based on relatively straightforward rules. Several “passes” may be required to determine the order between two strings. Characters may be ignored in some passes, but not in others; the strings may be compared in different directions; and simple string substitutions may be performed before strings are compared. This level is best described as “dictionary” ordering; it is based on the spelling, not the pronunciation, or meaning, of the words.
4. *Text search ordering*: This is a further refinement of the previous level, best described as “telephone book ordering”; some common homonyms (words spelled differently but with the same pronunciation) are collated together; numbers are collated as if they were spelled out, and so on.
5. *Semantic-level ordering*: Words and strings are collated based on their meaning; entire words (such as “the”) are eliminated; the ordering is not deterministic. This usually requires special software and is highly dependent on the intended use.

While the historical collation order formally is at level 1, for the English language it corresponds roughly to elements at level 2. The user expects to see the output from the *ls* utility sorted very much as it would be in a dictionary. While telephone book ordering would be an optimal goal for standard collation, this was ruled out as the order would be language-dependent. Furthermore, a requirement was that the order must be determined solely from the text string and the collation rules; no external information (for example, “pronunciation dictionaries”) could be required.

As a result, the goal for the collation support is at level 3. This also matches the requirements for the Canadian collation order, as well as other, known collation requirements for alphabetic scripts. It specifically rules out collation based on pronunciation rules or based on semantic analysis of the text.

The syntax for the *LC\_COLLATE* category source meets the requirements for level 3 and has been verified to produce the correct result with examples based on French, Canadian, and Danish collation order. Because it supports multi-character collating elements, it is also capable of supporting collation in codesets where a character is expressed using non-spacing characters followed by the base character (such as the ISO/IEC 6937: 2001 standard).

The directives that can be specified in an operand to the **order\_start** keyword are based on the requirements specified in several proposed standards and in customary use. The following is a rephrasing of rules defined for “lexical ordering in English and French” by the Canadian Standards Association (the text in square brackets is rephrased):

- Once special characters [punctuation] have been removed from original strings, the ordering is determined by scanning forwards (left to right) [disregarding case and diacriticals].
- In case of equivalence, special characters are once again removed from original strings and the ordering is determined by scanning backwards (starting from the rightmost character of the string and back), character by character [disregarding case but considering diacriticals].
- In case of repeated equivalence, special characters are removed again from original strings and the ordering is determined by scanning forwards, character by character [considering both case and diacriticals].



- If there is still an ordering equivalence after the first three rules have been applied, then only special characters and the position they occupy in the string are considered to determine ordering. The string that has a special character in the lowest position comes first. If two strings have a special character in the same position, the character [with the lowest collation value] comes first. In case of equality, the other special characters are considered until there is a difference or until all special characters have been exhausted.

It is estimated that this part of POSIX.1-2008 covers the requirements for all European languages, and no particular problems are anticipated with Slavic or Middle East character sets.

The Far East (particularly Japanese/Chinese) collations are often based on contextual information and pronunciation rules (the same ideogram can have different meanings and different pronunciations). Such collation, in general, falls outside the desired goal of POSIX.1-2008. There are, however, several other collation rules (stroke/radical or “most common pronunciation”) that can be supported with the mechanism described here.

The character order is defined by the order in which characters and elements are specified between the **order\_start** and **order\_end** keywords. Weights assigned to the characters and elements define the collation sequence; in the absence of weights, the character order is also the collation sequence.

The **position** keyword provides the capability to consider, in a compare, the relative position of characters not subject to **IGNORE**. As an example, consider the two strings "o-ring" and "or-ing". Assuming the <hyphen> is subject to **IGNORE** on the first pass, the two strings compare equal, and the position of the <hyphen> is immaterial. On second pass, all characters except the <hyphen> are subject to **IGNORE**, and in the normal case the two strings would again compare equal. By taking position into account, the first collates before the second.

#### 117821 A.7.3.3 LC\_MONETARY

The currency symbol does not appear in *LC\_MONETARY* because it is not defined in the C locale of the ISO C standard.

The ISO C standard limits the size of decimal points and thousands delimiters to single-byte values. In locales based on multi-byte coded character sets, this cannot be enforced; POSIX.1-2008 does not prohibit such characters, but makes the behavior unspecified (in the text “In contexts where other standards ...”).

The grouping specification is based on, but not identical to, the ISO C standard. The -1 indicates that no further grouping is performed; the equivalent of {CHAR\_MAX} in the ISO C standard.

The text “the value is not available in the locale” is taken from the ISO C standard and is used instead of the “unspecified” text in early proposals. There is no implication that omitting these keywords or assigning them values of " " or -1 produces unspecified results; such omissions or assignments eliminate the effects described for the keyword or produce zero-length strings, as appropriate.

The locale definition is an extension of the ISO C standard *localeconv()* specification. In particular, rules on how **currency\_symbol** is treated are extended to also cover **int\_curr\_symbol**, and **p\_set\_by\_space** and **n\_sep\_by\_space** have been augmented with the value 2, which places a <space> between the sign and the symbol. This has been updated to match the ISO/IEC 9899: 1999 standard requirements and is an incompatible change from UNIX 98 and the ISO POSIX-2 standard and the ISO POSIX-1: 1996 standard requirements. The following table shows the result of various combinations:



|                          |                        | <b>p_sep_by_space</b> |           |          |
|--------------------------|------------------------|-----------------------|-----------|----------|
|                          |                        | <b>2</b>              | <b>1</b>  | <b>0</b> |
| <b>p_cs_precedes = 1</b> | <b>p_sign_posn = 0</b> | (\$1.25)              | (\$ 1.25) | (\$1.25) |
|                          | <b>p_sign_posn = 1</b> | + \$1.25              | +\$ 1.25  | +\$1.25  |
|                          | <b>p_sign_posn = 2</b> | \$1.25 +              | \$ 1.25+  | \$1.25+  |
|                          | <b>p_sign_posn = 3</b> | + \$1.25              | +\$ 1.25  | +\$1.25  |
|                          | <b>p_sign_posn = 4</b> | \$ +1.25              | \$+ 1.25  | \$+1.25  |
| <b>p_cs_precedes = 0</b> | <b>p_sign_posn = 0</b> | (1.25 \$)             | (1.25 \$) | (1.25\$) |
|                          | <b>p_sign_posn = 1</b> | +1.25 \$              | +1.25 \$  | +1.25\$  |
|                          | <b>p_sign_posn = 2</b> | 1.25\$ +              | 1.25 \$+  | 1.25\$+  |
|                          | <b>p_sign_posn = 3</b> | 1.25+ \$              | 1.25 +\$  | 1.25+\$  |
|                          | <b>p_sign_posn = 4</b> | 1.25\$ +              | 1.25 \$+  | 1.25\$+  |

The following is an example of the interpretation of the **mon\_grouping** keyword. Assuming that the value to be formatted is 123 456 789 and the **mon\_thousands\_sep** is <apostrophe>, then the following table shows the result. The third column shows the equivalent string in the ISO C standard that would be used by the *localeconv()* function to accommodate this grouping.

| <b>mon_grouping</b> | <b>Formatted Value</b> | <b>ISO C String</b> |
|---------------------|------------------------|---------------------|
| 3;-1                | 123456'789             | "\3\177"            |
| 3                   | 123'456'789            | "\3"                |
| 3;2;-1              | 1234'56'789            | "\3\2\177"          |
| 3;2                 | 12'34'56'789           | "\3\2"              |
| -1                  | 123456789              | "\177"              |

In these examples, the octal value of {CHAR\_MAX} is 177.

IEEE Std 1003.1-2001/Cor 1-2002, item XBD/TC1/D6/6 adds a correction that permits the Euro currency symbol and addresses extensibility. The correction is stated using the term “should” intentionally, in order to make this a recommendation rather than a restriction on implementations. This allows for flexibility in implementations on how they handle future currency symbol additions.

IEEE Std 1003.1-2001/Cor 1-2002, tem XBD/TC1/D6/5 is applied, adding the **int\_[np]\_\*** values to the POSIX locale definition of *LC\_MONETARY*.

IEEE Std 1003.1-2001/Cor 2-2004, item XBD/TC2/D6/16 is applied, updating the descriptions of **p\_sep\_by\_space**, **n\_sep\_by\_space**, **int\_p\_sep\_by\_space**, and **int\_n\_sep\_by\_space** to match the description of these keywords in the ISO C standard and the System Interfaces volume of POSIX.1-2008, *localeconv()*.

#### A.7.3.4 *LC\_NUMERIC*

See the rationale for *LC\_MONETARY* for a description of the behavior of grouping.

#### A.7.3.5 *LC\_TIME*

Although certain of the conversion specifications in the POSIX locale (such as the name of the month) are shown with initial capital letters, this need not be the case in other locales. Programs using these conversion specifications may need to adjust the capitalization if the output is going to be used at the beginning of a sentence.

The *LC\_TIME* descriptions of **abday**, **day**, **mon**, and **abmon** imply a Gregorian style calendar (7-day weeks, 12-month years, leap years, and so on). Formatting time strings for other types of

117885 calendars is outside the scope of POSIX.1-2008.

117886 While the ISO 8601:2004 standard numbers the weekdays starting with Monday, historical  
 117887 practice is to use the Sunday as the first day. Rather than change the order and introduce  
 117888 potential confusion, the days must be specified beginning with Sunday; previous references to  
 117889 "first day" have been removed. Note also that the Shell and Utilities volume of POSIX.1-2008  
 117890 *date* utility supports numbering compliant with the ISO 8601:2004 standard.

117891 As specified under *date* in the Shell and Utilities volume of POSIX.1-2008 and *strftime()* in the  
 117892 System Interfaces volume of POSIX.1-2008, the conversion specifications corresponding to the  
 117893 optional keywords consist of a modifier followed by a traditional conversion specification (for  
 117894 instance, %Ex). If the optional keywords are not supported by the implementation or are  
 117895 unspecified for the current locale, these modified conversion specifications are treated as the  
 117896 traditional conversion specifications. For example, assume the following keywords:

117897 alt\_digits "0th"; "1st"; "2nd"; "3rd"; "4th"; "5th"; \  
 117898 "6th"; "7th"; "8th"; "9th"; "10th"

117899 d\_fmt "The %Od day of %B in %Y"

117900 On July 4th 1776, the %x conversion specifications would result in "The 4th day of July  
 117901 in 1776", while on July 14th 1789 it would result in "The 14 day of July in 1789". It  
 117902 can be noted that the above example is for illustrative purposes only; the %O modifier is  
 117903 primarily intended to provide for Kanji or Hindi digits in *date* formats.

117904 The following is an example for Japan that supports the current plus last three Emperors and  
 117905 reverts to Western style numbering for years prior to the Meiji era. The example also allows for  
 117906 the custom of using a special name for the first year of an era instead of using 1. (The examples  
 117907 substitute romaji where kanji should be used.)

117908 era\_d\_fmt "%EY%mgatsu%dnichi (%a)"

117909 era "+:2:1990/01/01:+\*:Heisei:%EC%Eynen";\  
 117910 "+:1:1989/01/08:1989/12/31:Heisei:%ECgannen";\  
 117911 "+:2:1927/01/01:1989/01/07:Shouwa:%EC%Eynen";\  
 117912 "+:1:1926/12/25:1926/12/31:Shouwa:%ECgannen";\  
 117913 "+:2:1913/01/01:1926/12/24:Taishou:%EC%Eynen";\  
 117914 "+:1:1912/07/30:1912/12/31:Taishou:%ECgannen";\  
 117915 "+:2:1869/01/01:1912/07/29:Meiji:%EC%Eynen";\  
 117916 "+:1:1868/09/08:1868/12/31:Meiji:%ECgannen";\  
 117917 "-:1868:1868/09/07:-\*::%Ey"

117918 Assuming that the current date is September 21, 1991, a request to *date* or *strftime()* would yield  
 117919 the following results:

117920 %Ec - Heisei3nen9gatsu2lnichi (Sat) 14:39:26  
 117921 %EC - Heisei  
 117922 %Ex - Heisei3nen9gatsu2lnichi (Sat)  
 117923 %Ey - 3  
 117924 %EY - Heisei3nen

117925 Example era definitions for the Republic of China:

117926 era "+:2:1913/01/01:+\*:ChungHwaMingGuo:%EC%EyNen";\  
 117927 "+:1:1912/1/1:1912/12/31:ChungHwaMingGuo:%ECYuenNen";\  
 117928 "+:1:1911/12/31:-\*:MingChien:%EC%EyNen"

117929 Example definitions for the Christian Era:

117930 era "+:1:0001/01/01:+\*:AD:%EC %Ey";\  
 117931 "-:0:0000/01/01:-\*:BC:%EC %Ey";\

|        |                                                                                                                 |   |
|--------|-----------------------------------------------------------------------------------------------------------------|---|
| 117931 | "+:1:-0001/12/31:-*:BC:%Ey %EC"                                                                                 |   |
| 117932 | A.7.3.6 LC_MESSAGES                                                                                             |   |
| 117933 | The <b>yesstr</b> and <b>nostr</b> locale keywords and the YESSTR and NOSTR <i>langinfo</i> items were formerly |   |
| 117934 | used to match user affirmative and negative responses. In POSIX.1-2008, the <b>yesexpr</b> , <b>noexpr</b> ,    |   |
| 117935 | YESEXPR, and NOEXPR extended regular expressions have replaced them. Applications                               |   |
| 117936 | should use the general locale-based messaging facilities to issue prompting messages which                      |   |
| 117937 | include sample desired responses.                                                                               |   |
| 117938 | Affirmative responses like:                                                                                     | + |
| 117939 | y                                                                                                               | + |
| 117940 | Yes                                                                                                             | + |
| 117941 | Yes!                                                                                                            | + |
| 117942 | and negative responses like:                                                                                    | + |
| 117943 | N                                                                                                               | + |
| 117944 | No                                                                                                              | + |
| 117945 | Never                                                                                                           | + |
| 117946 | No way!                                                                                                         | + |
| 117947 | should all be recognized as affirmative and negative responses, respectively, by the EREs                       | + |
| 117948 | identified by the <b>yesexpr</b> and <b>noexpr</b> keywords for English language-based locales. There is no     | + |
| 117949 | requirement that multi-line responses nor ambiguous responses like:                                             | + |
| 117950 | no or yes                                                                                                       | + |
| 117951 | yes or no                                                                                                       | + |
| 117952 | maybe                                                                                                           | + |
| 117953 | be correctly classified by either of these EREs. Application writers are encouraged to include                  | + |
| 117954 | locale-specific suggestions for affirmative and negative responses in prompts.                                  |   |
| 117955 | A.7.4 Locale Definition Grammar                                                                                 |   |
| 117956 | There is no additional rationale provided for this section.                                                     |   |
| 117957 | A.7.4.1 Locale Lexical Conventions                                                                              |   |
| 117958 | There is no additional rationale provided for this section.                                                     |   |
| 117959 | A.7.4.2 Locale Grammar                                                                                          |   |
| 117960 | There is no additional rationale provided for this section.                                                     |   |

117961 **A.7.5 Locale Definition Example**

117962 The following is an example of a locale definition file that could be used as input to the *localedf*  
 117963 utility. It assumes that the utility is executed with the *-f* option, naming a charmap file with (at  
 117964 least) the following content:

```
117965 CHARMAP
117966 <space> \x20
117967 <dollar> \x24
117968 <A> \101
117969 <a> \141
117970 <A-acute> \346
117971 <a-acute> \365
117972 <A-grave> \300
117973 <a-grave> \366
117974 \142
117975 <C> \103
117976 <c> \143
117977 <c-cedilla> \347
117978 <d> \x64
117979 <H> \110
117980 <h> \150
117981 <eszet> \xb7
117982 <s> \x73
117983 <z> \x7a
117984 END CHARMAP
```

117985 It should not be taken as complete or to represent any actual locale, but only to illustrate the  
 117986 syntax.

```
117987 #
117988 LC_CTYPE
117989 lower <a>;;<c>;<c-cedilla>;<d>;...;<z>
117990 upper A;B;C;Ç;...;Z
117991 space \x20;\x09;\x0a;\x0b;\x0c;\x0d
117992 blank \040;\011
117993 toupper (<a>,<A>);(b,B);(c,C);(ç,Ç);(d,D);(z,Z)
117994 END LC_CTYPE
117995 #
117996 LC_COLLATE
117997 #
117998 # The following example of collation is based on
117999 # Canadian standard Z243.4.1-1998, "Canadian Alphanumeric
118000 # Ordering Standard for Character Sets of CSA Z234.4 Standard".
118001 # (Other parts of this example locale definition file do not
118002 # purport to relate to Canada, or to any other real culture.)
118003 # The proposed standard defines a 4-weight collation, such that
118004 # in the first pass, characters are compared without regard to
118005 # case or accents; in the second pass, backwards-compare without
118006 # regard to case; in the third pass, forwards-compare without
118007 # regard to diacriticals. In the 3 first passes, non-alphabetic
118008 # characters are ignored; in the fourth pass, only special
118009 # characters are considered, such that "The string that has a
118010 # special character in the lowest position comes first. If two
118011 # strings have a special character in the same position, the
```

```

118012 # collation value of the special character determines ordering.
118013 #
118014 # Only a subset of the character set is used here; mostly to
118015 # illustrate the set-up.
118016 #
118017 collating-symbol <NULL>
118018 collating-symbol <LOW_VALUE>
118019 collating-symbol <LOWER-CASE>
118020 collating-symbol <SUBSCRIPT-LOWER>
118021 collating-symbol <SUPERScript-LOWER>
118022 collating-symbol <UPPER-CASE>
118023 collating-symbol <NO-ACCENT>
118024 collating-symbol <PECULIAR>
118025 collating-symbol <LIGATURE>
118026 collating-symbol <ACUTE>
118027 collating-symbol <GRAVE>
118028 # Further collating-symbols follow.
118029 #
118030 # Properly, the standard does not include any multi-character
118031 # collating elements; the one below is added for completeness.
118032 #
118033 collating_element <ch> from "<c><h>"
118034 collating_element <CH> from "<C><H>"
118035 collating_element <Ch> from "<C><h>"
118036 #
118037 order_start forward;backward;forward;forward,position
118038 #
118039 # Collating symbols are specified first in the sequence to allocate
118040 # basic collation values to them, lower than that of any character.
118041 <NULL>
118042 <LOW_VALUE>
118043 <LOWER-CASE>
118044 <SUBSCRIPT-LOWER>
118045 <SUPERScript-LOWER>
118046 <UPPER-CASE>
118047 <NO-ACCENT>
118048 <PECULIAR>
118049 <LIGATURE>
118050 <ACUTE>
118051 <GRAVE>
118052 <RING-ABOVE>
118053 <DIAERESIS>
118054 <TILDE>
118055 # Further collating symbols are given a basic collating value here.
118056 #
118057 # Here follow special characters.
118058 <space> IGNORE; IGNORE; IGNORE; <space>
118059 # Other special characters follow here.
118060 #
118061 # Here follow the regular characters.
118062 <a> <a>; <NO-ACCENT>; <LOWER-CASE>; IGNORE
118063 <A> <a>; <NO-ACCENT>; <UPPER-CASE>; IGNORE
118064 <a-acute> <a>; <ACUTE>; <LOWER-CASE>; IGNORE

```

```

118065 <A-acute> <a>; <ACUTE>; <UPPER-CASE>; IGNORE
118066 <a-grave> <a>; <GRAVE>; <LOWER-CASE>; IGNORE
118067 <A-grave> <a>; <GRAVE>; <UPPER-CASE>; IGNORE
118068 <ae> " <a><e>; " <LIGATURE><LIGATURE>; \
118069 " <LOWER-CASE><LOWER-CASE>; " IGNORE
118070 <AE> " <a><e>; " <LIGATURE><LIGATURE>; \
118071 " <UPPER-CASE><UPPER-CASE>; " IGNORE
118072 ; <NO-ACCENT>; <LOWER-CASE>; IGNORE
118073 ; <NO-ACCENT>; <UPPER-CASE>; IGNORE
118074 <c> <c>; <NO-ACCENT>; <LOWER-CASE>; IGNORE
118075 <C> <c>; <NO-ACCENT>; <UPPER-CASE>; IGNORE
118076 <ch> <ch>; <NO-ACCENT>; <LOWER-CASE>; IGNORE
118077 <Ch> <ch>; <NO-ACCENT>; <PECULIAR>; IGNORE
118078 <CH> <ch>; <NO-ACCENT>; <UPPER-CASE>; IGNORE
118079 #
118080 # As an example, the strings "Bach" and "bach" could be encoded (for
118081 # compare purposes) as:
118082 # "Bach" ; <a>; <ch>; <LOW_VALUE>; <NO-ACCENT>; <NO-ACCENT>; \
118083 # <NO-ACCENT>; <LOW_VALUE>; <UPPER-CASE>; <LOWER-CASE>; \
118084 # <LOWER-CASE>; <NULL>
118085 # "bach" ; <a>; <ch>; <LOW_VALUE>; <NO-ACCENT>; <NO-ACCENT>; \
118086 # <NO-ACCENT>; <LOW_VALUE>; <LOWER-CASE>; <LOWER-CASE>; \
118087 # <LOWER-CASE>; <NULL>
118088 #
118089 # The two strings are equal in pass 1 and 2, but differ in pass 3.
118090 #
118091 # Further characters follow.
118092 #
118093 UNDEFINED IGNORE; IGNORE; IGNORE; IGNORE
118094 #
118095 order_end
118096 #
118097 END LC_COLLATE
118098 #
118099 LC_MONETARY
118100 int_curr_symbol "USD "
118101 currency_symbol "$"
118102 mon_decimal_point "."
118103 mon_grouping 3;0
118104 positive_sign ""
118105 negative_sign "-"
118106 p_cs_precedes 1
118107 n_sign_posn 0
118108 END LC_MONETARY
118109 #
118110 LC_NUMERIC
118111 copy "US_en.ASCII"
118112 END LC_NUMERIC
118113 #
118114 LC_TIME
118115 abday "Sun"; "Mon"; "Tue"; "Wed"; "Thu"; "Fri"; "Sat"
118116 #
118117 day "Sunday"; "Monday"; "Tuesday"; "Wednesday"; \

```

```

118118 "Thursday"; "Friday"; "Saturday"
118119 #
118120 abmon "Jan"; "Feb"; "Mar"; "Apr"; "May"; "Jun"; \
118121 "Jul"; "Aug"; "Sep"; "Oct"; "Nov"; "Dec"
118122 #
118123 mon "January"; "February"; "March"; "April"; \
118124 "May"; "June"; "July"; "August"; "September"; \
118125 "October"; "November"; "December"
118126 #
118127 d_t_fmt "%a %b %d %T %Z %Y\n"
118128 END LC_TIME
118129 #
118130 LC_MESSAGES
118131 yesexpr "^[yY][[:alpha:]]*" | (OK) "
118132 #
118133 noexpr "^[nN][[:alpha:]]*"
118134 END LC_MESSAGES

```

## 118135 A.8 Environment Variables

### 118136 A.8.1 Environment Variable Definition

118137 The variable *environ* is not intended to be declared in any header, but rather to be declared by  
 118138 the user for accessing the array of strings that is the environment. This is the traditional usage of  
 118139 the symbol. Putting it into a header could break some programs that use the symbol for their  
 118140 own purposes.

118141 The decision to restrict conforming systems to the use of digits, uppercase letters, and  
 118142 underscores for environment variable names allows applications to use lowercase letters in their  
 118143 environment variable names without conflicting with any conforming system.

118144 In addition to the obvious conflict with the shell syntax for positional parameter substitution,  
 118145 some historical applications (including some shells) exclude names with leading digits from the  
 118146 environment.

### 118147 A.8.2 Internationalization Variables

118148 Utilities conforming to the Shell and Utilities volume of POSIX.1-2008 and written in standard C  
 118149 can access the locale variables by issuing the following call:

```
118150 setlocale(LC_ALL, "")
```

118151 If this were omitted, the ISO C standard specifies that the C locale would be used.

118152 The DESCRIPTION of *setlocale()* requires that when setting all categories of a locale, if the value  
 118153 of any of the environment variable searches yields a locale that is not supported (and non-null),  
 118154 the *setlocale()* function returns a null pointer and the global locale is unchanged.

118155 For the standard utilities, if any of the environment variables are invalid, it makes sense to  
 118156 default to an implementation-defined, consistent locale environment. It is more confusing for a

user to have partial settings occur in case of a mistake. All utilities would then behave in one language/cultural environment. Furthermore, it provides a way of forcing the whole environment to be the implementation-defined default. Disastrous results could occur if a pipeline of utilities partially uses the environment variables in different ways. In this case, it would be appropriate for utilities that use *LANG* and related variables to exit with an error if any of the variables are invalid. For example, users typing individual commands at a terminal might want *date* to work if *LC\_MONETARY* is invalid as long as *LC\_TIME* is valid. Since these are conflicting reasonable alternatives, POSIX.1-2008 leaves the results unspecified if the locale environment variables would not produce a complete locale matching the specification of the user.

The locale settings of individual categories cannot be truly independent and still guarantee correct results. For example, when collating two strings, characters must first be extracted from each string (governed by *LC\_CTYPE*) before being mapped to collating elements (governed by *LC\_COLLATE*) for comparison. That is, if *LC\_CTYPE* is causing parsing according to the rules of a large, multi-byte code set (potentially returning 20 000 or more distinct character codeset values), but *LC\_COLLATE* is set to handle only an 8-bit codeset with 256 distinct characters, meaningful results are obviously impossible.

The *LC\_MESSAGES* variable affects the language of messages generated by the standard utilities.

The description of the environment variable names starting with the characters “LC\_” acknowledges the fact that the interfaces presented may be extended as new international functionality is required. In the ISO C standard, names preceded by “LC\_” are reserved in the name space for future categories.

To avoid name clashes, new categories and environment variables are divided into two classifications: “implementation-independent” and “implementation-defined”.

Implementation-independent names will have the following format:

*LC\_NAME*

where *NAME* is the name of the new category and environment variable. Capital letters must be used for implementation-independent names.

Implementation-defined names must be in lowercase letters, as below:

*LC\_name*

### A.8.3 Other Environment Variables

#### COLUMNS, LINES

The default values for the number of column positions, *COLUMNS*, and screen height, *LINES*, are unspecified because historical implementations use different methods to determine values corresponding to the size of the screen in which the utility is run. This size is typically known to the implementation through the value of *TERM*, or by more elaborate methods such as extensions to the *stty* utility or knowledge of how the user is dynamically resizing windows on a bit-mapped display terminal. Users should not need to set these variables in the environment unless there is a specific reason to override the default behavior of the implementation, such as to display data in an area arbitrarily smaller than the terminal or window. Values for these variables that are not decimal integers greater than zero are implicitly undefined values; it is unnecessary to enumerate all of the possible values outside of the acceptable set.



**LOGNAME**

In most implementations, the value of such a variable is easily forged, so security-critical applications should rely on other means of determining user identity. *LOGNAME* is required to be constructed from the portable filename character set for reasons of interchange. No diagnostic condition is specified for violating this rule, and no requirement for enforcement exists. The intent of the requirement is that if extended characters are used, the “guarantee” of portability implied by a standard is void.

**PATH**

Many historical implementations of the Bourne shell do not interpret a trailing <colon> to represent the current working directory and are thus non-conforming. The C Shell and the KornShell conform to POSIX.1-2008 on this point. The usual name of dot may also be used to refer to the current working directory.

Many implementations historically have used a default value of */bin* and */usr/bin* for the *PATH* variable. POSIX.1-2008 does not mandate this default path be identical to that retrieved from *getconf PATH* because it is likely that the standardized utilities may be provided in another directory separate from the directories used by some historical applications.

**SHELL**

The *SHELL* variable names the preferred shell of the user; it is a guide to applications. There is no direct requirement that that shell conform to POSIX.1-2008; that decision should rest with the user. It is the intention of the standard developers that alternative shells be permitted, if the user chooses to develop or acquire one. An operating system that builds its shell into the “kernel” in such a manner that alternative shells would be impossible does not conform to the spirit of POSIX.1-2008.

**TZ**

The quoted form of the timezone variable allows timezone names of the form UTC+1 (or any name that contains the character plus ('+'), the character minus ('-'), or digits), which may be appropriate for countries that do not have an official timezone name. It would be coded as <UTC+1>+1<UTC+2>, which would cause *std* to have a value of UTC+1 and *dst* a value of UTC+2, each with a length of 5 characters. This does not appear to conflict with any existing usage. The characters '<' and '>' were chosen for quoting because they are easier to parse visually than a quoting character that does not provide some sense of bracketing (and in a string like this, such bracketing is helpful). They were also chosen because they do not need special treatment when assigning to the *TZ* variable. Users are often confused by embedding quotes in a string. Because '<' and '>' are meaningful to the shell, the whole string would have to be quoted, but that is easily explained. (Parentheses would have presented the same problems.) Although the '>' symbol could have been permitted in the string by either escaping it or doubling it, it seemed of little value to require that. This could be provided as an extension if there was a need. Timezone names of this new form lead to a requirement that the value of {\_POSIX\_TZNAME\_MAX} change from 3 to 6.

Since the *TZ* environment variable is usually inherited by all applications started by a user after the value of the *TZ* environment variable is changed and since many applications run using the C or POSIX locale, using characters that are not in the portable character set in the *std* and *dst* fields could cause unexpected results.

The format of the *TZ* environment variable is changed in Issue 6 to allow for the quoted form, as defined in earlier versions of the ISO POSIX-1 standard.

IEEE Std 1003.1-2001/Cor 1-2002, item XBD/TC1/D6/7 is applied, adding the *ctime\_r()* and

118246 `localtime_r()` functions to the list of functions that use the *TZ* environment variable.

## 118247 **A.9 Regular Expressions**

118248 Rather than repeating the description of REs for each utility supporting REs, the standard  
 118249 developers preferred a common, comprehensive description of regular expressions in one place.  
 118250 The most common behavior is described here, and exceptions or extensions to this are  
 118251 documented for the respective utilities, as appropriate.

118252 The BRE corresponds to the *ed* or historical *grep* type, and the ERE corresponds to the historical  
 118253 *egrep* type (now *grep -E*).

118254 The text is based on the *ed* description and substantially modified, primarily to aid developers  
 118255 and others in the understanding of the capabilities and limitations of REs. Much of this was  
 118256 influenced by internationalization requirements.

118257 It should be noted that the definitions in this section do not cover the *tr* utility; the *tr* syntax does  
 118258 not employ REs.

118259 The specification of REs is particularly important to internationalization because pattern  
 118260 matching operations are very basic operations in business and other operations. The syntax and  
 118261 rules of REs are intended to be as intuitive as possible to make them easy to understand and use.  
 118262 The historical rules and behavior do not provide that capability to non-English language users,  
 118263 and do not provide the necessary support for commonly used characters and language  
 118264 constructs. It was necessary to provide extensions to the historical RE syntax and rules to  
 118265 accommodate other languages.

118266 As they are limited to bracket expressions, the rationale for these modifications is in XBD [Section](#)  
 118267 [9.3.5](#) (on page 184).

### 118268 **A.9.1 Regular Expression Definitions**

118269 It is possible to determine what strings correspond to subexpressions by recursively applying  
 118270 the leftmost longest rule to each subexpression, but only with the proviso that the overall match  
 118271 is leftmost longest. For example, matching "`\(ac*\\)c*d[ac]*\1`" against *acdacaaa* matches  
 118272 *acdacaaa* (with `\1=a`); simply matching the longest match for "`\(ac*\\)`" would yield `\1=ac`, but  
 118273 the overall match would be smaller (*acdac*). Conceptually, the implementation must examine  
 118274 every possible match and among those that yield the leftmost longest total matches, pick the one  
 118275 that does the longest match for the leftmost subexpression, and so on. Note that this means that  
 118276 matching by subexpressions is context-dependent: a subexpression within a larger RE may  
 118277 match a different string from the one it would match as an independent RE, and two instances of  
 118278 the same subexpression within the same larger RE may match different lengths even in similar  
 118279 sequences of characters. For example, in the ERE "`(a.*b)(a.*b)`", the two identical  
 118280 subexpressions would match four and six characters, respectively, of *acbbaccccb*.

118281 The definition of single character has been expanded to include also collating elements  
 118282 consisting of two or more characters; this expansion is applicable only when a bracket  
 118283 expression is included in the BRE or ERE. An example of such a collating element may be the  
 118284 Dutch *ij*, which collates as a 'y'. In some encodings, a ligature "i with j" exists as a character  
 118285 and would represent a single-character collating element. In another encoding, no such ligature  
 118286 exists, and the two-character sequence *ij* is defined as a multi-character collating element.  
 118287 Outside brackets, the *ij* is treated as a two-character RE and matches the same characters in a  
 118288 string. Historically, a bracket expression only matched a single character. The ISO POSIX-2: 1993

standard required bracket expressions like "[[:lower:]]" to match multi-character collating elements such as "ij". However, this requirement led to behavior that many users did not expect and that could not feasibly be mimicked in user code, and it was rarely if ever implemented correctly. The current standard leaves it unspecified whether a bracket expression matches a multi-character collating element, allowing both historical and ISO POSIX-2:1993 standard implementations to conform.

Also, in the current standard, it is unspecified whether character class expressions like "[[:lower:]]" can include multi-character collating elements like "ij"; hence "[[:lower:]]" can match "ij", and "[^[:lower:]]" can fail to match "ij". Common practice is for a character class expression to match a collating element if it matches the collating element's first character.

## A.9.2 Regular Expression General Requirements

The definition of which sequence is matched when several are possible is based on the leftmost-longest rule historically used by deterministic recognizers. This rule is easier to define and describe, and arguably more useful, than the first-match rule historically used by non-deterministic recognizers. It is thought that dependencies on the choice of rule are rare; carefully contrived examples are needed to demonstrate the difference.

A formal expression of the leftmost-longest rule is:

The search is performed as if all possible suffixes of the string were tested for a prefix matching the pattern; the longest suffix containing a matching prefix is chosen, and the longest possible matching prefix of the chosen suffix is identified as the matching sequence.

Historically, most RE implementations only match lines, not strings. However, that is more an effect of the usage than of an inherent feature of REs themselves. Consequently, POSIX.1-2008 does not regard <newline> characters as special; they are ordinary characters, and both a <period> and a non-matching list can match them. Those utilities (like *grep*) that do not allow <newline> characters to match are responsible for eliminating any <newline> from strings before matching against the RE. The *regcomp()* function, however, can provide support for such processing without violating the rules of this section.

Some implementations of *egrep* have had very limited flexibility in handling complex EREs. POSIX.1-2008 does not attempt to define the complexity of a BRE or ERE, but does place a lower limit on it—any RE must be handled, as long as it can be expressed in 256 bytes or less. (Of course, this does not place an upper limit on the implementation.) There are historical programs using a non-deterministic-recognizer implementation that should have no difficulty with this limit. It is possible that a good approach would be to attempt to use the faster, but more limited, deterministic recognizer for simple expressions and to fall back on the non-deterministic recognizer for those expressions requiring it. Non-deterministic implementations must be careful to observe the rules on which match is chosen; the longest match, not the first match, starting at a given character is used.

The term “invalid” highlights a difference between this section and some others: POSIX.1-2008 frequently avoids mandating of errors for syntax violations because they can be used by implementors to trigger extensions. However, the authors of the internationalization features of REs wanted to mandate errors for certain conditions to identify usage problems or non-portable constructs. These are identified within this rationale as appropriate. The remaining syntax violations have been left implicitly or explicitly undefined. For example, the BRE construct "{1,2,3}" does not comply with the grammar. A conforming application cannot rely on it producing an error nor matching the literal characters "{1,2,3}".

118336 The term “undefined” was used in favor of “unspecified” because many of the situations are  
 118337 considered errors on some implementations, and the standard developers considered that  
 118338 consistency throughout the section was preferable to mixing undefined and unspecified.

### 118339 **A.9.3 Basic Regular Expressions**

118340 There is no additional rationale provided for this section.

#### 118341 *A.9.3.1 BREs Matching a Single Character or Collating Element*

118342 There is no additional rationale provided for this section.

#### 118343 *A.9.3.2 BRE Ordinary Characters*

118344 There is no additional rationale provided for this section.

#### 118345 *A.9.3.3 BRE Special Characters*

118346 There is no additional rationale provided for this section.

#### 118347 *A.9.3.4 Periods in BREs*

118348 There is no additional rationale provided for this section.

#### 118349 *A.9.3.5 RE Bracket Expression*

118350 Range expressions are, historically, an integral part of REs. However, the requirements of  
 118351 “natural language behavior” and portability do conflict. In the POSIX locale, ranges must be  
 118352 treated according to the collating sequence and include such characters that fall within the range  
 118353 based on that collating sequence, regardless of character values. In other locales, ranges have  
 118354 unspecified behavior.

118355 Some historical implementations allow range expressions where the ending range point of one  
 118356 range is also the starting point of the next (for instance, "[a-m-o]"). This behavior should not  
 118357 be permitted, but to avoid breaking historical implementations, it is now *undefined* whether it is  
 118358 a valid expression and how it should be interpreted.

118359 Current practice in *awk* and *lex* is to accept escape sequences in bracket expressions as per XBD  
 118360 [Table 5-1](#) (on page 121), while the normal ERE behavior is to regard such a sequence as  
 118361 consisting of two characters. Allowing the *awk/lex* behavior in EREs would change the normal  
 118362 behavior in an unacceptable way; it is expected that *awk* and *lex* will decode escape sequences in  
 118363 EREs before passing them to *regcomp()* or comparable routines. Each utility describes the escape  
 118364 sequences it accepts as an exception to the rules in this section; the list is not the same, for  
 118365 historical reasons.

118366 As noted previously, the new syntax and rules have been added to accommodate other  
 118367 languages than English. The remainder of this section describes the rationale for these  
 118368 modifications.

118369 In the POSIX locale, a regular expression that starts with a range expression matches a set of  
 118370 strings that are contiguously sorted, but this is not necessarily true in other locales. For example,  
 118371 a French locale might have the following behavior:

```

118372 $ ls
118373 alpha Alpha estimé ESTIMÉ été eurêka
118374 $ ls [a-e]*
118375 alpha Alpha estimé eurêka

```

Such disagreements between matching and contiguous sorting are unavoidable because POSIX sorting cannot be implemented in terms of a deterministic finite-state automaton (DFA), but range expressions by design are implementable in terms of DFAs.

Historical implementations used native character order to interpret range expressions. The ISO POSIX-2:1993 standard instead required collating element order (CEO): the order that collating elements were specified between the **order\_start** and **order\_end** keywords in the *LC\_COLLATE* category of the current locale. CEO had some advantages in portability over the native character order, but it also had some disadvantages:

- CEO could not feasibly be mimicked in user code, leading to inconsistencies between POSIX matchers and matchers in popular user programs like Emacs, *ksh*, and Perl.
- CEO caused range expressions to match accented and capitalized letters contrary to many users' expectations. For example, "[a-e]" typically matched both 'É' and 'á' but neither 'A' nor 'é'.
- CEO was not consistent across implementations. In practice, CEO was often less portable than native character order. For example, it was common for the CEOs of two implementation-supplied locales to disagree, even if both locales were named "da\_DK".

Because of these problems, some implementations of regular expressions continued to use native character order. Others used the collation sequence, which is more consistent with sorting than either CEO or native order, but which departs further from the traditional POSIX semantics because it generally requires "[a-e]" to match either 'A' or 'E' but not both. As a result of this kind of implementation variation, programmers who wanted to write portable regular expressions could not rely on the ISO POSIX-2:1993 standard guarantees in practice.

While revising the standard, lengthy consideration was given to proposals to attack this problem by adding an API for querying the CEO to allow user-mode matchers, but none of these proposals had implementation experience and none achieved consensus. Leaving the standard alone was also considered, but rejected due to the problems described above.

The current standard leaves unspecified the behavior of a range expression outside the POSIX locale. This makes it clearer that conforming applications should avoid range expressions outside the POSIX locale, and it allows implementations and compatible user-mode matchers to interpret range expressions using native order, CEO, collation sequence, or other, more advanced techniques. The concerns which led to this change were raised in IEEE PASC interpretation 1003.2 #43 and others, and related to ambiguities in the specification of how multi-character collating elements should be handled in range expressions. These ambiguities had led to multiple interpretations of the specification, in conflicting ways, which led to varying implementations. As noted above, efforts were made to resolve the differences, but no solution has been found that would be specific enough to allow for portable software while not invalidating existing implementations.

The standard developers recognize that collating elements are important, such elements being common in several European languages; for example, 'ch' or 'll' in traditional Spanish; 'aa' in several Scandinavian languages. Existing internationalized implementations have processed, and continue to process, these elements in range expressions. Efforts are expected to continue in the future to find a way to define the behavior of these elements precisely and portably.

The ISO POSIX-2:1993 standard required "[b-a]" to be an invalid expression in the POSIX

118420 locale, but this requirement has been relaxed in this version of the standard so that "[b-a]" can  
 118421 instead be treated as a valid expression that does not match any string.

118422 The standard specifies three possible behaviors for regular expressions such as "[alpha:]". +  
 118423 One behavior is the traditional implementation, which behaves like "[ahlp]". Another, for +  
 118424 alignment with the *tr* utility, is to treat it like "[[:alpha:]]". And finally, the standard allows +  
 118425 rejecting the regular expression as invalid, as a means of alerting a user to the non-portable +  
 118426 aspect of that regular expression. The set of regular expressions with this undefined behavior is +  
 118427 limited solely to the expressions where the outer '[' and ']' of the bracket expression can be +  
 118428 confused with the missing bracket pair '[' and ']' necessary to form a collating symbol, +  
 118429 equivalence class, or character class; thus "[\_alpha:]" or "[::]" do not trigger the +  
 118430 unspecified behavior.

#### 118431 A.9.3.6 BREs Matching Multiple Characters

118432 The limit of nine back-references to subexpressions in the RE is based on the use of a single-digit  
 118433 identifier; increasing this to multiple digits would break historical applications. This does not  
 118434 imply that only nine subexpressions are allowed in REs. The following is a valid BRE with ten  
 118435 subexpressions:

```
118436 \(\(ab\) *c\) *d\) \(ef\) *\ (gh\) \{2\} \(ij\) *\ (kl\) *\ (mn\) *\ (op\) *\ (qr\) *
```

118437 The standard developers regarded the common historical behavior, which supported "\n\*", but  
 118438 not "\n{min,max\}", "\(...\)\*", or "\(...\){min,max\}", as a non-intentional  
 118439 result of a specific implementation, and they supported both duplication and interval  
 118440 expressions following subexpressions and back-references.

118441 The changes to the processing of the back-reference expression remove an unspecified or  
 118442 ambiguous behavior in the Shell and Utilities volume of POSIX.1-2008, aligning it with the  
 118443 requirements specified for the *regcomp()* expression, and is the result of PASC Interpretation  
 118444 1003.2-92 #43 submitted for the ISO POSIX-2: 1993 standard.

#### 118445 A.9.3.7 BRE Precedence

118446 There is no additional rationale provided for this section.

#### 118447 A.9.3.8 BRE Expression Anchoring

118448 Often, the <dollar-sign> is viewed as matching the ending <newline> in text files. This is not  
 118449 strictly true; the <newline> is typically eliminated from the strings to be matched, and the  
 118450 <dollar-sign> matches the terminating null character.

118451 The ability of '^', '\$', and '\*' to be non-special in certain circumstances may be confusing to  
 118452 some programmers, but this situation was changed only in a minor way from historical practice  
 118453 to avoid breaking many historical scripts. Some consideration was given to making the use of  
 118454 the anchoring characters undefined if not escaped and not at the beginning or end of strings.  
 118455 This would cause a number of historical BREs, such as "2^10", "\$HOME", and "\$1.35", that  
 118456 relied on the characters being treated literally, to become invalid.

118457 However, one relatively uncommon case was changed to allow an extension used on some  
 118458 implementations. Historically, the BREs "^foo" and "\(^foo\)" did not match the same  
 118459 string, despite the general rule that subexpressions and entire BREs match the same strings. To  
 118460 increase consensus, POSIX.1-2008 has allowed an extension on some implementations to treat  
 118461 these two cases in the same way by declaring that anchoring *may* occur at the beginning or end  
 118462 of a subexpression. Therefore, portable BREs that require a literal <circumflex> at the beginning

118463 or a <dollar-sign> at the end of a subexpression must escape them. Note that a BRE such as  
 118464 "a\ (^bc\ )" will either match "a^bc" or nothing on different systems under the rules.

118465 ERE anchoring has been different from BRE anchoring in all historical systems. An unescaped  
 118466 anchor character has never matched its literal counterpart outside a bracket expression. Some  
 118467 implementations treated "foo\$bar" as a valid expression that never matched anything; others  
 118468 treated it as invalid. POSIX.1-2008 mandates the former, valid unmatched behavior.

118469 Some implementations have extended the BRE syntax to add alternation. For example, the  
 118470 subexpression "\ (foo\$|bar\ )" would match either "foo" at the end of the string or "bar"  
 118471 anywhere. The extension is triggered by the use of the undefined "\|" sequence. Because the  
 118472 BRE is undefined for portable scripts, the extending system is free to make other assumptions,  
 118473 such that the '\$' represents the end-of-line anchor in the middle of a subexpression. If it were  
 118474 not for the extension, the '\$' would match a literal <dollar-sign> under the rules.

#### 118475 A.9.4 Extended Regular Expressions

118476 As with BREs, the standard developers decided to make the interpretation of escaped ordinary  
 118477 characters undefined.

118478 The <right-parenthesis> is not listed as an ERE special character because it is only special in the  
 118479 context of a preceding <left-parenthesis>. If found without a preceding <left-parenthesis>, the  
 118480 <right-parenthesis> has no special meaning.

118481 The interval expression, "{m, n}", has been added to EREs. Historically, the interval expression  
 118482 has only been supported in some ERE implementations. The standard developers estimated that  
 118483 the addition of interval expressions to EREs would not decrease consensus and would also make  
 118484 BREs more of a subset of EREs than in many historical implementations.

118485 It was suggested that, in addition to interval expressions, back-references ('\n') should also be  
 118486 added to EREs. This was rejected by the standard developers as likely to decrease consensus.

118487 In historical implementations, multiple duplication symbols are usually interpreted from left to  
 118488 right and treated as additive. As an example, "a+b" matches zero or more instances of 'a'  
 118489 followed by a 'b'. In POSIX.1-2008, multiple duplication symbols are undefined; that is, they  
 118490 cannot be relied upon for conforming applications. One reason for this is to provide some scope  
 118491 for future enhancements.

118492 The precedence of operations differs between EREs and those in *lex*; in *lex*, for historical reasons,  
 118493 interval expressions have a lower precedence than concatenation.

##### 118494 A.9.4.1 EREs Matching a Single Character or Collating Element

118495 There is no additional rationale provided for this section.

##### 118496 A.9.4.2 ERE Ordinary Characters

118497 There is no additional rationale provided for this section.

118498 A.9.4.3 *ERE Special Characters*

118499 There is no additional rationale provided for this section.

118500 A.9.4.4 *Periods in EREs*

118501 There is no additional rationale provided for this section.

118502 A.9.4.5 *ERE Bracket Expression*

118503 There is no additional rationale provided for this section.

118504 A.9.4.6 *EREs Matching Multiple Characters*

118505 There is no additional rationale provided for this section.

118506 A.9.4.7 *ERE Alternation*

118507 There is no additional rationale provided for this section.

118508 A.9.4.8 *ERE Precedence*

118509 There is no additional rationale provided for this section.

118510 A.9.4.9 *ERE Expression Anchoring*

118511 There is no additional rationale provided for this section.

118512 **A.9.5 Regular Expression Grammar**

118513 The grammars are intended to represent the range of acceptable syntaxes available to  
 118514 conforming applications. There are instances in the text where undefined constructs are  
 118515 described; as explained previously, these allow implementation extensions. There is no intended  
 118516 requirement that an implementation extension must somehow fit into the grammars shown  
 118517 here.

118518 The BRE grammar does not permit L\_ANCHOR or R\_ANCHOR inside "\ (" and "\ )" (which  
 118519 implies that '^' and '\$' are ordinary characters). This reflects the semantic limits on the  
 118520 application, as noted in XBD [Section 9.3.8](#) (on page 187). Implementations are permitted to  
 118521 extend the language to interpret '^' and '\$' as anchors in these locations, and as such,  
 118522 conforming applications cannot use unescaped '^' and '\$' in positions inside "\ (" and "\ )"   
 118523 that might be interpreted as anchors.

118524 The ERE grammar does not permit several constructs that XBD [Section 9.4.2](#) (on page 188) and  
 118525 [Section 9.4.3](#) (on page 188) specify as having undefined results:

- 118526 • ORD\_CHAR preceded by <backslash>
- 118527 • ERE\_dupl\_symbol(s) appearing first in an ERE, or immediately following '|', '^', or '('
- 118528 • '{' not part of a valid ERE\_dupl\_symbol



- 118529 • ' | ' appearing first or last in an ERE, or immediately following ' | ' or ' (', or  
118530 immediately preceding ' ) '

118531 Implementations are permitted to extend the language to allow these. Conforming applications  
118532 cannot use such constructs.

## 118533 A.9.5.1 BRE/ERE Grammar Lexical Conventions

118534 There is no additional rationale provided for this section.

## 118535 A.9.5.2 RE and Bracket Expression Grammar

The removal of the *Back\_open\_paren Back\_close\_paren* option from the *nondupl\_RE* specification is the result of PASC Interpretation 1003.2-92 #43 submitted for the ISO POSIX-2:1993 standard. Although the grammar required support for null subexpressions, this section does not describe the meaning of, and historical practice did not support, this construct.

118540 A.9.5.3 *ERE Grammar*

118541 There is no additional rationale provided for this section.

## 118542 A.10 Directory Structure and Devices

## 118543 A.10.1 Directory Structure and Files

A description of the historical **/usr/tmp** was omitted, removing any concept of differences in emphasis between the **/** and **/usr** directories. The descriptions of **/bin**, **/usr/bin**, **/lib**, and **/usr/lib** were omitted because they are not useful for applications. In an early draft, a distinction was made between system and application directory usage, but this was not found to be useful.

The directories `/` and `/dev` are included because the notion of a hierarchical directory structure is key to other information presented elsewhere in POSIX.1-2008. In early drafts, it was argued that special devices and temporary files could conceivably be handled without a directory structure on some implementations. For example, the system could treat the characters `"/tmp"` as a special token that would store files using some non-POSIX file system structure. This notion was rejected by the standard developers, who required that all the files in this section be implemented via POSIX file systems.

The **/tmp** directory is retained in POSIX.1-2008 to accommodate historical applications that assume its availability. Implementations are encouraged to provide suitable directory names in the environment variable *TMPDIR* and applications are encouraged to use the contents of *TMPDIR* for creating temporary files.

118559 The standard files **/dev/null** and **/dev/tty** are required to be both readable and writable to allow  
118560 applications to have the intended historical access to these files.

118561 The standard file **/dev/console** has been added for alignment with the Single UNIX  
118562 Specification.

## 118563 A.10.2 Output Devices and Terminal Types

118564 IEEE Std 1003.1-2001/Cor 2-2004, item XBD/TC2/D6/17 is applied, making it clear that the  
118565 requirements for documenting terminal support are in the system documentation.

## 118566 A.11 General Terminal Interface

118567 If the implementation does not support this interface on any device types, it should behave as if  
118568 it were being used on a device that is not a terminal device (in most cases *errno* will be set to  
118569 [ENOTTY] on return from functions defined by this interface). This is based on the fact that  
118570 many applications are written to run both interactively and in some non-interactive mode, and  
118571 they adapt themselves at runtime. Requiring that they all be modified to test an environment  
118572 variable to determine whether they should try to adapt is unnecessary. On a system that  
118573 provides no general terminal interface, providing all the entry points as stubs that return  
118574 [ENOTTY] (or an equivalent, as appropriate) has the same effect and requires no changes to the  
118575 application.

118576 Although the needs of both interface implementors and application developers were addressed  
118577 throughout POSIX.1-2008, this section pays more attention to the needs of the latter. This is  
118578 because, while many aspects of the programming interface can be hidden from the user by the  
118579 application developer, the terminal interface is usually a large part of the user interface.  
118580 Although to some extent the application developer can build missing features or work around  
118581 inappropriate ones, the difficulties of doing that are greater in the terminal interface than  
118582 elsewhere. For example, efficiency prohibits the average program from interpreting every  
118583 character passing through it in order to simulate character erase, line kill, and so on. These  
118584 functions should usually be done by the operating system, possibly at the interrupt level.

118585 The *tc\**() functions were introduced as a way of avoiding the problems inherent in the  
118586 traditional *ioctl*() function and in variants of it that were proposed. For example, *tcsetattr*() is  
118587 specified in place of the use of the TCSETA *ioctl*() command function. This allows specification  
118588 of all the arguments in a manner consistent with the ISO C standard unlike the varying third  
118589 argument of *ioctl*(), which is sometimes a pointer (to any of many different types) and  
118590 sometimes an *int*.

118591 The advantages of this new method include:

- 118592 • It allows strict type checking.
- 118593 • The direction of transfer of control data is explicit.
- 118594 • Portable capabilities are clearly identified.
- 118595 • The need for a general interface routine is avoided.
- 118596 • Size of the argument is well-defined (there is only one type).

118597 The disadvantages include:

- 118598 • No historical implementation used the new method.
- 118599 • There are many small routines instead of one general-purpose one.
- 118600 • The historical parallel with *fcntl*() is broken.

118601 The issue of modem control was excluded from POSIX.1-2008 on the grounds that:

- 118602 • It was concerned with setting and control of hardware timers.
- 118603 • The appropriate timers and settings vary widely internationally.
- 118604 • Feedback from European computer manufacturers indicated that this facility was not
- 118605 consistent with European needs and that specification of such a facility was not a
- 118606 requirement for portability.

### 118607 A.11.1 Interface Characteristics

#### 118608 A.11.1.1 Opening a Terminal Device File

118609 The O\_TTY\_INIT flag for *open()* has been added to POSIX.1-2008 to solve a problem  
 118610 encountered by applications written for earlier versions of this standard which need to open a  
 118611 modem or similar device and initialize all of the parameter settings. Using the  
 118612 *tcgetattr()*-modify-*tcsetattr()* method mandated by the standard could result in non-conforming  
 118613 behavior if the device had previously been used with non-conforming parameter settings, on  
 118614 implementations which do not reset the parameter settings in between the last close of the  
 118615 device by one application and the first open by another application. To avoid this problem, some  
 118616 application developers were resorting to using *memset()* to zero the **termios** structure before  
 118617 setting all of the standard parameters, but this risks non-conforming behavior on systems where  
 118618 some non-standard parameter needs a non-zero value in order for the terminal to behave in a  
 118619 conforming manner.

118620 On systems which do reset the parameter settings to defaults between uses of a terminal device,  
 118621 it is expected that either O\_TTY\_INIT will have the value zero or *open(ttypath,*  
 118622 *O\_RDWR|O\_TTY\_INIT)* will do nothing additional.

118623 The standard developers considered an alternative solution of a special *fildev* argument for the  
 118624 *tcgetattr()* call to obtain default parameters. However, this would not be adequate if a system  
 118625 supports several different types of terminal device and the default settings need to differ  
 118626 between the different types. With the O\_TTY\_INIT open flag, the implementor can determine  
 118627 which device type is being opened.

118628 The standard developers also considered a special POSIX\_TTY\_INIT value for the **termios**  
 118629 structure used in *tcsetattr()*, which would reset the values if used immediately after an *open()*  
 118630 call. However, it was felt that this would lead to confusion amongst application developers who  
 118631 wanted to reset the parameters at other points, and implementations might diverge.

#### 118632 A.11.1.2 Process Groups

118633 There is a potential race when the members of the foreground process group on a terminal leave  
 118634 that process group, either by exit or by changing process groups. After the last process exits the  
 118635 process group, but before the foreground process group ID of the terminal is changed (usually  
 118636 by a job control shell), it would be possible for a new process to be created with its process ID  
 118637 equal to the terminal's foreground process group ID. That process might then become the  
 118638 process group leader and accidentally be placed into the foreground on a terminal that was not  
 118639 necessarily its controlling terminal. As a result of this problem, the controlling terminal is  
 118640 defined to not have a foreground process group during this time.

118641 The cases where a controlling terminal has no foreground process group occur when all  
 118642 processes in the foreground process group either terminate and are waited for or join other  
 118643 process groups via *setpgid()* or *setsid()*. If the process group leader terminates, this is the first

case described; if it leaves the process group via *setpgid()*, this is the second case described (a process group leader cannot successfully call *setsid()*). When one of those cases causes a controlling terminal to have no foreground process group, it has two visible effects on applications. The first is the value returned by *tcgetpgrp()*. The second (which occurs only in the case where the process group leader terminates) is the sending of signals in response to special input characters. The intent of POSIX.1-2008 is that no process group be wrongly identified as the foreground process group by *tcgetpgrp()* or unintentionally receive signals because of placement into the foreground.

In 4.3 BSD, the old process group ID continues to be used to identify the foreground process group and is returned by the function equivalent to *tcgetpgrp()*. In that implementation it is possible for a newly created process to be assigned the same value as a process ID and then form a new process group with the same value as a process group ID. The result is that the new process group would receive signals from this terminal for no apparent reason, and POSIX.1-2008 precludes this by forbidding a process group from entering the foreground in this way. It would be more direct to place part of the requirement made by the last sentence under *fork()*, but there is no convenient way for that section to refer to the value that *tcgetpgrp()* returns, since in this case there is no process group and thus no process group ID.

One possibility for a conforming implementation is to behave similarly to 4.3 BSD, but to prevent this reuse of the ID, probably in the implementation of *fork()*, as long as it is in use by the terminal.

Another possibility is to recognize when the last process stops using the terminal's foreground process group ID, which is when the process group lifetime ends, and to change the terminal's foreground process group ID to a reserved value that is never used as a process ID or process group ID. (See the definition of *process group lifetime* in the definitions section.) The process ID can then be reserved until the terminal has another foreground process group.

The 4.3 BSD implementation permits the leader (and only member) of the foreground process group to leave the process group by calling the equivalent of *setpgid()* and to later return, expecting to return to the foreground. There are no known application needs for this behavior, and POSIX.1-2008 neither requires nor forbids it (except that it is forbidden for session leaders) by leaving it unspecified.

#### A.11.1.3 The Controlling Terminal

POSIX.1-2008 does not specify a mechanism by which to allocate a controlling terminal. This is normally done by a system utility (such as *getty*) and is considered an administrative feature outside the scope of POSIX.1-2008.

Historical implementations allocate controlling terminals on certain *open()* calls. Since *open()* is part of POSIX.1, its behavior had to be dealt with. The traditional behavior is not required because it is not very straightforward or flexible for either implementations or applications. However, because of its prevalence, it was not practical to disallow this behavior either. Thus, a mechanism was standardized to ensure portable, predictable behavior in *open()*.

Some historical implementations deallocate a controlling terminal on the last system-wide close. This behavior is neither required nor prohibited. Even on implementations that do provide this behavior, applications generally cannot depend on it due to its system-wide nature.

## 118686 A.11.1.4 Terminal Access Control

118687 The access controls described in this section apply only to a process that is accessing its  
 118688 controlling terminal. A process accessing a terminal that is not its controlling terminal is  
 118689 effectively treated the same as a member of the foreground process group. While this may seem  
 118690 unintuitive, note that these controls are for the purpose of job control, not security, and job  
 118691 control relates only to the controlling terminal of a process. Normal file access permissions  
 118692 handle security.

118693 If the process calling *read()* or *write()* is in a background process group that is orphaned, it is not  
 118694 desirable to stop the process group, as it is no longer under the control of a job control shell that  
 118695 could put it into the foreground again. Accordingly, calls to *read()* or *write()* functions by such  
 118696 processes receive an immediate error return. This is different from 4.2 BSD, which kills orphaned  
 118697 processes that receive terminal stop signals.

118698 The foreground/background/orphaned process group check performed by the terminal driver  
 118699 must be repeatedly performed until the calling process moves into the foreground or until the  
 118700 process group of the calling process becomes orphaned. That is, when the terminal driver  
 118701 determines that the calling process is in the background and should receive a job control signal,  
 118702 it sends the appropriate signal (SIGTTIN or SIGTTOU) to every process in the process group of  
 118703 the calling process and then it allows the calling process to immediately receive the signal. The  
 118704 latter is typically performed by blocking the process so that the signal is immediately noticed.  
 118705 Note, however, that after the process finishes receiving the signal and control is returned to the  
 118706 driver, the terminal driver must re-execute the foreground/background/orphaned process  
 118707 group check. The process may still be in the background, either because it was continued in the  
 118708 background by a job control shell, or because it caught the signal and did nothing.

118709 The terminal driver repeatedly performs the foreground/background/orphaned process group  
 118710 checks whenever a process is about to access the terminal. In the case of *write()* or the control  
 118711 *tc\*()* functions, the check is performed at the entry of the function. In the case of *read()*, the  
 118712 check is performed not only at the entry of the function, but also after blocking the process to  
 118713 wait for input characters (if necessary). That is, once the driver has determined that the process  
 118714 calling the *read()* function is in the foreground, it attempts to retrieve characters from the input  
 118715 queue. If the queue is empty, it blocks the process waiting for characters. When characters are  
 118716 available and control is returned to the driver, the terminal driver must return to the repeated  
 118717 foreground/background/orphaned process group check again. The process may have moved  
 118718 from the foreground to the background while it was blocked waiting for input characters.

## 118719 A.11.1.5 Input Processing and Reading Data

118720 There is no additional rationale provided for this section.

## 118721 A.11.1.6 Canonical Mode Input Processing

118722 The term “character” is intended here. ERASE should erase the last character, not the last byte.  
 118723 In the case of multi-byte characters, these two may be different.

118724 4.3 BSD has a WERASE character that erases the last “word” typed (but not any preceding  
 118725 <blank> or <tab> characters). A word is defined as a sequence of non-<blank> characters, with  
 118726 <tab> characters counted as <blank> characters. Like ERASE, WERASE does not erase beyond  
 118727 the beginning of the line. This WERASE feature has not been specified in POSIX.1 because it is  
 118728 difficult to define in the international environment. It is only useful for languages where words  
 118729 are delimited by <blank> characters. In some ideographic languages, such as Japanese and  
 118730 Chinese, words are not delimited at all. The WERASE character should presumably go back to  
 118731 the beginning of a sentence in those cases; practically, this means it would not be used much for

118732 those languages.

118733 It should be noted that there is a possible inherent deadlock if the application and  
118734 implementation conflict on the value of {MAX\_CANON}. With ICANON set (if IXOFF is  
118735 enabled) and more than {MAX\_CANON} characters transmitted without a <linefeed>,  
118736 transmission will be stopped, the <linefeed> (or <carriage-return> when ICRLF is set) will never  
118737 arrive, and the *read()* will never be satisfied.

118738 An application should not set IXOFF if it is using canonical mode unless it knows that (even in  
118739 the face of a transmission error) the conditions described previously cannot be met or unless it is  
118740 prepared to deal with the possible deadlock in some other way, such as timeouts.

118741 It should also be noted that this can be made to happen in non-canonical mode if the trigger  
118742 value for sending IXOFF is less than VMIN and VTIME is zero.

#### 118743 A.11.1.7 Non-Canonical Mode Input Processing

118744 Some points to note about MIN and TIME:

- 118745 1. The interactions of MIN and TIME are not symmetric. For example, when MIN>0 and  
118746 TIME=0, TIME has no effect. However, in the opposite case where MIN=0 and TIME>0,  
118747 both MIN and TIME play a role in that MIN is satisfied with the receipt of a single  
118748 character.
- 118749 2. Also note that in case A (MIN>0, TIME>0), TIME represents an inter-character timer,  
118750 while in case C (MIN=0, TIME>0), TIME represents a read timer.

118751 These two points highlight the dual purpose of the MIN/TIME feature. Cases A and B, where  
118752 MIN>0, exist to handle burst-mode activity (for example, file transfer programs) where a  
118753 program would like to process at least MIN characters at a time. In case A, the inter-character  
118754 timer is activated by a user as a safety measure; in case B, it is turned off.

118755 Cases C and D exist to handle single-character timed transfers. These cases are readily adaptable  
118756 to screen-based applications that need to know if a character is present in the input queue before  
118757 refreshing the screen. In case C, the read is timed; in case D, it is not.

118758 Another important note is that MIN is always just a minimum. It does not denote a record  
118759 length. That is, if a program does a read of 20 bytes, MIN is 10, and 25 characters are present, 20  
118760 characters are returned to the user. In the special case of MIN=0, this still applies: if more than  
118761 one character is available, they all will be returned immediately.

#### 118762 A.11.1.8 Writing Data and Output Processing

118763 There is no additional rationale provided for this section.

#### 118764 A.11.1.9 Special Characters

118765 There is no additional rationale provided for this section.

#### 118766 A.11.1.10 Modem Disconnect

118767 There is no additional rationale provided for this section.

118768 A.11.1.11 *Closing a Terminal Device File*

118769 POSIX.1-2008 does not specify that a *close()* on a terminal device file include the equivalent of a  
118770 call to *tcflow(fd,TCOON)*.

118771 An implementation that discards output at the time *close()* is called after reporting the return  
118772 value to the *write()* call that data was written does not conform with POSIX.1-2008. An  
118773 application has functions such as *tcdrain()*, *tcflush()*, and *tcflow()* available to obtain the detailed  
118774 behavior it requires with respect to flushing of output.

118775 At the time of the last close on a terminal device, an application relinquishes any ability to exert  
118776 flow control via *tcflow()*.

118777 **A.11.2 Parameters that Can be Set**118778 A.11.2.1 *The termios Structure*

118779 This structure is part of an interface that, in general, retains the historic grouping of flags.  
118780 Although a more optimal structure for implementations may be possible, the degree of change  
118781 to applications would be significantly larger.

118782 A.11.2.2 *Input Modes*

118783 Some historical implementations treated a long break as multiple events, as many as one per  
118784 character time. The wording in POSIX.1 explicitly prohibits this.

118785 Although the ISTRIP flag is normally superfluous with today's terminal hardware and software,  
118786 it is historically supported. Therefore, applications may be using ISTRIP, and there is no  
118787 technical problem with supporting this flag. Also, applications may wish to receive only 7-bit  
118788 input bytes and may not be connected directly to the hardware terminal device (for example,  
118789 when a connection traverses a network).

118790 Also, there is no requirement in general that the terminal device ensures that high-order bits  
118791 beyond the specified character size are cleared. ISTRIP provides this function for 7-bit  
118792 characters, which are common.

118793 In dealing with multi-byte characters, the consequences of a parity error in such a character, or  
118794 in an escape sequence affecting the current character set, are beyond the scope of POSIX.1 and  
118795 are best dealt with by the application processing the multi-byte characters.

118796 A.11.2.3 *Output Modes*

118797 POSIX.1 does not describe post-processing of output to a terminal or detailed control of that  
118798 from a conforming application. (That is, translation of <newline> to <carriage-return> followed  
118799 by <linefeed> or <tab> processing.) There is nothing that a conforming application should do to  
118800 its output for a terminal because that would require knowledge of the operation of the terminal.  
118801 It is the responsibility of the operating system to provide post-processing appropriate to the  
118802 output device, whether it is a terminal or some other type of device.

118803 Extensions to POSIX.1 to control the type of post-processing already exist and are expected to  
118804 continue into the future. The control of these features is primarily to adjust the interface between  
118805 the system and the terminal device so the output appears on the display correctly. This should  
118806 be set up before use by any application.

118807 In general, both the input and output modes should not be set absolutely, but rather modified  
 118808 from the inherited state.

#### 118809 A.11.2.4 Control Modes

118810 This section could be misread that the symbol “CSIZE” is a title in the **termios** *c\_cflag* field.  
 118811 Although it does serve that function, it is also a required symbol, as a literal reading of POSIX.1  
 118812 (and the caveats about typography) would indicate.

#### 118813 A.11.2.5 Local Modes

118814 Non-canonical mode is provided to allow fast bursts of input to be read efficiently while still  
 118815 allowing single-character input.

118816 The ECHONL function historically has been in many implementations. Since there seems to be  
 118817 no technical problem with supporting ECHONL, it is included in POSIX.1 to increase consensus.

118818 The alternate behavior possible when ECHOK or ECHOE are specified with ICANON is  
 118819 permitted as a compromise depending on what the actual terminal hardware can do. Erasing  
 118820 characters and lines is preferred, but is not always possible.

#### 118821 A.11.2.6 Special Control Characters

118822 Permitting VMIN and VTIME to overlap with VEOF and VEOL was a compromise for historical  
 118823 implementations. Only when backwards-compatibility of object code is a serious concern to an  
 118824 implementor should an implementation continue this practice. Correct applications that work  
 118825 with the overlap (at the source level) should also work if it is not present, but not the reverse.

## 118826 A.12 Utility Conventions

### 118827 A.12.1 Utility Argument Syntax

118828 The standard developers considered that recent trends toward diluting the SYNOPSIS sections  
 118829 of historical reference pages to the equivalent of:

118830 `command [options] [operands]`

118831 were a disservice to the reader. Therefore, considerable effort was placed into rigorous  
 118832 definitions of all the command line arguments and their interrelationships. The relationships  
 118833 depicted in the synopses are normative parts of POSIX.1-2008; this information is sometimes  
 118834 repeated in textual form, but that is only for clarity within context.

118835 The use of “undefined” for conflicting argument usage and for repeated usage of the same  
 118836 option is meant to prevent conforming applications from using conflicting arguments or  
 118837 repeated options unless specifically allowed (as is the case with *ls*, which allows simultaneous,  
 118838 repeated use of the *-C*, *-l*, and *-1* options). Many historical implementations will tolerate this  
 118839 usage, choosing either the first or the last applicable argument. This tolerance can continue, but  
 118840 conforming applications cannot rely upon it. (Other implementations may choose to print usage  
 118841 messages instead.)

118842 The use of “undefined” for conflicting argument usage also allows an implementation to make



reasonable extensions to utilities where the implementor considers mutually-exclusive options according to POSIX.1-2008 to have a sensible meaning and result.

POSIX.1-2008 does not define the result of a command when an option-argument or operand is not followed by ellipses and the application specifies more than one of that option-argument or operand. This allows an implementation to define valid (although non-standard) behavior for the utility when more than one such option or operand is specified.

The requirements for option-arguments are summarized as follows:

|                                      | SYNOPSIS Shows:                            |                                        |
|--------------------------------------|--------------------------------------------|----------------------------------------|
|                                      | <code>-a arg</code>                        | <code>-c [arg]</code>                  |
| Conforming application uses:         | <code>-a arg</code>                        | <code>-carg</code> or <code>-c</code>  |
| System supports:                     | <code>-a arg</code> and <code>-aarg</code> | <code>-carg</code> and <code>-c</code> |
| Non-conforming applications may use: | <code>-aarg</code>                         | N/A                                    |

Earlier versions of this standard included obsolescent syntax which showed some options with (mandatory) adjacent option-arguments in the SYNOPSIS for some utilities. These have since been removed. For all options with mandatory option-arguments, the SYNOPSIS now shows <blank> characters between the option and the option-argument; however, historical usage has not been consistent in this area; therefore, <blank> characters are required to be used by conforming applications and to be handled by all implementations, but implementations are also required to handle an adjacent option-argument in order to preserve backwards-compatibility for old scripts. One of the justifications for selecting the multiple-argument method was that the single-argument case is inherently ambiguous when the option-argument can legitimately be a null string.

POSIX.1-2008 explicitly states that digits are permitted as operands and option-arguments. The lower and upper bounds for the values of the numbers used for operands and option-arguments were derived from the ISO C standard values for {LONG\_MIN} and {LONG\_MAX}. The requirement on the standard utilities is that numbers in the specified range do not cause a syntax error, although the specification of a number need not be semantically correct for a particular operand or option-argument of a utility. For example, the specification of:

```
dd obs=3000000000
```

would yield undefined behavior for the application and could be a syntax error because the number 3 000 000 000 is outside of the range -2 147 483 647 to +2 147 483 647. On the other hand:

```
dd obs=2000000000
```

may cause some error, such as “blocksize too large”, rather than a syntax error.

**A.12.2 Utility Syntax Guidelines**

This section is based on the rules listed in the SVID. It was included for two reasons:

1. The individual utility descriptions in XCU Chapter 4 (on page 2423) needed a set of common (although not universal) actions on which they could anchor their descriptions of option and operand syntax. Most of the standard utilities actually do use these guidelines, and many of their historical implementations use the `getopt()` function for their parsing. Therefore, it was simpler to cite the rules and merely identify exceptions.
2. Developers of conforming applications need suggested guidelines if the POSIX community is to avoid the chaos of historical UNIX system command syntax.

It is recommended that all *future* utilities and applications use these guidelines to enhance “user

portability". The fact that some historical utilities could not be changed (to avoid breaking historical applications) should not deter this future goal.

The voluntary nature of the guidelines is highlighted by repeated uses of the word *should* throughout. This usage should not be misinterpreted to imply that utilities that claim conformance in their OPTIONS sections do not always conform.

Guidelines 1 and 2 encourage utility writers to use only characters from the portable character set because use of locale-specific characters may make the utility inaccessible from other locales. Use of uppercase letters is discouraged due to problems associated with porting utilities to systems that do not distinguish between uppercase and lowercase characters in filenames. Use of non-alphanumeric characters is discouraged due to the number of utilities that treat non-alphanumeric characters in "special" ways depending on context (such as the shell using white-space characters to delimit arguments, various quote characters for quoting, the <dollar-sign> to introduce variable expansion, etc.).

In XCU [Section 2.9.1](#) (on page 2338), it is further stated that a command used in the Shell Command Language cannot be named with a trailing <colon>.

Guideline 3 was changed to allow alphanumeric characters (letters and digits) from the character set to allow compatibility with historical usage. Historical practice allows the use of digits wherever practical, and there are no portability issues that would prohibit the use of digits. In fact, from an internationalization viewpoint, digits (being non-language-dependent) are preferable over letters (a -2 is intuitively self-explanatory to any user, while in the -f *filename* the letter 'f' is a mnemonic aid only to speakers of Latin-based languages where "filename" happens to translate to a word that begins with 'f'). Since Guideline 3 still retains the word "single", multi-digit options are not allowed. Instances of historical utilities that used them have been marked obsolescent, with the numbers being changed from option names to option-arguments.

It was difficult to achieve a satisfactory solution to the problem of name space in option characters. When the standard developers desired to extend the historical *cc* utility to accept ISO C standard programs, they found that all of the portable alphabet was already in use by various vendors. Thus, they had to devise a new name, *c89* (now superseded by *c99*), rather than something like *cc -X*. There were suggestions that implementors be restricted to providing extensions through various means (such as using a <plus-sign> as the option delimiter or using option characters outside the alphanumeric set) that would reserve all of the remaining alphanumeric characters for future POSIX standards. These approaches were resisted because they lacked the historical style of UNIX systems. Furthermore, if a vendor-provided option should become commonly used in the industry, it would be a candidate for standardization. It would be desirable to standardize such a feature using historical practice for the syntax (the semantics can be standardized with any syntax). This would not be possible if the syntax was one reserved for the vendor. However, since the standardization process may lead to minor changes in the semantics, it may prove to be better for a vendor to use a syntax that will not be affected by standardization.

Guideline 8 includes the concept of <comma>-separated lists in a single argument. It is up to the utility to parse such a list itself because *getopt()* just returns the single string. This situation was retained so that certain historical utilities would not violate the guidelines. Applications preparing for international use should be aware of an occasional problem with <comma>-separated lists: in some locales, the <comma> is used as the radix character. Thus, if an application is preparing operands for a utility that expects a <comma>-separated list, it should avoid generating non-integer values through one of the means that is influenced by setting the *LC\_NUMERIC* variable (such as *awk*, *bc*, *printf*, or *printf()*).

Unless explicitly stated otherwise in the utility description, Guideline 9 requires applications to

put options before operands, and requires utilities to accept any such usage without misinterpreting operands as options. For example, if an implementation of the *printf* utility supports a *-e* option as an extension, the command:

```
printf %s -e
```

must output the string "-e" without interpreting the *-e* as an option. Similarly, the command:

```
ls myfile -l
```

must interpret the *-l* argument as a second file operand, not as a *-l* option.

Applications calling any utility with a first operand starting with '-' should usually specify *--*, as indicated by Guideline 10, to mark the end of the options. This is true even if the SYNOPSIS in the Shell and Utilities volume of POSIX.1-2008 does not specify any options; implementations may provide options as extensions to the Shell and Utilities volume of POSIX.1-2008. The standard utilities that do not support Guideline 10 indicate that fact in the OPTIONS section of the utility description.

Guideline 7 allows any string to be an option-argument; an option-argument can begin with any character, can be - or --, and can be an empty string. For example, the commands *pr -h -*, *pr -h --*, *pr -h -d*, *pr -h +2*, and *pr -h ''* contain the option-arguments -, --, -d, +2, and an empty string, respectively. Conversely, the command *pr -h -- -d* treats -d as an option, not as an argument, because the -- is an option-argument here, not a delimiter.

Guideline 11 was modified to clarify that the order of different options should not matter relative to one another. However, the order of repeated options that also have option-arguments may be significant; therefore, such options are required to be interpreted in the order that they are specified. The *make* utility is an instance of a historical utility that uses repeated options in which the order is significant. Multiple files are specified by giving multiple instances of the *-f* option; for example:

```
make -f common_header -f specific_rules target
```

Guideline 13 does not imply that all of the standard utilities automatically accept the operand '-' to mean standard input or output, nor does it specify the actions of the utility upon encountering multiple '-' operands. It simply says that, by default, '-' operands are not used for other purposes in the file reading or writing (but not when using *stat()*, *unlink()*, *touch*, and so on) utilities. In earlier versions of this standard, all information concerning actual treatment of the '-' operand is found in the individual utility sections. Many implementations, however, treated '-' as standard input or output and many applications depended on this behavior even though it was not standard. This behavior is now implementation-defined. Portable applications should not use '-' to mean standard input or output unless it is explicitly stated to do so in the utility description and they should always use './-' if they intend to refer to a file named - in the current working directory.

Guideline 14 is intended to prohibit implementations that would treat the command *ls -l -d* as if it were *ls -- -l -d* or *ls -l -- -d*.

The standard permits implementations to have extensions that violate the Utility Syntax Guidelines so long as when the utility is used in line with the forms defined by the standard it follows the Utility Syntax Guidelines. Thus, *head-42file* and *ls--help* are permitted extensions. The intent is to allow extensions so long as the standard form is accepted and follows the guidelines.

An area of concern was that as implementations mature, implementation-defined utilities and implementation-defined utility options will result. The idea was expressed that there needed to be a standard way, say an environment variable or some such mechanism, to identify implementation-defined utilities separately from standard utilities that may have the same

118982 name. It was decided that there already exist several ways of dealing with this situation and that  
 118983 it is outside of the scope to attempt to standardize in the area of non-standard items. A method  
 118984 that exists on some historical implementations is the use of the so-called **/local/bin** or  
 118985 **/usr/local/bin** directory to separate local or additional copies or versions of utilities. Another  
 118986 method that is also used is to isolate utilities into completely separate domains. Still another  
 118987 method to ensure that the desired utility is being used is to request the utility by its full  
 118988 pathname. There are many approaches to this situation; the examples given above serve to  
 118989 illustrate that there is more than one.

## 118990 **A.13 Headers**

### 118991 **A.13.1 Format of Entries**

118992 Each header reference page has a common layout of sections describing the interface. This  
 118993 layout is similar to the manual page or “man” page format shipped with most UNIX systems,  
 118994 and each header has sections describing the SYNOPSIS and DESCRIPTION. These are the two  
 118995 sections that relate to conformance.

118996 Additional sections are informative, and add considerable information for the application  
 118997 developer. APPLICATION USAGE sections provide additional caveats, issues, and  
 118998 recommendations to the developer. RATIONALE sections give additional information on the  
 118999 decisions made in defining the interface.

119000 FUTURE DIRECTIONS sections act as pointers to related work that may impact the interface in  
 119001 the future, and often cautions the developer to architect the code to account for a change in this  
 119002 area. Note that a future directions statement should not be taken as a commitment to adopt a  
 119003 feature or interface in the future.

119004 The CHANGE HISTORY section describes when the interface was introduced, and how it has  
 119005 changed.

119006 Option labels and margin markings in the page can be useful in guiding the application  
 119007 developer.

### 119008 **A.13.2 Removed Headers in Issue 7**

119009 The headers removed in Issue 7 (from the Issue 6 base document) are as follows:

119010  
119011

| Removed Headers in Issue 7 |              |
|----------------------------|--------------|
| <sys/timeb.h>              | <ucontext.h> |

119012

# *Rationale (Informative)*

119013

## **Part B:**

119014

## **System Interfaces**

119015

*The Open Group*

119016

*The Institute of Electrical and Electronics Engineers, Inc.*



# *Rationale for System Interfaces*

## 119019 **B.1 Introduction**

### 119020 **B.1.1 Change History**

119021 The change history is provided as an informative section, to track changes from earlier versions  
119022 of this standard.

119023 The following sections describe changes made to the System Interfaces volume of POSIX.1-2008  
119024 since Issue 6 of the base document. The CHANGE HISTORY section for each entry details the  
119025 technical changes that have been made to that entry from Issue 5. Changes between earlier  
119026 versions of the base document and Issue 5 are not included.

#### 119027 **Changes from Issue 6 to Issue 7 (POSIX.1-2008)**

119028 The following list summarizes the major changes that were made in the System Interfaces  
119029 volume of POSIX.1-2008 from Issue 6 to Issue 7:

- 119030 • The Open Group Technical Standard, 2006, Extended API Set Part 1 is incorporated.
- 119031 • The Open Group Technical Standard, 2006, Extended API Set Part 2 is incorporated.
- 119032 • The Open Group Technical Standard, 2006, Extended API Set Part 3 is incorporated.
- 119033 • The Open Group Technical Standard, 2006, Extended API Set Part 4 is incorporated.
- 119034 • Existing functionality is aligned with ISO/IEC 9899:1999, Programming Languages — C,  
119035 ISO/IEC 9899:1999/Cor.2:2004(E)
- 119036 • Austin Group defect reports, IEEE Interpretations against IEEE Std 1003.1, and responses  
119037 to ISO/IEC defect reports against ISO/IEC 9945 are applied.
- 119038 • The Open Group corrigenda and resolutions are applied.
- 119039 • Features, marked legacy or obsolescent in the base document, have been considered for  
119040 removal in this version.
- 119041 • The options within the standard have been revised.

**New Features in Issue 7**

The functions first introduced in Issue 7 (over the Issue 6 base document) are as follows:

**New Functions in Issue 7**

|                      |                                      |                       |
|----------------------|--------------------------------------|-----------------------|
| <i>alphasort()</i>   | <i>iswdigit_l()</i>                  | <i>strfmon_l()</i>    |
| <i>dirfd()</i>       | <i>iswgraph_l()</i>                  | <i>strncascmp_l()</i> |
| <i>dprintf()</i>     | <i>iswlower_l()</i>                  | <i>strndup()</i>      |
| <i>duplocale()</i>   | <i>iswprint_l()</i>                  | <i>strnlen()</i>      |
| <i>faccessat()</i>   | <i>iswpunct_l()</i>                  | <i>strsignal()</i>    |
| <i>fchmodat()</i>    | <i>iswspace_l()</i>                  | <i>strxfrm_l()</i>    |
| <i>fchownat()</i>    | <i>iswupper_l()</i>                  | <i>symlinkat()</i>    |
| <i>fdopendir()</i>   | <i>iswxdigit_l()</i>                 | <i>tolower_l()</i>    |
| <i>fexecve()</i>     | <i>isxdigit_l()</i>                  | <i>toupper_l()</i>    |
| <i>fnmopen()</i>     | <i>linkat()</i>                      | <i>towctrans_l()</i>  |
| <i>freelocale()</i>  | <i>mbsnrtowcs()</i>                  | <i>tolower_l()</i>    |
| <i>fstatat()</i>     | <i>mkdirat()</i>                     | <i>towupper_l()</i>   |
| <i>futimens()</i>    | <i>mkdtemp()</i>                     | <i>unlinkat()</i>     |
| <i>getdelim()</i>    | <i>mkffloat()</i>                    | <i>uselocale()</i>    |
| <i>getline()</i>     | <i>mknodat()</i>                     | <i>utimensat()</i>    |
| <i>isalnum_l()</i>   | <i>newlocale()</i>                   | <i>vdprintf()</i>     |
| <i>isalpha_l()</i>   | <i>openat()</i>                      | <i>wcpcpy()</i>       |
| <i>isblank_l()</i>   | <i>open_memstream()</i>              | <i>wcpncpy()</i>      |
| <i>iscntrl_l()</i>   | <i>open_wmemstream()</i>             | <i>wscascmp_l()</i>   |
| <i>isdigit_l()</i>   | <i>psiginfo()</i>                    | <i>wscascmp_l()</i>   |
| <i>isgraph_l()</i>   | <i>psignal()</i>                     | <i>wscoll_l()</i>     |
| <i>islower_l()</i>   | <i>pthread_mutexattr_getrobust()</i> | <i>wcsdup()</i>       |
| <i>isprint_l()</i>   | <i>pthread_mutexattr_setrobust()</i> | <i>wcsncascmp_l()</i> |
| <i>ispunct_l()</i>   | <i>pthread_mutex_consistent()</i>    | <i>wcsncascmp_l()</i> |
| <i>isspace_l()</i>   | <i>readlinkat()</i>                  | <i>wcsnlen()</i>      |
| <i>isupper_l()</i>   | <i>renameat()</i>                    | <i>wcsnrtombs()</i>   |
| <i>iswalnum_l()</i>  | <i>scandir()</i>                     | <i>wcsxfrm_l()</i>    |
| <i>iswalphal_l()</i> | <i>stpcpy()</i>                      | <i>wctrans_l()</i>    |
| <i>iswblank_l()</i>  | <i>stpncpy()</i>                     | <i>wctype_l()</i>     |
| <i>iswcntrl_l()</i>  | <i>strcascmp_l()</i>                 |                       |
| <i>iswctype_l()</i>  | <i>strcoll_l()</i>                   |                       |

**Newly Mandated Functions in Issue 7**

The functions that were previously part of an option group but are now mandatory in Issue 7 are as follows:



119079

**Newly Mandated Functions in Issue 7**

|        |                                         |                                      |                                     |
|--------|-----------------------------------------|--------------------------------------|-------------------------------------|
| 119080 | <i>aio_cancel()</i>                     | <i>pthread_atfork()</i>              | <i>pthread_rwlock_tryrdlock()</i>   |
| 119081 | <i>aio_error()</i>                      | <i>pthread_attr_destroy()</i>        | <i>pthread_rwlock_trywrlock()</i>   |
| 119082 | <i>aio_fsync()</i>                      | <i>pthread_attr_getdetachstate()</i> | <i>pthread_rwlock_unlock()</i>      |
| 119083 | <i>aio_read()</i>                       | <i>pthread_attr_getguardsize()</i>   | <i>pthread_rwlock_wrlock()</i>      |
| 119084 | <i>aio_return()</i>                     | <i>pthread_attr_getschedparam()</i>  | <i>pthread_rwlockattr_destroy()</i> |
| 119085 | <i>aio_suspend()</i>                    | <i>pthread_attr_init()</i>           | <i>pthread_rwlockattr_init()</i>    |
| 119086 | <i>aio_write()</i>                      | <i>pthread_attr_setdetachstate()</i> | <i>pthread_self()</i>               |
| 119087 | <i>asctime_r()</i>                      | <i>pthread_attr_setguardsize()</i>   | <i>pthread_setcancelstate()</i>     |
| 119088 | <i>catclose()</i>                       | <i>pthread_attr_setschedparam()</i>  | <i>pthread_setcanceltype()</i>      |
| 119089 | <i>catgets()</i>                        | <i>pthread_barrier_destroy()</i>     | <i>pthread_setspecific()</i>        |
| 119090 | <i>catopen()</i>                        | <i>pthread_barrier_init()</i>        | <i>pthread_spin_destroy()</i>       |
| 119091 | <i>clock_getres()</i>                   | <i>pthread_barrier_wait()</i>        | <i>pthread_spin_init()</i>          |
| 119092 | <i>clock_gettime()</i>                  | <i>pthread_barrierattr_destroy()</i> | <i>pthread_spin_lock()</i>          |
| 119093 | <i>clock_nanosleep()</i>                | <i>pthread_barrierattr_init()</i>    | <i>pthread_spin_trylock()</i>       |
| 119094 | <i>clock_settime()</i>                  | <i>pthread_cancel()</i>              | <i>pthread_spin_unlock()</i>        |
| 119095 | <i>ctime_r()</i>                        | <i>pthread_cleanup_pop()</i>         | <i>pthread_testcancel()</i>         |
| 119096 | <i>dlclose()</i>                        | <i>pthread_cleanup_push()</i>        | <i>putc_unlocked()</i>              |
| 119097 | <i>dlderror()</i>                       | <i>pthread_cond_broadcast()</i>      | <i>putchar_unlocked()</i>           |
| 119098 | <i>dlopen()</i>                         | <i>pthread_cond_destroy()</i>        | <i>pwrite()</i>                     |
| 119099 | <i>dlsym()</i>                          | <i>pthread_cond_init()</i>           | <i>rand_r()</i>                     |
| 119100 | <i>fchdir()</i>                         | <i>pthread_cond_signal()</i>         | <i>readdir_r()</i>                  |
| 119101 | <i>flockfile()</i>                      | <i>pthread_cond_timedwait()</i>      | <i>sem_close()</i>                  |
| 119102 | <i>fstatvfs()</i>                       | <i>pthread_cond_wait()</i>           | <i>sem_destroy()</i>                |
| 119103 | <i>ftrylockfile()</i>                   | <i>pthread_condattr_destroy()</i>    | <i>sem_getvalue()</i>               |
| 119104 | <i>funlockfile()</i>                    | <i>pthread_condattr_getclock()</i>   | <i>sem_init()</i>                   |
| 119105 | <i>getc_unlocked()</i>                  | <i>pthread_condattr_init()</i>       | <i>sem_open()</i>                   |
| 119106 | <i>getchar_unlocked()</i>               | <i>pthread_condattr_setclock()</i>   | <i>sem_post()</i>                   |
| 119107 | <i>getgrgid_r()</i>                     | <i>pthread_create()</i>              | <i>sem_timedwait()</i>              |
| 119108 | <i>getgrnam_r()</i>                     | <i>pthread_detach()</i>              | <i>sem_trywait()</i>                |
| 119109 | <i>getlogin_r()</i>                     | <i>pthread_equal()</i>               | <i>sem_unlink()</i>                 |
| 119110 | <i>getpgid()</i>                        | <i>pthread_exit()</i>                | <i>sem_wait()</i>                   |
| 119111 | <i>getpwnam_r()</i>                     | <i>pthread_getspecific()</i>         | <i>sigqueue()</i>                   |
| 119112 | <i>getpwuid_r()</i>                     | <i>pthread_join()</i>                | <i>sigqueue()</i>                   |
| 119113 | <i>getsid()</i>                         | <i>pthread_key_create()</i>          | <i>sigtimedwait()</i>               |
| 119114 | <i>getsubopt()</i>                      | <i>pthread_key_delete()</i>          | <i>sigwaitinfo()</i>                |
| 119115 | <i>gmtime_r()</i>                       | <i>pthread_mutex_destroy()</i>       | <i>statvfs()</i>                    |
| 119116 | <i>iconv()</i>                          | <i>pthread_mutex_init()</i>          | <i>strcasecmp()</i>                 |
| 119117 | <i>iconv_close()</i>                    | <i>pthread_mutex_lock()</i>          | <i>strdup()</i>                     |
| 119118 | <i>iconv_open()</i>                     | <i>pthread_mutex_timedlock()</i>     | <i>strerror_r()</i>                 |
| 119119 | <i>lchown()</i>                         | <i>pthread_mutex_trylock()</i>       | <i>strfmon()</i>                    |
| 119120 | <i>lio_listio()</i>                     | <i>pthread_mutex_unlock()</i>        | <i>strncasecmp()</i>                |
| 119121 | <i>localtime_r()</i>                    | <i>pthread_mutexattr_destroy()</i>   | <i>strtok_r()</i>                   |
| 119122 | <i>mkstemp()</i>                        | <i>pthread_mutexattr_gettype()</i>   | <i>tcgetsid()</i>                   |
| 119123 | <i>mmap()</i>                           | <i>pthread_mutexattr_init()</i>      | <i>timer_create()</i>               |
| 119124 | <i>mprotect()</i>                       | <i>pthread_mutexattr_settype()</i>   | <i>timer_delete()</i>               |
| 119125 | <i>munmap()</i>                         | <i>pthread_once()</i>                | <i>timer_getoverrun()</i>           |
| 119126 | <i>nanosleep()</i>                      | <i>pthread_rwlock_destroy()</i>      | <i>timer_gettime()</i>              |
| 119127 | <i>nl_langinfo()</i>                    | <i>pthread_rwlock_init()</i>         | <i>timer_settime()</i>              |
| 119128 | <i>poll()</i>                           | <i>pthread_rwlock_rdlock()</i>       | <i>truncate()</i>                   |
| 119129 | <i>posix_trace_timedgetnext_event()</i> | <i>pthread_rwlock_timedrdlock()</i>  | <i>ttyname_r()</i>                  |
| 119130 | <i>pread()</i>                          | <i>pthread_rwlock_timedwrlock()</i>  | <i>waitid()</i>                     |

**Obsolescent Functions in Issue 7**

The base functions moved to obsolescent status in Issue 7 (from the Issue 6 base document) are as follows:

**Obsolescent Base Functions in Issue 7**

|                    |                 |
|--------------------|-----------------|
| <i>asctime()</i>   | <i>gets()</i>   |
| <i>asctime_r()</i> | <i>rand_r()</i> |
| <i>ctime()</i>     | <i>tmpnam()</i> |
| <i>ctime_r()</i>   | <i>utime()</i>  |

The XSI functions moved to obsolescent status in Issue 7 (from the Issue 6 base document) are as follows:

**Obsolescent XSI Functions in Issue 7**

|                       |                                 |                       |
|-----------------------|---------------------------------|-----------------------|
| <i>_longjmp()</i>     | <i>pthread_getconcurrency()</i> | <i>sigset()</i>       |
| <i>_setjmp()</i>      | <i>pthread_setconcurrency()</i> | <i>siginterrupt()</i> |
| <i>_tolower()</i>     | <i>setitimer()</i>              | <i>tempnam()</i>      |
| <i>_toupper()</i>     | <i>setpggrp()</i>               | <i>toascii()</i>      |
| <i>ftw()</i>          | <i>sighold()</i>                | <i>ulimit()</i>       |
| <i>getitimer()</i>    | <i>sigignore()</i>              |                       |
| <i>gettimeofday()</i> | <i>sigpause()</i>               |                       |
| <i>isascii()</i>      | <i>sigrelse()</i>               |                       |

**Removed Functions and Symbols in Issue 7**

The functions and symbols removed in Issue 7 (from the Issue 6 base document) are as follows:

**Removed Functions and Symbols in Issue 7**

|                     |                                    |                      |
|---------------------|------------------------------------|----------------------|
| <i>bcmp()</i>       | <i>gethostbyaddr()</i>             | <i>rindex()</i>      |
| <i>bcopy()</i>      | <i>gethostbyname()</i>             | <i>scalb()</i>       |
| <i>bsd_signal()</i> | <i>getwd()</i>                     | <i>setcontext()</i>  |
| <i>bzero()</i>      | <i>h_errno</i>                     | <i>swapcontext()</i> |
| <i>ecvt()</i>       | <i>index()</i>                     | <i>ualarm()</i>      |
| <i>fcvt()</i>       | <i>makecontext()</i>               | <i>usleep()</i>      |
| <i>ftime()</i>      | <i>mktemp()</i>                    | <i>vfork()</i>       |
| <i>gcvt()</i>       | <i>pthread_attr_getstackaddr()</i> | <i>wcswcs()</i>      |
| <i>getcontext()</i> | <i>pthread_attr_setstackaddr()</i> |                      |

**B.1.2 Relationship to Other Formal Standards**

There is no additional rationale provided for this section.

**B.1.3 Format of Entries**

Each system interface reference page has a common layout of sections describing the interface. This layout is similar to the manual page or “man” page format shipped with most UNIX systems, and each header has sections describing the SYNOPSIS, DESCRIPTION, RETURN VALUE, and ERRORS. These are the four sections that relate to conformance.

Additional sections are informative, and add considerable information for the application developer. EXAMPLES sections provide example usage. APPLICATION USAGE sections

119171 provide additional caveats, issues, and recommendations to the developer. RATIONALE  
 119172 sections give additional information on the decisions made in defining the interface.

119173 FUTURE DIRECTIONS sections act as pointers to related work that may impact the interface in  
 119174 the future, and often cautions the developer to architect the code to account for a change in this  
 119175 area. Note that a future directions statement should not be taken as a commitment to adopt a  
 119176 feature or interface in the future.

119177 The CHANGE HISTORY section describes when the interface was introduced, and how it has  
 119178 changed.

119179 Option labels and margin markings in the page can be useful in guiding the application  
 119180 developer.

## 119181 B.2 General Information

### 119182 B.2.1 Use and Implementation of Interfaces

119183 The information concerning the use of functions was adapted from a description in the ISO C  
 119184 standard. Here is an example of how an application program can protect itself from functions  
 119185 that may or may not be macros, rather than true functions:

119186 The *atoi()* function may be used in any of several ways:

- 119187 • By use of its associated header (possibly generating a macro expansion):

```
119188 #include <stdlib.h>
119189 /* ... */
119190 i = atoi(str);
```

- 119191 • By use of its associated header (assuredly generating a true function call):

```
119192 #include <stdlib.h>
119193 #undef atoi
119194 /* ... */
119195 i = atoi(str);
```

119196 or:

```
119197 #include <stdlib.h>
119198 /* ... */
119199 i = (atoi) (str);
```

- 119200 • By explicit declaration:

```
119201 extern int atoi (const char *);
119202 /* ... */
119203 i = atoi(str);
```

- 119204 • By implicit declaration:

```
119205 /* ... */
119206 i = atoi(str);
```

119207 (Assuming no function prototype is in scope. This is not allowed by the ISO C standard for  
 119208 functions with variable arguments; furthermore, parameter type conversion “widening” is

119209 subject to different rules in this case.)

119210 Note that the ISO C standard reserves names starting with ‘\_’ for the compiler. Therefore, the  
 119211 compiler could, for example, implement an intrinsic, built-in function `_asm_builtin_atoi()`, which  
 119212 it recognized and expanded into inline assembly code. Then, in `<stdlib.h>`, there could be the  
 119213 following:

```
119214 #define atoi(X) _asm_builtin_atoi(X)
```

119215 The user’s “normal” call to `atoi()` would then be expanded inline, but the implementor would  
 119216 also be required to provide a callable function named `atoi()` for use when the application  
 119217 requires it; for example, if its address is to be stored in a function pointer variable.

119218 Implementors should note that since applications can **#undef** a macro in order to ensure that the  
 119219 function is used, this means that it is not safe for implementations to use the names of any  
 119220 standard functions in macro values, since the application could use **#undef** to ensure that no  
 119221 macro exists and then use the same name for an identifier with local scope. For example,  
 119222 historically it was common for a `getchar()` macro to be defined in `<stdio.h>` as:

```
119223 #define getchar() getc(stdin)
```

119224 This definition does not conform, because an application is allowed to use the identifier `getc`  
 119225 with local scope, and the expansion of the `getchar()` macro would then pick up the local `getc`.  
 119226 The following is conforming code, but would not compile with the above definition of `getchar()`:

```
119227 #include <stdio.h>
```

```
119228 #undef getc
```

```
119229 int main(void)
```

```
119230 {
```

```
119231 int getc;
```

```
119232 getc = getchar();
```

```
119233 return getc;
```

```
119234 }
```

119235 This does not only affect function-like macros. For example, the following definition does not  
 119236 conform because there could be a local `sysconf` variable in scope when `SIGRTMIN` is expanded:

```
119237 #define SIGRTMIN ((int)sysconf(_SC_SIGRT_MIN))
```

119238 Implementors can avoid the problem by using aliases for standard functions instead of the  
 119239 actual function, with names that conforming applications cannot use for local variables. For  
 119240 example:

```
119241 #define SIGRTMIN ((int)__sysconf(_SC_SIGRT_MIN))
```

## 119242 B.2.2 The Compilation Environment

### 119243 B.2.2.1 POSIX.1 Symbols

119244 This and the following section address the issue of “name space pollution”. The ISO C standard  
 119245 requires that the name space beyond what it reserves not be altered except by explicit action of  
 119246 the application developer. This section defines the actions to add the POSIX.1 symbols for those  
 119247 headers where both the ISO C standard and POSIX.1 need to define symbols, and also where the

119248 XSI option extends the base standard.

119249 When headers are used to provide symbols, there is a potential for introducing symbols that the  
 119250 application developer cannot predict. Ideally, each header should only contain one set of  
 119251 symbols, but this is not practical for historical reasons. Thus, the concept of feature test macros is  
 119252 included. Two feature test macros are explicitly defined by POSIX.1-2008; it is expected that  
 119253 future versions may add to this.

119254 **Note:** Feature test macros allow an application to announce to the implementation its desire to have  
 119255 certain symbols and prototypes exposed. They should not be confused with the version test  
 119256 macros and constants for options in `<unistd.h>` which are the implementation's way of  
 119257 announcing functionality to the application.

119258 It is further intended that these feature test macros apply only to the headers specified by  
 119259 POSIX.1-2008. Implementations are expressly permitted to make visible symbols not specified  
 119260 by POSIX.1-2008, within both POSIX.1 and other headers, under the control of feature test  
 119261 macros that are not defined by POSIX.1-2008.

#### 119262 **The `_POSIX_C_SOURCE` Feature Test Macro**

119263 The POSIX.1-1990 standard specified a macro called `_POSIX_SOURCE`. This has been  
 119264 superseded by `_POSIX_C_SOURCE`. This symbol will allow implementations to support various  
 119265 versions of this standard simultaneously. For instance, when `_POSIX_C_SOURCE` is defined as  
 119266 200809L, the system should make visible the same name space as permitted and required by the  
 119267 POSIX.1-2008 standard. A special case is the one where the implementation wishes to make  
 119268 available support for the 1990 version of the POSIX standard, in which instance when either  
 119269 `_POSIX_SOURCE` is defined or `_POSIX_C_SOURCE` is defined as 1, the system should make  
 119270 visible the same name space as permitted and required by the POSIX.1-1990 standard.

119271 It is expected that C bindings to future POSIX standards will define new values for  
 119272 `_POSIX_C_SOURCE`, with each new value reserving the name space for that new standard.

#### 119273 **The `_XOPEN_SOURCE` Feature Test Macro**

119274 The feature test macro `_XOPEN_SOURCE` is provided as the announcement mechanism for the  
 119275 application that it requires functionality from the Single UNIX Specification. `_XOPEN_SOURCE`  
 119276 must be defined to the value 700 before the inclusion of any header to enable the functionality in  
 119277 the Single UNIX Specification Version 4. Its definition subsumes the use of `_POSIX_C_SOURCE`.

119278 An extract of code from a conforming application, that appears before any **#include** statements,  
 119279 is given below:

```
119280 #define _XOPEN_SOURCE 700 /* Single UNIX Specification, Version 4 */
119281 #include ...
```

119282 Note that the definition of `_XOPEN_SOURCE` with the value 700 makes the definition of  
 119283 `_POSIX_C_SOURCE` redundant and it can safely be omitted.

#### 119284 **B.2.2.2 The Name Space**

119285 The reservation of identifiers is paraphrased from the ISO C standard. The text is included  
 119286 because it needs to be part of POSIX.1-2008, regardless of possible changes in future versions of  
 119287 the ISO C standard.

119288 These identifiers may be used by implementations, particularly for feature test macros.  
 119289 Implementations should not use feature test macro names that might be reasonably used by a  
 119290 standard.

Including headers more than once is a reasonably common practice, and it should be carried forward from the ISO C standard. More significantly, having definitions in more than one header is explicitly permitted. Where the potential declaration is “benign” (the same definition twice) the declaration can be repeated, if that is permitted by the compiler. (This is usually true of macros, for example.) In those situations where a repetition is not benign (for example, **typedefs**), conditional compilation must be used. The situation actually occurs both within the ISO C standard and within POSIX.1: **time\_t** should be in **<sys/types.h>**, and the ISO C standard mandates that it be in **<time.h>**.

The area of name space pollution *versus* additions to structures is difficult because of the macro structure of C. The following discussion summarizes all the various problems with and objections to the issue.

Note the phrase “user-defined macro”. Users are not permitted to define macro names (or any other name) beginning with “\_**[A-Z\_]**”. Thus, the conflict cannot occur for symbols reserved to the vendor’s name space, and the permission to add fields automatically applies, without qualification, to those symbols.

1. Data structures (and unions) need to be defined in headers by implementations to meet certain requirements of POSIX.1 and the ISO C standard.

2. The structures defined by POSIX.1 are typically minimal, and any practical implementation would wish to add fields to these structures either to hold additional related information or for backwards-compatibility (or both). Future standards (and *de facto* standards) would also wish to add to these structures. Issues of field alignment make it impractical (at least in the general case) to simply omit fields when they are not defined by the particular standard involved.

The **dirent** structure is an example of such a minimal structure (although one could argue about whether the other fields need visible names). The **st\_rdev** field of most implementations’ **stat** structure is a common example where extension is needed and where a conflict could occur.

3. Fields in structures are in an independent name space, so the addition of such fields presents no problem to the C language itself in that such names cannot interact with identically named user symbols because access is qualified by the specific structure name.

4. There is an exception to this: macro processing is done at a lexical level. Thus, symbols added to a structure might be recognized as user-provided macro names at the location where the structure is declared. This only can occur if the user-provided name is declared as a macro before the header declaring the structure is included. The user’s use of the name after the declaration cannot interfere with the structure because the symbol is hidden and only accessible through access to the structure. Presumably, the user would not declare such a macro if there was an intention to use that field name.

5. Macros from the same or a related header might use the additional fields in the structure, and those field names might also collide with user macros. Although this is a less frequent occurrence, since macros are expanded at the point of use, no constraint on the order of use of names can apply.

6. An “obvious” solution of using names in the reserved name space and then redefining them as macros when they should be visible does not work because this has the effect of exporting the symbol into the general name space. For example, given a (hypothetical) system-provided header **<h.h>**, and two parts of a C program in **a.c** and **b.c**, in header **<h.h>**:

```
struct foo {
 int __i;
```

```

119339 }
119340 #ifdef _FEATURE_TEST
119341 #define i __i;
119342 #endif

```

119343 In file **a.c**:

```

119344 #include h.h
119345 extern int i;
119346 ...

```

119347 In file **b.c**:

```

119348 extern int i;
119349 ...

```

119350 The symbol that the user thinks of as *i* in both files has an external name of `__i` in **a.c**; the  
 119351 same symbol *i* in **b.c** has an external name *i* (ignoring any hidden manipulations the  
 119352 compiler might perform on the names). This would cause a mysterious name resolution  
 119353 problem when **a.o** and **b.o** are linked.

119354 Simply avoiding definition then causes alignment problems in the structure.

119355 A structure of the form:

```

119356 struct foo {
119357 union {
119358 int __i;
119359 #ifdef _FEATURE_TEST
119360 int i;
119361 #endif
119362 } __ii;
119363 }

```

119364 does not work because the name of the logical field *i* is `__ii.i`, and introduction of a macro  
 119365 to restore the logical name immediately reintroduces the problem discussed previously  
 119366 (although its manifestation might be more immediate because a syntax error would result  
 119367 if a recursive macro did not cause it to fail first).

119368 7. A more workable solution would be to declare the structure:

```

119369 struct foo {
119370 #ifdef _FEATURE_TEST
119371 int i;
119372 #else
119373 int __i;
119374 #endif
119375 }

```

119376 However, if a macro (particularly one required by a standard) is to be defined that uses  
 119377 this field, two must be defined: one that uses *i*, the other that uses `__i`. If more than one  
 119378 additional field is used in a macro and they are conditional on distinct combinations of  
 119379 features, the complexity goes up as  $2^n$ .

119380 All this leaves a difficult situation: vendors must provide very complex headers to deal with  
 119381 what is conceptually simple and safe—adding a field to a structure. It is the possibility of user-  
 119382 provided macros with the same name that makes this difficult.

119383 Several alternatives were proposed that involved constraining the user's access to part of the

name space available to the user (as specified by the ISO C standard). In some cases, this was only until all the headers had been included. There were two proposals discussed that failed to achieve consensus:

1. Limiting it for the whole program.
2. Restricting the use of identifiers containing only uppercase letters until after all system headers had been included. It was also pointed out that because macros might wish to access fields of a structure (and macro expansion occurs totally at point of use) restricting names in this way would not protect the macro expansion, and thus the solution was inadequate.

It was finally decided that reservation of symbols would occur, but as constrained.

The current wording also allows the addition of fields to a structure, but requires that user macros of the same name not interfere. This allows vendors to do one of the following:

- Not create the situation (do not extend the structures with user-accessible names or use the solution in (7) above)
- Extend their compilers to allow some way of adding names to structures and macros safely

There are at least two ways that the compiler might be extended: add new preprocessor directives that turn off and on macro expansion for certain symbols (without changing the value of the macro) and a function or lexical operation that suppresses expansion of a word. The latter seems more flexible, particularly because it addresses the problem in macros as well as in declarations.

The following seems to be a possible implementation extension to the C language that will do this: any token that during macro expansion is found to be preceded by three '#' symbols shall not be further expanded in exactly the same way as described for macros that expand to their own name as in Section 3.8.3.4 of the ISO C standard. A vendor may also wish to implement this as an operation that is lexically a function, which might be implemented as:

```
#define __safe_name(x) ###x
```

Using a function notation would insulate vendors from changes in standards until such a functionality is standardized (if ever). Standardization of such a function would be valuable because it would then permit third parties to take advantage of it portably in software they may supply.

The symbols that are “explicitly permitted, but not required by POSIX.1-2008” include those classified below. (That is, the symbols classified below might, but are not required to, be present when `_POSIX_C_SOURCE` is defined to have the value 200809L.)

- Symbols in `<limits.h>` and `<unistd.h>` that are defined to indicate support for options or limits that are constant at compile-time
- Symbols in the name space reserved for the implementation by the ISO C standard
- Symbols in a name space reserved for a particular type of extension (for example, type names ending with `_t` in `<sys/types.h>`)
- Additional members of structures or unions whose names do not reduce the name space reserved for applications

Since both implementations and future versions of this standard and other POSIX standards may use symbols in the reserved spaces described in these tables, there is a potential for name space clashes. To avoid future name space clashes when adding symbols, implementations should not use the `posix_`, `POSIX_`, or `_POSIX_` prefixes.



IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/2 is applied, deleting the entries `POSIX_`, `_POSIX_`, and `posix_` from the column of allowed name space prefixes for use by an implementation in the first table. The presence of these prefixes was contradicting later text which states that: “The prefixes `posix_`, `POSIX_`, and `_POSIX` are reserved for use by XCU Chapter 2 (on page 2319) and other POSIX standards. Implementations may add symbols to the headers shown in the following table, provided the identifiers ... do not use the reserved prefixes `posix_`, `POSIX_`, or `_POSIX`.”.

IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/3 is applied, correcting the reserved macro prefix from: “`PRI[a-z]`, `SCN[a-z]`” to: “`PRI[Xa-z]`, `SCN[Xa-z]`” in the second table. The change was needed since the ISO C standard allows implementations to define macros of the form `PRI` or `SCN` followed by any lowercase letter or ‘`x`’ in `<inttypes.h>`. (The ISO/IEC 9899:1999 standard, Subclause 7.26.4.)

IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/4 is applied, adding a new section listing reserved names for the `<stdint.h>` header. This change is for alignment with the ISO C standard.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/2 is applied, making it clear that implementations are permitted to have symbols with the prefix `_POSIX_` visible in any header.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/3 is applied, updating the table of allowed macro prefixes to include the prefix `FP_[A-Z]` for `<math.h>`. This text is added for consistency with the `<math.h>` reference page in the Base Definitions volume of POSIX.1-2008 which permits additional implementation-defined floating-point classifications.

Austin Group Interpretation 1003.1-2001 #048 is applied, reserving `SEEK_` in the name space.

### 119449 B.2.3 Error Numbers

It was the consensus of the standard developers that to allow the conformance document to state that an error occurs and under what conditions, but to disallow a statement that it never occurs, does not make sense. It could be implied by the current wording that this is allowed, but to reduce the possibility of future interpretation requests, it is better to make an explicit statement.

The ISO C standard requires that `errno` be an assignable lvalue. Originally, the definition in POSIX.1 was stricter than that in the ISO C standard, `extern int errno`, in order to support historical usage. In a multi-threaded environment, implementing `errno` as a global variable results in non-deterministic results when accessed. It is required, however, that `errno` work as a per-thread error reporting mechanism. In order to do this, a separate `errno` value has to be maintained for each thread. The following section discusses the various alternative solutions that were considered.

In order to avoid this problem altogether for new functions, these functions avoid using `errno` and, instead, return the error number directly as the function return value; a return value of zero indicates that no error was detected.

For any function that can return errors, the function return value is not used for any purpose other than for reporting errors. Even when the output of the function is scalar, it is passed through a function argument. While it might have been possible to allow some scalar outputs to be coded as negative function return values and mixed in with positive error status returns, this was rejected—using the return value for a mixed purpose was judged to be of limited use and error prone.

Checking the value of `errno` alone is not sufficient to determine the existence or type of an error, since it is not required that a successful function call clear `errno`. The variable `errno` should only be examined when the return value of a function indicates that the value of `errno` is meaningful. In that case, the function is required to set the variable to something other than zero.

- 119474 The variable *errno* is never set to zero by any function call; to do so would contradict the ISO C  
119475 standard.
- 119476 POSIX.1 requires (in the ERRORS sections of function descriptions) certain error values to be set  
119477 in certain conditions because many existing applications depend on them. Some error numbers,  
119478 such as [EFAULT], are entirely implementation-defined and are noted as such in their  
119479 description in the ERRORS section. This section otherwise allows wide latitude to the  
119480 implementation in handling error reporting.
- 119481 Some of the ERRORS sections in POSIX.1-2008 have two subsections. The first:  
119482 “The function shall fail if:”
- 119483 could be called the “mandatory” section.
- 119484 The second:  
119485 “The function may fail if:”
- 119486 could be informally known as the “optional” section.
- 119487 Attempting to infer the quality of an implementation based on whether it detects optional error  
119488 conditions is not useful.
- 119489 Following each one-word symbolic name for an error, there is a description of the error. The  
119490 rationale for some of the symbolic names follows:
- 119491 [ECANCELED] This spelling was chosen as being more common.
- 119492 [EFAULT] Most historical implementations do not catch an error and set *errno* when an  
119493 invalid address is given to the functions *wait()*, *time()*, or *times()*. Some  
119494 implementations cannot reliably detect an invalid address. And most systems  
119495 that detect invalid addresses will do so only for a system call, not for a library  
119496 routine.
- 119497 [EFTYPE] This error code was proposed in earlier proposals as “Inappropriate operation  
119498 for file type”, meaning that the operation requested is not appropriate for the  
119499 file specified in the function call. This code was proposed, although the same  
119500 idea was covered by [ENOTTY], because the connotations of the name would  
119501 be misleading. It was pointed out that the *fcntl()* function uses the error code  
119502 [EINVAL] for this notion, and hence all instances of [EFTYPE] were changed  
119503 to this code.
- 119504 [EINTR] POSIX.1 prohibits conforming implementations from restarting interrupted  
119505 system calls of conforming applications unless the SA\_RESTART flag is in  
119506 effect for the signal. However, it does not require that [EINTR] be returned  
119507 when another legitimate value may be substituted; for example, a partial  
119508 transfer count when *read()* or *write()* are interrupted. This is only given when  
119509 the signal-catching function returns normally as opposed to returns by  
119510 mechanisms like *longjmp()* or *siglongjmp()*.
- 119511 [ELOOP] In specifying conditions under which implementations would generate this  
119512 error, the following goals were considered:
- 119513 • To ensure that actual loops are detected, including loops that result from  
119514 symbolic links across distributed file systems.
  - 119515 • To ensure that during pathname resolution an application can rely on  
119516 the ability to follow at least {SYMLOOP\_MAX} symbolic links in the  
119517 absence of a loop.

|        |                |                                                                                                     |
|--------|----------------|-----------------------------------------------------------------------------------------------------|
| 119518 |                | • To allow implementations to provide the capability of traversing more                             |
| 119519 |                | than {SYMLOOP_MAX} symbolic links in the absence of a loop.                                         |
| 119520 |                | • To allow implementations to detect loops and generate the error prior to                          |
| 119521 |                | encountering {SYMLOOP_MAX} symbolic links.                                                          |
| 119522 | [ENAMETOOLONG] |                                                                                                     |
| 119523 |                | When a symbolic link is encountered during pathname resolution, the                                 |
| 119524 |                | contents of that symbolic link are used to create a new pathname. The                               |
| 119525 |                | standard developers intended to allow, but not require, that implementations                        |
| 119526 |                | enforce the restriction of {PATH_MAX} on the result of this pathname                                |
| 119527 |                | substitution.                                                                                       |
| 119528 |                | Implementations are allowed, but not required, to treat a pathname longer                           |
| 119529 |                | than {PATH_MAX} passed into the system as an error. Implementations are                             |
| 119530 |                | required to return a pathname (even if it is longer than {PATH_MAX}) when                           |
| 119531 |                | the user supplies a buffer with an interface that specifies the buffer size, as                     |
| 119532 |                | long as the user-supplied buffer is large enough to hold the entire pathname                        |
| 119533 |                | (see XSH <i>getcwd()</i> for an example of this type of interface). Implementations                 |
| 119534 |                | are required to treat a request to pass a pathname longer than {PATH_MAX}                           |
| 119535 |                | from the system to a user-supplied buffer of an unspecified size (usually                           |
| 119536 |                | assumed to be of size {PATH_MAX}) as an error (see XSH <i>realpath()</i> for an                     |
| 119537 |                | example of this type of interface).                                                                 |
| 119538 | [ENOMEM]       | The term “main memory” is not used in POSIX.1 because it is                                         |
| 119539 |                | implementation-defined.                                                                             |
| 119540 | [ENOTSUP]      | This error code is to be used when an implementation chooses to implement                           |
| 119541 |                | the required functionality of POSIX.1-2008 but does not support optional                            |
| 119542 |                | facilities defined by POSIX.1-2008. The return of [ENOSYS] is to be taken to                        |
| 119543 |                | indicate that the function of the interface is not supported at all; the function                   |
| 119544 |                | will always fail with this error code.                                                              |
| 119545 | [ENOTTY]       | The symbolic name for this error is derived from a time when device control                         |
| 119546 |                | was done by <i>ioctl()</i> and that operation was only permitted on a terminal                      |
| 119547 |                | interface. The term “TTY” is derived from “teletypewriter”, the devices to                          |
| 119548 |                | which this error originally applied.                                                                |
| 119549 | [EOVERFLOW]    | Most of the uses of this error code are related to large file support. Typically,                   |
| 119550 |                | these cases occur on systems which support multiple programming                                     |
| 119551 |                | environments with different sizes for <b>off_t</b> , but they may also occur in                     |
| 119552 |                | connection with remote file systems.                                                                |
| 119553 |                | In addition, when different programming environments have different widths                          |
| 119554 |                | for types such as <b>int</b> and <b>uid_t</b> , several functions may encounter a condition         |
| 119555 |                | where a value in a particular environment is too wide to be represented. In                         |
| 119556 |                | that case, this error should be raised. For example, suppose the currently                          |
| 119557 |                | running process has 64-bit <b>int</b> , and file descriptor 9 223 372 036 854 775 807 is            |
| 119558 |                | open and does not have the close-on-exec flag set. If the process then uses                         |
| 119559 |                | <i>execl()</i> to <i>exec</i> a file compiled in a programming environment with 32-bit <b>int</b> , |
| 119560 |                | the call to <i>execl()</i> can fail with <i>errno</i> set to [EOVERFLOW]. A similar failure         |
| 119561 |                | can occur with <i>execl()</i> if any of the user IDs or any of the group IDs to be                  |
| 119562 |                | assigned to the new process image are out of range for the executed file’s                          |
| 119563 |                | programming environment.                                                                            |
| 119564 |                | Note, however, that this condition cannot occur for functions that are                              |
| 119565 |                | explicitly described as always being successful, such as <i>getpid()</i> .                          |

119566 [EPIPE] This condition normally generates the signal SIGPIPE; the error is returned if  
 119567 the generation of the signal is suppressed or the signal does not terminate the  
 119568 process.

119569 [EROFS] In historical implementations, attempting to *unlink()* or *rmdir()* a mount point  
 119570 would generate an [EBUSY] error. An implementation could be envisioned  
 119571 where such an operation could be performed without error. In this case, if  
 119572 *either* the directory entry or the actual data structures reside on a read-only file  
 119573 system, [EROFS] is the appropriate error to generate. (For example, changing  
 119574 the link count of a file on a read-only file system could not be done, as is  
 119575 required by *unlink()*, and thus an error should be reported.)

119576 Three error numbers, [EDOM], [EILSEQ], and [ERANGE], were added to this section primarily  
 119577 for consistency with the ISO C standard.

#### 119578 **Alternative Solutions for Per-Thread *errno***

119579 The usual implementation of *errno* as a single global variable does not work in a multi-threaded  
 119580 environment. In such an environment, a thread may make a POSIX.1 call and get a -1 error  
 119581 return, but before that thread can check the value of *errno*, another thread might have made a  
 119582 second POSIX.1 call that also set *errno*. This behavior is unacceptable in robust programs. There  
 119583 were a number of alternatives that were considered for handling the *errno* problem:

- 119584 • Implement *errno* as a per-thread integer variable.
- 119585 • Implement *errno* as a service that can access the per-thread error number.
- 119586 • Change all POSIX.1 calls to accept an extra status argument and avoid setting *errno*.
- 119587 • Change all POSIX.1 calls to raise a language exception.

119588 The first option offers the highest level of compatibility with existing practice but requires  
 119589 special support in the linker, compiler, and/or virtual memory system to support the new  
 119590 concept of thread private variables. When compared with current practice, the third and fourth  
 119591 options are much cleaner, more efficient, and encourage a more robust programming style, but  
 119592 they require new versions of all of the POSIX.1 functions that might detect an error. The second  
 119593 option offers compatibility with existing code that uses the **<errno.h>** header to define the  
 119594 symbol *errno*. In this option, *errno* may be a macro defined:

```
119595 #define errno (*__errno())
119596 extern int *__errno();
```

119597 This option may be implemented as a per-thread variable whereby an *errno* field is allocated in  
 119598 the user space object representing a thread, and whereby the function *\_\_errno()* makes a system  
 119599 call to determine the location of its user space object and returns the address of the *errno* field of  
 119600 that object. Another implementation, one that avoids calling the kernel, involves allocating  
 119601 stacks in chunks. The stack allocator keeps a side table indexed by chunk number containing a  
 119602 pointer to the thread object that uses that chunk. The *\_\_errno()* function then looks at the stack  
 119603 pointer, determines the chunk number, and uses that as an index into the chunk table to find its  
 119604 thread object and thus its private value of *errno*. On most architectures, this can be done in four  
 119605 to five instructions. Some compilers may wish to implement *\_\_errno()* inline to improve  
 119606 performance.

## Disallowing Return of the [EINTR] Error Code

Many blocking interfaces defined by POSIX.1-2008 may return [EINTR] if interrupted during their execution by a signal handler. Blocking interfaces introduced under the threads functionality do not have this property. Instead, they require that the interface appear to be atomic with respect to interruption. In particular, clients of blocking interfaces need not handle any possible [EINTR] return as a special case since it will never occur. If it is necessary to restart operations or complete incomplete operations following the execution of a signal handler, this is handled by the implementation, rather than by the application.

Requiring applications to handle [EINTR] errors on blocking interfaces has been shown to be a frequent source of often unreproducible bugs, and it adds no compelling value to the available functionality. Thus, blocking interfaces introduced for use by multi-threaded programs do not use this paradigm. In particular, in none of the functions *flockfile()*, *pthread\_cond\_timedwait()*, *pthread\_cond\_wait()*, *pthread\_join()*, *pthread\_mutex\_lock()*, and *sigwait()* did providing [EINTR] returns add value, or even particularly make sense. Thus, these functions do not provide for an [EINTR] return, even when interrupted by a signal handler. The same arguments can be applied to *sem\_wait()*, *sem\_trywait()*, *sigwaitinfo()*, and *sigtimedwait()*, but implementations are permitted to return [EINTR] error codes for these functions for compatibility with earlier versions of this standard. Applications cannot rely on calls to these functions returning [EINTR] error codes when signals are delivered to the calling thread, but they should allow for the possibility.

Austin Group Interpretation 1003.1-2001 #050 is applied, allowing [ENOTSUP] and [EOPNOTSUP] to be the same values.

### B.2.3.1 Additional Error Numbers

The ISO C standard defines the name space for implementations to add additional error numbers.

## B.2.4 Signal Concepts

Historical implementations of signals, using the *signal()* function, have shortcomings that make them unreliable for many application uses. Because of this, a new signal mechanism, based very closely on the one of 4.2 BSD and 4.3 BSD, was added to POSIX.1.

### Signal Names

The restriction on the actual type used for **sigset\_t** is intended to guarantee that these objects can always be assigned, have their address taken, and be passed as parameters by value. It is not intended that this type be a structure including pointers to other data structures, as that could impact the portability of applications performing such operations. A reasonable implementation could be a structure containing an array of some integer type.

The signals described in POSIX.1-2008 must have unique values so that they may be named as parameters of **case** statements in the body of a C-language **switch** clause. However, implementation-defined signals may have values that overlap with each other or with signals specified in POSIX.1-2008. An example of this is SIGABRT, which traditionally overlaps some other signal, such as SIGIOT.

SIGKILL, SIGTERM, SIGUSR1, and SIGUSR2 are ordinarily generated only through the explicit use of the *kill()* function, although some implementations generate SIGKILL under extraordinary circumstances. SIGTERM is traditionally the default signal sent by the *kill* command.

The signals SIGBUS, SIGEMT, SIGIOT, SIGTRAP, and SIGSYS were omitted from POSIX.1 because their behavior is implementation-defined and could not be adequately categorized. Conforming implementations may deliver these signals, but must document the circumstances under which they are delivered and note any restrictions concerning their delivery. The signals SIGFPE, SIGILL, and SIGSEGV are similar in that they also generally result only from programming errors. They were included in POSIX.1 because they do indicate three relatively well-categorized conditions. They are all defined by the ISO C standard and thus would have to be defined by any system with an ISO C standard binding, even if not explicitly included in POSIX.1.

There is very little that a Conforming POSIX.1 Application can do by catching, ignoring, or masking any of the signals SIGILL, SIGTRAP, SIGIOT, SIGEMT, SIGBUS, SIGSEGV, SIGSYS, or SIGFPE. They will generally be generated by the system only in cases of programming errors. While it may be desirable for some robust code (for example, a library routine) to be able to detect and recover from programming errors in other code, these signals are not nearly sufficient for that purpose. One portable use that does exist for these signals is that a command interpreter can recognize them as the cause of termination of a process (with *wait()*) and print an appropriate message. The mnemonic tags for these signals are derived from their PDP-11 origin.

The signals SIGSTOP, SIGTSTP, SIGTTIN, SIGTTOU, and SIGCONT are provided for job control and are unchanged from 4.2 BSD. The signal SIGCHLD is also typically used by job control shells to detect children that have terminated or, as in 4.2 BSD, stopped.

Some implementations, including System V, have a signal named SIGCLD, which is similar to SIGCHLD in 4.2 BSD. POSIX.1 permits implementations to have a single signal with both names. POSIX.1 carefully specifies ways in which conforming applications can avoid the semantic differences between the two different implementations. The name SIGCHLD was chosen for POSIX.1 because most current application usages of it can remain unchanged in conforming applications. SIGCLD in System V has more cases of semantics that POSIX.1 does not specify, and thus applications using it are more likely to require changes in addition to the name change.

The signals SIGUSR1 and SIGUSR2 are commonly used by applications for notification of exceptional behavior and are described as “reserved as application-defined” so that such use is not prohibited. Implementations should not generate SIGUSR1 or SIGUSR2, except when explicitly requested by *kill()*. It is recommended that libraries not use these two signals, as such use in libraries could interfere with their use by applications calling the libraries. If such use is unavoidable, it should be documented. It is prudent for non-portable libraries to use non-standard signals to avoid conflicts with use of standard signals by portable libraries.

There is no portable way for an application to catch or ignore non-standard signals. Some implementations define the range of signal numbers, so applications can install signal-catching functions for all of them. Unfortunately, implementation-defined signals often cause problems when caught or ignored by applications that do not understand the reason for the signal. While the desire exists for an application to be more robust by handling all possible signals (even those only generated by *kill()*), no existing mechanism was found to be sufficiently portable to include in POSIX.1. The value of such a mechanism, if included, would be diminished given that SIGKILL would still not be catchable.

A number of new signal numbers are reserved for applications because the two user signals defined by POSIX.1 are insufficient for many realtime applications. A range of signal numbers is specified, rather than an enumeration of additional reserved signal names, because different applications and application profiles will require a different number of application signals. It is not desirable to burden all application domains and therefore all implementations with the maximum number of signals required by all possible applications. Note that in this context, signal numbers are essentially different signal priorities.

119701 The relatively small number of required additional signals, {\_POSIX\_RTSIG\_MAX}, was chosen  
 119702 so as not to require an unreasonably large signal mask/set. While this number of signals  
 119703 defined in POSIX.1 will fit in a single 32-bit word signal mask, it is recognized that most existing  
 119704 implementations define many more signals than are specified in POSIX.1 and, in fact, many  
 119705 implementations have already exceeded 32 signals (including the “null signal”). Support of  
 119706 {\_POSIX\_RTSIG\_MAX} additional signals may push some implementation over the single 32-bit  
 119707 word line, but is unlikely to push any implementations that are already over that line beyond  
 119708 the 64-signal line.

#### 119709 B.2.4.1 Signal Generation and Delivery

119710 The terms defined in this section are not used consistently in documentation of historical  
 119711 systems. Each signal can be considered to have a lifetime beginning with generation and ending  
 119712 with delivery or acceptance. The POSIX.1 definition of “delivery” does not exclude ignored  
 119713 signals; this is considered a more consistent definition. This revised text in several parts of  
 119714 POSIX.1-2008 clarifies the distinct semantics of asynchronous signal delivery and synchronous  
 119715 signal acceptance. The previous wording attempted to categorize both under the term  
 119716 “delivery”, which led to conflicts over whether the effects of asynchronous signal delivery  
 119717 applied to synchronous signal acceptance.

119718 Signals generated for a process are delivered to only one thread. Thus, if more than one thread  
 119719 is eligible to receive a signal, one has to be chosen. The choice of threads is left entirely up to the  
 119720 implementation both to allow the widest possible range of conforming implementations and to  
 119721 give implementations the freedom to deliver the signal to the “easiest possible” thread should  
 119722 there be differences in ease of delivery between different threads.

119723 Note that should multiple delivery among cooperating threads be required by an application,  
 119724 this can be trivially constructed out of the provided single-delivery semantics. The construction  
 119725 of a *sigwait\_multiple()* function that accomplishes this goal is presented with the rationale for  
 119726 *sigwaitinfo()*.

119727 Implementations should deliver unblocked signals as soon after they are generated as possible.  
 119728 However, it is difficult for POSIX.1 to make specific requirements about this, beyond those in  
 119729 *kill()* and *sigprocmask()*. Even on systems with prompt delivery, scheduling of higher priority  
 119730 processes is always likely to cause delays.

119731 In general, the interval between the generation and delivery of unblocked signals cannot be  
 119732 detected by an application. Thus, references to pending signals generally apply to blocked,  
 119733 pending signals. An implementation registers a signal as pending on the process when no  
 119734 thread has the signal unblocked and there are no threads blocked in a *sigwait()* function for that  
 119735 signal. Thereafter, the implementation delivers the signal to the first thread that unblocks the  
 119736 signal or calls a *sigwait()* function on a signal set containing this signal rather than choosing the  
 119737 recipient thread at the time the signal is sent.

119738 In the 4.3 BSD system, signals that are blocked and set to SIG\_IGN are discarded immediately  
 119739 upon generation. For a signal that is ignored as its default action, if the action is SIG\_DFL and  
 119740 the signal is blocked, a generated signal remains pending. In the 4.1 BSD system and in  
 119741 System V Release 3 (two other implementations that support a somewhat similar signal  
 119742 mechanism), all ignored blocked signals remain pending if generated. Because it is not normally  
 119743 useful for an application to simultaneously ignore and block the same signal, it was unnecessary  
 119744 for POSIX.1 to specify behavior that would invalidate any of the historical implementations.

119745 There is one case in some historical implementations where an unblocked, pending signal does  
 119746 not remain pending until it is delivered. In the System V implementation of *signal()*, pending  
 119747 signals are discarded when the action is set to SIG\_DFL or a signal-catching routine (as well as  
 119748 to SIG\_IGN). Except in the case of setting SIGCHLD to SIG\_DFL, implementations that do this

do not conform completely to POSIX.1. Some earlier proposals for POSIX.1 explicitly stated this, but these statements were redundant due to the requirement that functions defined by POSIX.1 not change attributes of processes defined by POSIX.1 except as explicitly stated.

POSIX.1 specifically states that the order in which multiple, simultaneously pending signals are delivered is unspecified. This order has not been explicitly specified in historical implementations, but has remained quite consistent and been known to those familiar with the implementations. Thus, there have been cases where applications (usually system utilities) have been written with explicit or implicit dependencies on this order. Implementors and others porting existing applications may need to be aware of such dependencies.

When there are multiple pending signals that are not blocked, implementations should arrange for the delivery of all signals at once, if possible. Some implementations stack calls to all pending signal-catching routines, making it appear that each signal-catcher was interrupted by the next signal. In this case, the implementation should ensure that this stacking of signals does not violate the semantics of the signal masks established by *sigaction()*. Other implementations process at most one signal when the operating system is entered, with remaining signals saved for later delivery. Although this practice is widespread, this behavior is neither standardized nor endorsed. In either case, implementations should attempt to deliver signals associated with the current state of the process (for example, SIGFPE) before other signals, if possible.

In 4.2 BSD and 4.3 BSD, it is not permissible to ignore or explicitly block SIGCONT, because if blocking or ignoring this signal prevented it from continuing a stopped process, such a process could never be continued (only killed by SIGKILL). However, 4.2 BSD and 4.3 BSD do block SIGCONT during execution of its signal-catching function when it is caught, creating exactly this problem. A proposal was considered to disallow catching SIGCONT in addition to ignoring and blocking it, but this limitation led to objections. The consensus was to require that SIGCONT always continue a stopped process when generated. This removed the need to disallow ignoring or explicit blocking of the signal; note that SIG\_IGN and SIG\_DFL are equivalent for SIGCONT.

#### B.2.4.2 Realtime Signal Generation and Delivery

The realtime signals functionality is required in this version of the standard for the following reasons:

- The **sigevent** structure is used by other POSIX.1 functions that result in asynchronous event notifications to specify the notification mechanism to use and other information needed by the notification mechanism. POSIX.1-2008 defines only three symbolic values for the notification mechanism:
  - SIGEV\_NONE is used to indicate that no notification is required when the event occurs. This is useful for applications that use asynchronous I/O with polling for completion.
  - SIGEV\_SIGNAL indicates that a signal is generated when the event occurs.
  - SIGEV\_THREAD provides for “callback functions” for asynchronous notifications done by a function call within the context of a new thread. This provides a multi-threaded process with a more natural means of notification than signals.

The primary difficulty with previous notification approaches has been to specify the environment of the notification routine.

- One approach is to limit the notification routine to call only functions permitted in a signal handler. While the list of permissible functions is clearly stated, this is overly restrictive.



- 119795 — A second approach is to define a new list of functions or classes of functions that are  
 119796 explicitly permitted or not permitted. This would give a programmer more lists to  
 119797 deal with, which would be awkward.
- 119798 — The third approach is to define completely the environment for execution of the  
 119799 notification function. A clear definition of an execution environment for notification  
 119800 is provided by executing the notification function in the environment of a newly  
 119801 created thread.
- 119802 Implementations may support additional notification mechanisms by defining new values  
 119803 for *sigev\_notify*.
- 119804 For a notification type of SIGEV\_SIGNAL, the other members of the **sigevent** structure  
 119805 defined by POSIX.1-2008 specify the realtime signal—that is, the signal number and  
 119806 application-defined value that differentiates between occurrences of signals with the same  
 119807 number—that will be generated when the event occurs. The structure is defined in  
 119808 **<signal.h>**, even though the structure is not directly used by any of the signal functions,  
 119809 because it is part of the signals interface used by the POSIX.1b “client functions”. When the  
 119810 client functions include **<signal.h>** to define the signal names, the **sigevent** structure will  
 119811 also be defined.
- 119812 An application-defined value passed to the signal handler is used to differentiate between  
 119813 different “events” instead of requiring that the application use different signal numbers for  
 119814 several reasons:
- 119815 — Realtime applications potentially handle a very large number of different events.  
 119816 Requiring that implementations support a correspondingly large number of distinct  
 119817 signal numbers will adversely impact the performance of signal delivery because the  
 119818 signal masks to be manipulated on entry and exit to the handlers will become large.
- 119819 — Event notifications are prioritized by signal number (the rationale for this is  
 119820 explained in the following paragraphs) and the use of different signal numbers to  
 119821 differentiate between the different event notifications overloads the signal number  
 119822 more than has already been done. It also requires that the application developer  
 119823 make arbitrary assignments of priority to events that are logically of equal priority.
- 119824 A union is defined for the application-defined value so that either an integer constant or a  
 119825 pointer can be portably passed to the signal-catching function. On some architectures a  
 119826 pointer cannot be cast to an **int** and *vice versa*.
- 119827 Use of a structure here with an explicit notification type discriminant rather than explicit  
 119828 parameters to realtime functions, or embedded in other realtime structures, provides for  
 119829 future extensions to POSIX.1-2008. Additional, perhaps more efficient, notification  
 119830 mechanisms can be supported for existing realtime function interfaces, such as timers and  
 119831 asynchronous I/O, by extending the **sigevent** structure appropriately. The existing  
 119832 realtime function interfaces will not have to be modified to use any such new notification  
 119833 mechanism. The revised text concerning the SIGEV\_SIGNAL value makes consistent the  
 119834 semantics of the members of the **sigevent** structure, particularly in the definitions of  
 119835 *lio\_listio()* and *aio\_fsync()*. For uniformity, other revisions cause this specification to be  
 119836 referred to rather than inaccurately duplicated in the descriptions of functions and  
 119837 structures using the **sigevent** structure. The revised wording does not relax the  
 119838 requirement that the signal number be in the range SIGRTMIN to SIGRTMAX to guarantee  
 119839 queuing and passing of the application value, since that requirement is still implied by the  
 119840 signal names.

- POSIX.1-2008 is intentionally vague on whether “non-realtime” signal-generating mechanisms can result in a **siginfo\_t** being supplied to the handler on delivery. In one existing implementation, a **siginfo\_t** is posted on signal generation, even though the implementation does not support queuing of multiple occurrences of a signal. It is not the intent of POSIX.1-2008 to preclude this, independent of the mandate to define signals that do support queuing. Any interpretation that appears to preclude this is a mistake in the reading or writing of the standard.
- Signals handled by realtime signal handlers might be generated by functions or conditions that do not allow the specification of an application-defined value and do not queue. POSIX.1-2008 specifies the *si\_code* member of the **siginfo\_t** structure used in existing practice and defines additional codes so that applications can detect whether an application-defined value is present or not. The code `SI_USER` for *kill()*-generated signals is adopted from existing practice.
- The *sigaction()* *sa\_flags* value `SA_SIGINFO` tells the implementation that the signal-catching function expects two additional arguments. When the flag is not set, a single argument, the signal number, is passed as specified by POSIX.1-2008. Although POSIX.1-2008 does not explicitly allow the *info* argument to the handler function to be `NULL`, this is existing practice. This provides for compatibility with programs whose signal-catching functions are not prepared to accept the additional arguments. POSIX.1-2008 is explicitly unspecified as to whether signals actually queue when `SA_SIGINFO` is not set for a signal, as there appear to be no benefits to applications in specifying one behavior or another. One existing implementation queues a **siginfo\_t** on each signal generation, unless the signal is already pending, in which case the implementation discards the new **siginfo\_t**; that is, the queue length is never greater than one. This implementation only examines `SA_SIGINFO` on signal delivery, discarding the queued **siginfo\_t** if its delivery was not requested.

The third argument to the signal-catching function, *context*, is left undefined by POSIX.1-2008, but is specified in the interface because it matches existing practice for the `SA_SIGINFO` flag. It was considered undesirable to require a separate implementation for `SA_SIGINFO` for POSIX conformance on implementations that already support the two additional parameters.
- The requirement to deliver lower numbered signals in the range `SIGRTMIN` to `SIGRTMAX` first, when multiple unblocked signals are pending, results from several considerations:

  - A method is required to prioritize event notifications. The signal number was chosen instead of, for instance, associating a separate priority with each request, because an implementation has to check pending signals at various points and select one for delivery when more than one is pending. Specifying a selection order is the minimal additional semantic that will achieve prioritized delivery. If a separate priority were to be associated with queued signals, it would be necessary for an implementation to search all non-empty, non-blocked signal queues and select from among them the pending signal with the highest priority. This would significantly increase the cost of and decrease the determinism of signal delivery.
  - Given the specified selection of the lowest numeric unblocked pending signal, preemptive priority signal delivery can be achieved using signal numbers and signal masks by ensuring that the *sa\_mask* for each signal number blocks all signals with a higher numeric value.

For realtime applications that want to use only the newly defined realtime signal numbers without interference from the standard signals, this can be achieved by blocking all of the standard signals in the thread signal mask and in the *sa\_mask*

- 119890 installed by the signal action for the realtime signal handlers.
- 119891 POSIX.1-2008 explicitly leaves unspecified the ordering of signals outside of the range of  
 119892 realtime signals and the ordering of signals within this range with respect to those outside  
 119893 the range. It was believed that this would unduly constrain implementations or standards  
 119894 in the future definition of new signals.
- 119895 *B.2.4.3 Signal Actions*
- 119896 Early proposals mentioned SIGCONT as a second exception to the rule that signals are not  
 119897 delivered to stopped processes until continued. Because POSIX.1-2008 now specifies that  
 119898 SIGCONT causes the stopped process to continue when it is generated, delivery of SIGCONT is  
 119899 not prevented because a process is stopped, even without an explicit exception to this rule.
- 119900 Ignoring a signal by setting the action to SIG\_IGN (or SIG\_DFL for signals whose default action  
 119901 is to ignore) is not the same as installing a signal-catching function that simply returns. Invoking  
 119902 such a function will interrupt certain system functions that block processes (for example, *wait()*,  
 119903 *sigsuspend()*, *pause()*, *read()*, *write()*) while ignoring a signal has no such effect on the process.
- 119904 Historical implementations discard pending signals when the action is set to SIG\_IGN.  
 119905 However, they do not always do the same when the action is set to SIG\_DFL and the default  
 119906 action is to ignore the signal. POSIX.1-2008 requires this for the sake of consistency and also for  
 119907 completeness, since the only signal this applies to is SIGCHLD, and POSIX.1-2008 disallows  
 119908 setting its action to SIG\_IGN.
- 119909 Some implementations (System V, for example) assign different semantics for SIGCHLD  
 119910 depending on whether the action is set to SIG\_IGN or SIG\_DFL. Since POSIX.1 requires that the  
 119911 default action for SIGCHLD be to ignore the signal, applications should always set the action to  
 119912 SIG\_DFL in order to avoid SIGCHLD.
- 119913 Whether or not an implementation allows SIG\_IGN as a SIGCHLD disposition to be inherited  
 119914 across a call to one of the *exec* family of functions or *posix\_spawn()* is explicitly left as  
 119915 unspecified. This change was made as a result of IEEE PASC Interpretation 1003.1 #132, and  
 119916 permits the implementation to decide between the following alternatives:
- 119917 • Unconditionally leave SIGCHLD set to SIG\_IGN, in which case the implementation would  
 119918 not allow applications that assume inheritance of SIG\_DFL to conform to POSIX.1-2008  
 119919 without change. The implementation would, however, retain an ability to control  
 119920 applications that create child processes but never call on the *wait* family of functions,  
 119921 potentially filling up the process table.
  - 119922 • Unconditionally reset SIGCHLD to SIG\_DFL, in which case the implementation would  
 119923 allow applications that assume inheritance of SIG\_DFL to conform. The implementation  
 119924 would, however, lose an ability to control applications that spawn child processes but  
 119925 never reap them.
  - 119926 • Provide some mechanism, not specified in POSIX.1-2008, to control inherited SIGCHLD  
 119927 dispositions.
- 119928 Some implementations (System V, for example) will deliver a SIGCHLD signal immediately when  
 119929 a process establishes a signal-catching function for SIGCHLD when that process has a child that  
 119930 has already terminated. Other implementations, such as 4.3 BSD, do not generate a new  
 119931 SIGCHLD signal in this way. In general, a process should not attempt to alter the signal action  
 119932 for the SIGCHLD signal while it has any outstanding children. However, it is not always  
 119933 possible for a process to avoid this; for example, shells sometimes start up processes in pipelines  
 119934 with other processes from the pipeline as children. Processes that cannot ensure that they have  
 119935 no children when altering the signal action for SIGCHLD thus need to be prepared for, but not

depend on, generation of an immediate SIGCHLD signal.

The default action of the stop signals (SIGSTOP, SIGTSTP, SIGTTIN, SIGTTOU) is to stop a process that is executing. If a stop signal is delivered to a process that is already stopped, it has no effect. In fact, if a stop signal is generated for a stopped process whose signal mask blocks the signal, the signal will never be delivered to the process since the process must receive a SIGCONT, which discards all pending stop signals, in order to continue executing.

The SIGCONT signal continues a stopped process even if SIGCONT is blocked (or ignored). However, if a signal-catching routine has been established for SIGCONT, it will not be entered until SIGCONT is unblocked.

If a process in an orphaned process group stops, it is no longer under the control of a job control shell and hence would not normally ever be continued. Because of this, orphaned processes that receive terminal-related stop signals (SIGTSTP, SIGTTIN, SIGTTOU, but not SIGSTOP) must not be allowed to stop. The goal is to prevent stopped processes from languishing forever. (As SIGSTOP is sent only via *kill()*, it is assumed that the process or user sending a SIGSTOP can send a SIGCONT when desired.) Instead, the system must discard the stop signal. As an extension, it may also deliver another signal in its place. 4.3 BSD sends a SIGKILL, which is overly effective because SIGKILL is not catchable. Another possible choice is SIGHUP. 4.3 BSD also does this for orphaned processes (processes whose parent has terminated) rather than for members of orphaned process groups; this is less desirable because job control shells manage process groups. POSIX.1 also prevents SIGTTIN and SIGTTOU signals from being generated for processes in orphaned process groups as a direct result of activity on a terminal, preventing infinite loops when *read()* and *write()* calls generate signals that are discarded; see [Section A.11.1.4](#) (on page 3511). A similar restriction on the generation of SIGTSTP was considered, but that would be unnecessary and more difficult to implement due to its asynchronous nature.

Although POSIX.1 requires that signal-catching functions be called with only one argument, there is nothing to prevent conforming implementations from extending POSIX.1 to pass additional arguments, as long as Strictly Conforming POSIX.1 Applications continue to compile and execute correctly. Most historical implementations do, in fact, pass additional, signal-specific arguments to certain signal-catching routines.

There was a proposal to change the declared type of the signal handler to:

```
void func (int sig, ...);
```

The usage of ellipses ("*...*") is ISO C standard syntax to indicate a variable number of arguments. Its use was intended to allow the implementation to pass additional information to the signal handler in a standard manner.

Unfortunately, this construct would require all signal handlers to be defined with this syntax because the ISO C standard allows implementations to use a different parameter passing mechanism for variable parameter lists than for non-variable parameter lists. Thus, all existing signal handlers in all existing applications would have to be changed to use the variable syntax in order to be standard and portable. This is in conflict with the goal of Minimal Changes to Existing Application Code.

When terminating a process from a signal-catching function, processes should be aware of any interpretation that their parent may make of the status returned by *wait()*, *waitid()*, or *waitpid()*. In particular, a signal-catching function should not call *exit(0)* or *\_exit(0)* unless it wants to indicate successful termination. A non-zero argument to *exit()* or *\_exit()* can be used to indicate unsuccessful termination. Alternatively, the process can use *kill()* to send itself a fatal signal (first ensuring that the signal is set to the default action and not blocked). See also the RATIONALE section of the *\_exit()* function.

The behavior of *unsafe* functions, as defined by this section, is undefined when they are invoked

from signal-catching functions in certain circumstances. The behavior of async-signal-safe functions, as defined by this section, is as specified by POSIX.1, regardless of invocation from a signal-catching function. This is the only intended meaning of the statement that async-signal-safe functions may be used in signal-catching functions without restriction. Applications must still consider all effects of such functions on such things as data structures, files, and process state. In particular, application developers need to consider the restrictions on interactions when interrupting *sleep()* (see *sleep()*) and interactions among multiple handles for a file description. The fact that any specific function is listed as async-signal-safe does not necessarily mean that invocation of that function from a signal-catching function is recommended.

In order to prevent errors arising from interrupting non-async-signal-safe function calls, applications should protect calls to these functions either by blocking the appropriate signals or through the use of some programmatic semaphore. POSIX.1 does not address the more general problem of synchronizing access to shared data structures. Note in particular that even the “safe” functions may modify the global variable *errno*; the signal-catching function may want to save and restore its value. The same principles apply to the async-signal-safety of application routines and asynchronous data access.

Note that *longjmp()* and *siglongjmp()* are not in the list of async-signal-safe functions. This is because the code executing after *longjmp()* or *siglongjmp()* can call any unsafe functions with the same danger as calling those unsafe functions directly from the signal handler. Applications that use *longjmp()* or *siglongjmp()* out of signal handlers require rigorous protection in order to be portable. Many of the other functions that are excluded from the list are traditionally implemented using either the C language *malloc()* or *free()* functions or the ISO C standard I/O library, both of which traditionally use data structures in a non-async-signal-safe manner. Because any combination of different functions using a common data structure can cause async-signal-safety problems, POSIX.1 does not define the behavior when any unsafe function is called in a signal handler that interrupts any unsafe function.

The only realtime extension to signal actions is the addition of the additional parameters to the signal-catching function. This extension has been explained and motivated in the previous section. In making this extension, though, developers of POSIX.1b ran into issues relating to function prototypes. In response to input from the POSIX.1 standard developers, members were added to the **sigaction** structure to specify function prototypes for the newer signal-catching function specified by POSIX.1b. These members follow changes that are being made to POSIX.1. Note that POSIX.1-2008 explicitly states that these fields may overlap so that a union can be defined. This enabled existing implementations of POSIX.1 to maintain binary-compatibility when these extensions were added.

The **siginfo\_t** structure was adopted for passing the application-defined value to match existing practice, but the existing practice has no provision for an application-defined value, so this was added. Note that POSIX normally reserves the “\_t” type designation for opaque types. The **siginfo\_t** structure breaks with this convention to follow existing practice and thus promote portability.

POSIX.1-2008 specifies several values for the *si\_code* member of the **siginfo\_t** structure. Some were introduced in POSIX.1b; others were XSI functionality in the Single UNIX Specification, Version 2 and Version 3, that has now become Base functionality. Historically, an *si\_code* value of less than or equal to zero indicated that the signal was generated by a process via the *kill()* function, and values of *si\_code* that provided additional information for implementation-generated signals, such as SIGFPE or SIGSEGV, were all positive. This functionality is partially specified for XSI systems in that if *si\_code* is less than or equal to zero, the signal was generated by a process. However, since POSIX.1b did not specify that SI\_USER (or SI\_QUEUE) had a value less than or equal to zero, it is not true that when the signal is generated by a process, the value of *si\_code* will always be less than or equal to zero. XSI applications should check whether *si\_code*

120034 is `SI_USER` or `SI_QUEUE` in addition to checking whether it is less than or equal to zero.  
 120035 Applications on systems that do not support the XSI option should just check for `SI_USER` and  
 120036 `SI_QUEUE`.

120037 If an implementation chooses to define additional values for *si\_code*, these values have to be  
 120038 different from the values of the non-signal-specific symbols specified by POSIX.1-2008. This will  
 120039 allow conforming applications to differentiate between signals generated by standard events  
 120040 and those generated by other implementation events in a manner compatible with existing  
 120041 practice.

120042 The unique values of *si\_code* for the POSIX.1b asynchronous events have implications for  
 120043 implementations of, for example, asynchronous I/O or message passing in user space library  
 120044 code. Such an implementation will be required to provide a hidden interface to the signal  
 120045 generation mechanism that allows the library to specify the standard values of *si\_code*.

120046 POSIX.1-2008 also specifies additional members of `siginfo_t`, beyond those that were in  
 120047 POSIX.1b. Like the *si\_code* values mentioned above, these were XSI functionality in the Single  
 120048 UNIX Specification, Version 2 and Version 3, that has now become Base functionality. They  
 120049 provide additional information when *si\_code* has one of the values that moved from XSI to Base.

120050 Although it is not explicitly visible to applications, there are additional semantics for signal  
 120051 actions implied by queued signals and their interaction with other POSIX.1b realtime functions.  
 120052 Specifically:

- 120053 • It is not necessary to queue signals whose action is `SIG_IGN`.
- 120054 • For implementations that support POSIX.1b timers, some interaction with the timer  
 120055 functions at signal delivery is implied to manage the timer overrun count.

120056 IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/5 is applied, reordering the RTS shaded  
 120057 text under the third and fourth paragraphs of the `SIG_DFL` description. This corrects an earlier  
 120058 editorial error in this section.

120059 IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/6 is applied, adding the *abort()* function  
 120060 to the list of async-signal-safe functions.

120061 IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/4 is applied, adding the *socketmark()*  
 120062 function to the list of functions that shall be either reentrant or non-interruptible by signals and  
 120063 shall be async-signal-safe.

#### 120064 B.2.4.4 Signal Effects on Other Functions

120065 The most common behavior of an interrupted function after a signal-catching function returns is  
 120066 for the interrupted function to give an `[EINTR]` error unless the `SA_RESTART` flag is in effect for  
 120067 the signal. However, there are a number of specific exceptions, including *sleep()* and certain  
 120068 situations with *read()* and *write()*.

120069 The historical implementations of many functions defined by POSIX.1-2008 are not interruptible,  
 120070 but delay delivery of signals generated during their execution until after they complete. This is  
 120071 never a problem for functions that are guaranteed to complete in a short (imperceptible to a  
 120072 human) period of time. It is normally those functions that can suspend a process indefinitely or  
 120073 for long periods of time (for example, *wait()*, *pause()*, *sigsuspend()*, *sleep()*, or *read()/write()* on a  
 120074 slow device like a terminal) that are interruptible. This permits applications to respond to  
 120075 interactive signals or to set timeouts on calls to most such functions with *alarm()*. Therefore,  
 120076 implementations should generally make such functions (including ones defined as extensions)  
 120077 interruptible.

120078 Functions not mentioned explicitly as interruptible may be so on some implementations,

possibly as an extension where the function gives an [EINTR] error. There are several functions (for example, *getpid()*, *getuid()*) that are specified as never returning an error, which can thus never be extended in this way.

If a signal-catching function returns while the SA\_RESTART flag is in effect, an interrupted function is restarted at the point it was interrupted. Conforming applications cannot make assumptions about the internal behavior of interrupted functions, even if the functions are async-signal-safe. For example, suppose the *read()* function is interrupted with SA\_RESTART in effect, the signal-catching function closes the file descriptor being read from and returns, and the *read()* function is then restarted; in this case the application cannot assume that the *read()* function will give an [EBADF] error, since *read()* might have checked the file descriptor for validity before being interrupted.

## 120090 B.2.5 Standard I/O Streams

Although the ISO C standard guarantees that, at program start-up, *stdin* is open for reading and *stdout* and *stderr* are open for writing, this guarantee is contingent (as are all guarantees made by the ISO C and POSIX standards) on the program being executed in a conforming environment. Programs executed with file descriptor 0 not open for reading or with file descriptor 1 or 2 not open for writing are executed in a non-conforming environment. Application writers are warned (in *exec*, *posix\_spawn()*, and Section C.2.7, on page 3689) not to execute a standard utility or a conforming application with file descriptor 0 not open for reading or with file descriptor 1 or 2 not open for writing.

### 120099 B.2.5.1 Interaction of File Descriptors and Standard I/O Streams

120100 There is no additional rationale provided for this section.

### 120101 B.2.5.2 Stream Orientation and Encoding Rules

120102 There is no additional rationale provided for this section.

## 120103 B.2.6 STREAMS

STREAMS are included into POSIX.1-2008 as part of the alignment with the Single UNIX Specification, but marked as an option in recognition that not all systems may wish to implement the facility. The option within POSIX.1-2008 is denoted by the XSR margin marker. The standard developers made this option independent of the XSI option. In this version of the standard this option is marked obsolescent.

STREAMS are a method of implementing network services and other character-based input/output mechanisms, with the STREAM being a full-duplex connection between a process and a device. STREAMS provides direct access to protocol modules, and optional protocol modules can be interposed between the process-end of the STREAM and the device-driver at the device-end of the STREAM. Pipes can be implemented using the STREAMS mechanism, so they can provide process-to-process as well as process-to-device communications.

This section introduces STREAMS I/O, the message types used to control them, an overview of the priority mechanism, and the interfaces used to access them.

120117 B.2.6.1 *Accessing STREAMS*

120118 There is no additional rationale provided for this section.

120119 **B.2.7 XSI Interprocess Communication**

120120 There are two forms of IPC supported as options in POSIX.1-2008. The traditional System V IPC  
 120121 routines derived from the SVID—that is, the *msg\**(), *sem\**(), and *shm\**() interfaces—are  
 120122 mandatory on XSI-conformant systems. Thus, all XSI-conformant systems provide the same  
 120123 mechanisms for manipulating messages, shared memory, and semaphores.

120124 In addition, the POSIX Realtime Extension provides an alternate set of routines for those systems  
 120125 supporting the appropriate options.

120126 The application developer is presented with a choice: the System V interfaces or the POSIX  
 120127 interfaces (loosely derived from the Berkeley interfaces). The XSI profile prefers the System V  
 120128 interfaces, but the POSIX interfaces may be more suitable for realtime or other performance-  
 120129 sensitive applications.

120130 B.2.7.1 *IPC General Information*

120131 General information that is shared by all three mechanisms is described in this section. The  
 120132 common permissions mechanism is briefly introduced, describing the mode bits, and how they  
 120133 are used to determine whether or not a process has access to read or write/alter the appropriate  
 120134 instance of one of the IPC mechanisms. All other relevant information is contained in the  
 120135 reference pages themselves.

120136 The semaphore type of IPC allows processes to communicate through the exchange of  
 120137 semaphore values. A semaphore is a positive integer. Since many applications require the use of  
 120138 more than one semaphore, XSI-conformant systems have the ability to create sets or arrays of  
 120139 semaphores.

120140 Calls to support semaphores include:

120141 *semctl()*, *semget()*, *semop()*

120142 Semaphore sets are created by using the *semget()* function.

120143 The message type of IPC allows processes to communicate through the exchange of data stored  
 120144 in buffers. This data is transmitted between processes in discrete portions known as messages.

120145 Calls to support message queues include:

120146 *msgctl()*, *msgget()*, *msgrcv()*, *msgsnd()*

120147 The shared memory type of IPC allows two or more processes to share memory and  
 120148 consequently the data contained therein. This is done by allowing processes to set up access to a  
 120149 common memory address space. This sharing of memory provides a fast means of exchange of  
 120150 data between processes.

120151 Calls to support shared memory include:

120152 *shmctl()*, *shmdt()*, *shmget()*

120153 The *ftok()* interface is also provided.



**B.2.8 Realtime****Advisory Information**

POSIX.1b contains an Informative Annex with proposed interfaces for “realtime files”. These interfaces could determine groups of the exact parameters required to do “direct I/O” or “extents”. These interfaces were objected to by a significant portion of the balloting group as too complex. A conforming application had little chance of correctly navigating the large parameter space to match its desires to the system. In addition, they only applied to a new type of file (realtime files) and they told the implementation exactly what to do as opposed to advising the implementation on application behavior and letting it optimize for the system the (portable) application was running on. For example, it was not clear how a system that had a disk array should set its parameters.

There seemed to be several overall goals:

- Optimizing sequential access
- Optimizing caching behavior
- Optimizing I/O data transfer
- Preallocation

The advisory interfaces, *posix\_fadvise()* and *posix\_madvise()*, satisfy the first two goals. The `POSIX_FADV_SEQUENTIAL` and `POSIX_MADV_SEQUENTIAL` advice tells the implementation to expect serial access. Typically the system will prefetch the next several serial accesses in order to overlap I/O. It may also free previously accessed serial data if memory is tight. If the application is not doing serial access it can use `POSIX_FADV_WILLNEED` and `POSIX_MADV_WILLNEED` to accomplish I/O overlap, as required. When the application advises `POSIX_FADV_RANDOM` or `POSIX_MADV_RANDOM` behavior, the implementation usually tries to fetch a minimum amount of data with each request and it does not expect much locality. `POSIX_FADV_DONTNEED` and `POSIX_MADV_DONTNEED` allow the system to free up caching resources as the data will not be required in the near future.

`POSIX_FADV_NOREUSE` tells the system that caching the specified data is not optimal. For file I/O, the transfer should go directly to the user buffer instead of being cached internally by the implementation. To portably perform direct disk I/O on all systems, the application must perform its I/O transfers according to the following rules:

1. The user buffer should be aligned according to the `{POSIX_REC_XFER_ALIGN} pathconf()` variable.
2. The number of bytes transferred in an I/O operation should be a multiple of the `{POSIX_ALLOC_SIZE_MIN} pathconf()` variable.
3. The offset into the file at the start of an I/O operation should be a multiple of the `{POSIX_ALLOC_SIZE_MIN} pathconf()` variable.
4. The application should ensure that all threads which open a given file specify `POSIX_FADV_NOREUSE` to be sure that there is no unexpected interaction between threads using buffered I/O and threads using direct I/O to the same file.

In some cases, a user buffer must be properly aligned in order to be transferred directly to/from the device. The `{POSIX_REC_XFER_ALIGN} pathconf()` variable tells the application the proper alignment.

The preallocation goal is met by the space control function, *posix\_fallocate()*. The application can use *posix\_fallocate()* to guarantee no `[ENOSPC]` errors and to improve performance by prepaying any overhead required for block allocation.

Implementations may use information conveyed by a previous *posix\_fadvise()* call to influence the manner in which allocation is performed. For example, if an application did the following calls:

```
fd = open("file");
posix_fadvise(fd, offset, len, POSIX_FADV_SEQUENTIAL);
posix_fallocate(fd, len, size);
```

an implementation might allocate the file contiguously on disk.

Finally, the *pathconf()* variables {POSIX\_REC\_MIN\_XFER\_SIZE}, {POSIX\_REC\_MAX\_XFER\_SIZE}, and {POSIX\_REC\_INCR\_XFER\_SIZE} tell the application a range of transfer sizes that are recommended for best I/O performance.

Where bounded response time is required, the vendor can supply the appropriate settings of the advisories to achieve a guaranteed performance level.

The interfaces meet the goals while allowing applications using regular files to take advantage of performance optimizations. The interfaces tell the implementation expected application behavior which the implementation can use to optimize performance on a particular system with a particular dynamic load.

The *posix\_memalign()* function was added to allow for the allocation of specifically aligned buffers; for example, for {POSIX\_REC\_XFER\_ALIGN}.

The working group also considered the alternative of adding a function which would return an aligned pointer to memory within a user-supplied buffer. This was not considered to be the best method, because it potentially wastes large amounts of memory when buffers need to be aligned on large alignment boundaries.

## Message Passing

This section provides the rationale for the definition of the message passing interface in POSIX.1-2008. This is presented in terms of the objectives, models, and requirements imposed upon this interface.

- Objectives

Many applications, including both realtime and database applications, require a means of passing arbitrary amounts of data between cooperating processes comprising the overall application on one or more processors. Many conventional interfaces for interprocess communication are insufficient for realtime applications in that efficient and deterministic data passing methods cannot be implemented. This has prompted the definition of message passing interfaces providing these facilities:

- Open a message queue.
- Send a message to a message queue.
- Receive a message from a queue, either synchronously or asynchronously.
- Alter message queue attributes for flow and resource control.

It is assumed that an application may consist of multiple cooperating processes and that these processes may wish to communicate and coordinate their activities. The message passing facility described in POSIX.1-2008 allows processes to communicate through system-wide queues. These message queues are accessed through names that may be pathnames. A message queue can be opened for use by multiple sending and/or multiple receiving processes.

- Background on Embedded Applications

Interprocess communication utilizing message passing is a key facility for the construction of deterministic, high-performance realtime applications. The facility is present in all realtime systems and is the framework upon which the application is constructed. The performance of the facility is usually a direct indication of the performance of the resulting application.

Realtime applications, especially for embedded systems, are typically designed around the performance constraints imposed by the message passing mechanisms. Applications for embedded systems are typically very tightly constrained. Application developers expect to design and control the entire system. In order to minimize system costs, the writer will attempt to use all resources to their utmost and minimize the requirement to add additional memory or processors.

The embedded applications usually share address spaces and only a simple message passing mechanism is required. The application can readily access common data incurring only mutual-exclusion overheads. The models desired are the simplest possible with the application building higher-level facilities only when needed.

- Requirements

The following requirements determined the features of the message passing facilities defined in POSIX.1-2008:

- Naming of Message Queues

The mechanism for gaining access to a message queue is a pathname evaluated in a context that is allowed to be a file system name space, or it can be independent of any file system. This is a specific attempt to allow implementations based on either method in order to address both embedded systems and to also allow implementation in larger systems.

The interface of `mq_open()` is defined to allow but not require the access control and name conflicts resulting from utilizing a file system for name resolution. All required behavior is specified for the access control case. Yet a conforming implementation, such as an embedded system kernel, may define that there are no distinctions between users and may define that all processes have all access privileges.

- Embedded System Naming

Embedded systems need to be able to utilize independent name spaces for accessing the various system objects. They typically do not have a file system, precluding its utilization as a common name resolution mechanism. The modularity of an embedded system limits the connections between separate mechanisms that can be allowed.

Embedded systems typically do not have any access protection. Since the system does not support the mixing of applications from different areas, and usually does not even have the concept of an authorization entity, access control is not useful.

- Large System Naming

On systems with more functionality, the name resolution must support the ability to use the file system as the name resolution mechanism/object storage medium and to have control over access to the objects. Utilizing the pathname space can result in further errors when the names conflict with other objects.

— Fixed Size of Messages

The interfaces impose a fixed upper bound on the size of messages that can be sent to a specific message queue. The size is set on an individual queue basis and cannot be changed dynamically.

The purpose of the fixed size is to increase the ability of the system to optimize the implementation of *mq\_send()* and *mq\_receive()*. With fixed sizes of messages and fixed numbers of messages, specific message blocks can be pre-allocated. This eliminates a significant amount of checking for errors and boundary conditions. Additionally, an implementation can optimize data copying to maximize performance. Finally, with a restricted range of message sizes, an implementation is better able to provide deterministic operations.

— Prioritization of Messages

Message prioritization allows the application to determine the order in which messages are received. Prioritization of messages is a key facility that is provided by most realtime kernels and is heavily utilized by the applications. The major purpose of having priorities in message queues is to avoid priority inversions in the message system, where a high-priority message is delayed behind one or more lower-priority messages. This allows the applications to be designed so that they do not need to be interrupted in order to change the flow of control when exceptional conditions occur. The prioritization does add additional overhead to the message operations in those cases it is actually used but a clever implementation can optimize for the FIFO case to make that more efficient.

— Asynchronous Notification

The interface supports the ability to have a task asynchronously notified of the availability of a message on the queue. The purpose of this facility is to allow the task to perform other functions and yet still be notified that a message has become available on the queue.

To understand the requirement for this function, it is useful to understand two models of application design: a single task performing multiple functions and multiple tasks performing a single function. Each of these models has advantages.

Asynchronous notification is required to build the model of a single task performing multiple operations. This model typically results from either the expectation that interruption is less expensive than utilizing a separate task or from the growth of the application to include additional functions.

## Semaphores

Semaphores are a high-performance process synchronization mechanism. Semaphores are named by null-terminated strings of characters.

A semaphore is created using the *sem\_init()* function or the *sem\_open()* function with the *O\_CREAT* flag set in *oflag*.

To use a semaphore, a process has to first initialize the semaphore or inherit an open descriptor for the semaphore via *fork()*.

A semaphore preserves its state when the last reference is closed. For example, if a semaphore has a value of 13 when the last reference is closed, it will have a value of 13 when it is next opened.

When a semaphore is created, an initial state for the semaphore has to be provided. This value is

a non-negative integer. Negative values are not possible since they indicate the presence of blocked processes. The persistence of any of these objects across a system crash or a system reboot is undefined. Conforming applications must not depend on any sort of persistence across a system reboot or a system crash.

- Models and Requirements

A realtime system requires synchronization and communication between the processes comprising the overall application. An efficient and reliable synchronization mechanism has to be provided in a realtime system that will allow more than one schedulable process mutually-exclusive access to the same resource. This synchronization mechanism has to allow for the optimal implementation of synchronization or systems implementors will define other, more cost-effective methods.

At issue are the methods whereby multiple processes (tasks) can be designed and implemented to work together in order to perform a single function. This requires interprocess communication and synchronization. A semaphore mechanism is the lowest level of synchronization that can be provided by an operating system.

A semaphore is defined as an object that has an integral value and a set of blocked processes associated with it. If the value is positive or zero, then the set of blocked processes is empty; otherwise, the size of the set is equal to the absolute value of the semaphore value. The value of the semaphore can be incremented or decremented by any process with access to the semaphore and must be done as an indivisible operation. When a semaphore value is less than or equal to zero, any process that attempts to lock it again will block or be informed that it is not possible to perform the operation.

A semaphore may be used to guard access to any resource accessible by more than one schedulable task in the system. It is a global entity and not associated with any particular process. As such, a method of obtaining access to the semaphore has to be provided by the operating system. A process that wants access to a critical resource (section) has to wait on the semaphore that guards that resource. When the semaphore is locked on behalf of a process, it knows that it can utilize the resource without interference by any other cooperating process in the system. When the process finishes its operation on the resource, leaving it in a well-defined state, it posts the semaphore, indicating that some other process may now obtain the resource associated with that semaphore.

In this section, mutexes and condition variables are specified as the synchronization mechanisms between threads.

These primitives are typically used for synchronizing threads that share memory in a single process. However, this section provides an option allowing the use of these synchronization interfaces and objects between processes that share memory, regardless of the method for sharing memory.

Much experience with semaphores shows that there are two distinct uses of synchronization: locking, which is typically of short duration; and waiting, which is typically of long or unbounded duration. These distinct usages map directly onto mutexes and condition variables, respectively.

Semaphores are provided in POSIX.1-2008 primarily to provide a means of synchronization for processes; these processes may or may not share memory. Mutexes and condition variables are specified as synchronization mechanisms between threads; these threads always share (some) memory. Both are synchronization paradigms that have been in widespread use for a number of years. Each set of primitives is particularly well matched to certain problems.

With respect to binary semaphores, experience has shown that condition variables and

mutexes are easier to use for many synchronization problems than binary semaphores. The primary reason for this is the explicit appearance of a Boolean predicate that specifies when the condition wait is satisfied. This Boolean predicate terminates a loop, including the call to `pthread_cond_wait()`. As a result, extra wakeups are benign since the predicate governs whether the thread will actually proceed past the condition wait. With stateful primitives, such as binary semaphores, the wakeup in itself typically means that the wait is satisfied. The burden of ensuring correctness for such waits is thus placed on *all* signalers of the semaphore rather than on an *explicitly coded* Boolean predicate located at the condition wait. Experience has shown that the latter creates a major improvement in safety and ease-of-use.

Counting semaphores are well matched to dealing with producer/consumer problems, including those that might exist between threads of different processes, or between a signal handler and a thread. In the former case, there may be little or no memory shared by the processes; in the latter case, one is not communicating between co-equal threads, but between a thread and an interrupt-like entity. It is for these reasons that POSIX.1-2008 allows semaphores to be used by threads.

Mutexes and condition variables have been effectively used with and without priority inheritance, priority ceiling, and other attributes to synchronize threads that share memory. The efficiency of their implementation is comparable to or better than that of other synchronization primitives that are sometimes harder to use (for example, binary semaphores). Furthermore, there is at least one known implementation of Ada tasking that uses these primitives. Mutexes and condition variables together constitute an appropriate, sufficient, and complete set of inter-thread synchronization primitives.

Efficient multi-threaded applications require high-performance synchronization primitives. Considerations of efficiency and generality require a small set of primitives upon which more sophisticated synchronization functions can be built.

#### • Standardization Issues

It is possible to implement very high-performance semaphores using test-and-set instructions on shared memory locations. The library routines that implement such a high-performance interface have to properly ensure that a `sem_wait()` or `sem_trywait()` operation that cannot be performed will issue a blocking semaphore system call or properly report the condition to the application. The same interface to the application program would be provided by a high-performance implementation.

### 120412 B.2.8.1 Realtime Signals

#### 120413 Realtime Signals Extension

This portion of the rationale presents models, requirements, and standardization issues relevant to the Realtime Signals Extension. This extension provides the capability required to support reliable, deterministic, asynchronous notification of events. While a new mechanism, unencumbered by the historical usage and semantics of POSIX.1 signals, might allow for a more efficient implementation, the application requirements for event notification can be met with a small number of extensions to signals. Therefore, a minimal set of extensions to signals to support the application requirements is specified.

The realtime signal extensions specified in this section are used by other realtime functions requiring asynchronous notification:

#### • Models

The model supported is one of multiple cooperating processes, each of which handles

multiple asynchronous external events. Events represent occurrences that are generated as the result of some activity in the system. Examples of occurrences that can constitute an event include:

- Completion of an asynchronous I/O request
- Expiration of a POSIX.1b timer
- Arrival of an interprocess message
- Generation of a user-defined event

Processing of these events may occur synchronously via polling for event notifications or asynchronously via a software interrupt mechanism. Existing practice for this model is well established for traditional proprietary realtime operating systems, realtime executives, and realtime extended POSIX-like systems.

A contrasting model is that of “cooperating sequential processes” where each process handles a single priority of events via polling. Each process blocks while waiting for events, and each process depends on the preemptive, priority-based process scheduling mechanism to arbitrate between events of different priority that need to be processed concurrently. Existing practice for this model is also well established for small realtime executives that typically execute in an unprotected physical address space, but it is just emerging in the context of a fuller function operating system with multiple virtual address spaces.

It could be argued that the cooperating sequential process model, and the facilities supported by the POSIX Threads Extension obviate a software interrupt model. But, even with the cooperating sequential process model, the need has been recognized for a software interrupt model to handle exceptional conditions and process aborting, so the mechanism must be supported in any case. Furthermore, it is not the purview of POSIX.1-2008 to attempt to convince realtime practitioners that their current application models based on software interrupts are “broken” and should be replaced by the cooperating sequential process model. Rather, it is the charter of POSIX.1-2008 to provide standard extensions to mechanisms that support existing realtime practice.

#### • Requirements

This section discusses the following realtime application requirements for asynchronous event notification:

- Reliable delivery of asynchronous event notification

The events notification mechanism guarantees delivery of an event notification. Asynchronous operations (such as asynchronous I/O and timers) that complete significantly after they are invoked have to guarantee that delivery of the event notification can occur at the time of completion.

- Prioritized handling of asynchronous event notifications

The events notification mechanism supports the assigning of a user function as an event notification handler. Furthermore, the mechanism supports the preemption of an event handler function by a higher priority event notification and supports the selection of the highest priority pending event notification when multiple notifications (of different priority) are pending simultaneously.

The model here is based on hardware interrupts. Asynchronous event handling allows the application to ensure that time-critical events are immediately processed when delivered, without the indeterminism of being at a random location within a polling loop. Use of handler priority allows the specification of how handlers are

- 120471 interrupted by other higher priority handlers.
- 120472 — Differentiation between multiple occurrences of event notifications of the same type
- 120473 The events notification mechanism passes an application-defined value to the event
- 120474 handler function. This value can be used for a variety of purposes, such as enabling
- 120475 the application to identify which of several possible events of the same type (for
- 120476 example, timer expirations) has occurred.
- 120477 — Polled reception of asynchronous event notifications
- 120478 The events notification mechanism supports blocking and non-blocking polls for
- 120479 asynchronous event notification.
- 120480 The polled mode of operation is often preferred over the interrupt mode by those
- 120481 practitioners accustomed to this model. Providing support for this model facilitates
- 120482 the porting of applications based on this model to POSIX.1b conforming systems.
- 120483 — Deterministic response to asynchronous event notifications
- 120484 The events notification mechanism does not preclude implementations that provide
- 120485 deterministic event dispatch latency and minimizes the number of system calls
- 120486 needed to use the event facilities during realtime processing.
- 120487 • Rationale for Extension
- 120488 POSIX.1 signals have many of the characteristics necessary to support the asynchronous
- 120489 handling of event notifications, and the Realtime Signals Extension addresses the
- 120490 following deficiencies in the POSIX.1 signal mechanism:
- 120491 — Signals do not support reliable delivery of event notification. Subsequent
- 120492 occurrences of a pending signal are not guaranteed to be delivered.
- 120493 — Signals do not support prioritized delivery of event notifications. The order of signal
- 120494 delivery when multiple unblocked signals are pending is undefined.
- 120495 — Signals do not support the differentiation between multiple signals of the same type.
- 120496 *B.2.8.2 Asynchronous I/O*
- 120497 Many applications need to interact with the I/O subsystem in an asynchronous manner. The
- 120498 asynchronous I/O mechanism provides the ability to overlap application processing and I/O
- 120499 operations initiated by the application. The asynchronous I/O mechanism allows a single
- 120500 process to perform I/O simultaneously to a single file multiple times or to multiple files
- 120501 multiple times.
- 120502 **Overview**
- 120503 Asynchronous I/O operations proceed in logical parallel with the processing done by the
- 120504 application after the asynchronous I/O has been initiated. Other than this difference,
- 120505 asynchronous I/O behaves similarly to normal I/O using *read()*, *write()*, *lseek()*, and *fsync()*.
- 120506 The effect of issuing an asynchronous I/O request is as if a separate thread of execution were to
- 120507 perform atomically the implied *lseek()* operation, if any, and then the requested I/O operation
- 120508 (either *read()*, *write()*, or *fsync()*). There is no seek implied with a call to *aio\_fsync()*. Concurrent
- 120509 asynchronous operations and synchronous operations applied to the same file update the file as
- 120510 if the I/O operations had proceeded serially.
- 120511 When asynchronous I/O completes, a signal can be delivered to the application to indicate the
- 120512 completion of the I/O. This signal can be used to indicate that buffers and control blocks used
- 120513 for asynchronous I/O can be reused. Signal delivery is not required for an asynchronous



operation and may be turned off on a per-operation basis by the application. Signals may also be synchronously polled using *aio\_suspend()*, *sigtimedwait()*, or *sigwaitinfo()*.

Normal I/O has a return value and an error status associated with it. Asynchronous I/O returns a value and an error status when the operation is first submitted, but that only relates to whether the operation was successfully queued up for servicing. The I/O operation itself also has a return status and an error value. To allow the application to retrieve the return status and the error value, functions are provided that, given the address of an asynchronous I/O control block, yield the return and error status associated with the operation. Until an asynchronous I/O operation is done, its error status is [EINPROGRESS]. Thus, an application can poll for completion of an asynchronous I/O operation by waiting for the error status to become equal to a value other than [EINPROGRESS]. The return status of an asynchronous I/O operation is undefined so long as the error status is equal to [EINPROGRESS].

Storage for asynchronous operation return and error status may be limited. Submission of asynchronous I/O operations may fail if this storage is exceeded. When an application retrieves the return status of a given asynchronous operation, therefore, any system-maintained storage used for this status and the error status may be reclaimed for use by other asynchronous operations.

Asynchronous I/O can be performed on file descriptors that have been enabled for POSIX.1b synchronized I/O. In this case, the I/O operation still occurs asynchronously, as defined herein; however, the asynchronous operation I/O in this case is not completed until the I/O has reached either the state of synchronized I/O data integrity completion or synchronized I/O file integrity completion, depending on the sort of synchronized I/O that is enabled on the file descriptor.

## Models

Three models illustrate the use of asynchronous I/O: a journalization model, a data acquisition model, and a model of the use of asynchronous I/O in supercomputing applications.

- Journalization Model

Many realtime applications perform low-priority journalizing functions. Journalizing requires that logging records be queued for output without blocking the initiating process.

- Data Acquisition Model

A data acquisition process may also serve as a model. The process has two or more channels delivering intermittent data that must be read within a certain time. The process issues one asynchronous read on each channel. When one of the channels needs data collection, the process reads the data and posts it through an asynchronous write to secondary memory for future processing.

- Supercomputing Model

The supercomputing community has used asynchronous I/O much like that specified in POSIX.1 for many years. This community requires the ability to perform multiple I/O operations to multiple devices with a minimal number of entries to “the system”; each entry to “the system” provokes a major delay in operations when compared to the normal progress made by the application. This existing practice motivated the use of combined *lseek()* and *read()* or *write()* calls, as well as the *lio\_listio()* call. Another common practice is to disable signal notification for I/O completion, and simply poll for I/O completion at some interval by which the I/O should be completed. Likewise, interfaces like *aio\_cancel()* have been in successful commercial use for many years. Note also that an underlying implementation of asynchronous I/O will require the ability, at least internally, to cancel outstanding asynchronous I/O, at least when the process exits. (Consider an asynchronous read from a terminal, when the process intends to exit immediately.)

**Requirements**

Asynchronous input and output for realtime implementations have these requirements:

- The ability to queue multiple asynchronous read and write operations to a single open instance. Both sequential and random access should be supported.
- The ability to queue asynchronous read and write operations to multiple open instances.
- The ability to obtain completion status information by polling and/or asynchronous event notification.
- Asynchronous event notification on asynchronous I/O completion is optional.
- It has to be possible for the application to associate the event with the *aioctx* for the operation that generated the event.
- The ability to cancel queued requests.
- The ability to wait upon asynchronous I/O completion in conjunction with other types of events.
- The ability to accept an *aio\_read()* and an *aio\_cancel()* for a device that accepts a *read()*, and the ability to accept an *aio\_write()* and an *aio\_cancel()* for a device that accepts a *write()*. This does not imply that the operation is asynchronous.

**Standardization Issues**

The following issues are addressed by the standardization of asynchronous I/O:

- Rationale for New Interface

Non-blocking I/O does not satisfy the needs of either realtime or high-performance computing models; these models require that a process overlap program execution and I/O processing. Realtime applications will often make use of direct I/O to or from the address space of the process, or require synchronized (unbuffered) I/O; they also require the ability to overlap this I/O with other computation. In addition, asynchronous I/O allows an application to keep a device busy at all times, possibly achieving greater throughput. Supercomputing and database architectures will often have specialized hardware that can provide true asynchrony underlying the logical asynchrony provided by this interface. In addition, asynchronous I/O should be supported by all types of files and devices in the same manner.

- Effect of Buffering

If asynchronous I/O is performed on a file that is buffered prior to being actually written to the device, it is possible that asynchronous I/O will offer no performance advantage over normal I/O; the cycles *stolen* to perform the asynchronous I/O will be taken away from the running process and the I/O will occur at interrupt time. This potential lack of gain in performance in no way obviates the need for asynchronous I/O by realtime applications, which very often will use specialized hardware support, multiple processors, and/or unbuffered, synchronized I/O.

**B.2.8.3 Memory Management**

All memory management and shared memory definitions are located in the `<sys/mman.h>` header. This is for alignment with historical practice.

IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/7 is applied, correcting the shading and margin markers in the introduction to Section 2.8.3.1.

## Memory Locking Functions

This portion of the rationale presents models, requirements, and standardization issues relevant to process memory locking.

- Models

Realtime systems that conform to POSIX.1-2008 are expected (and desired) to be supported on systems with demand-paged virtual memory management, non-paged swapping memory management, and physical memory systems with no memory management hardware. The general case, however, is the demand-paged, virtual memory system with each POSIX process running in a virtual address space. Note that this includes architectures where each process resides in its own virtual address space and architectures where the address space of each process is only a portion of a larger global virtual address space.

The concept of memory locking is introduced to eliminate the indeterminacy introduced by paging and swapping, and to support an upper bound on the time required to access the memory mapped into the address space of a process. Ideally, this upper bound will be the same as the time required for the processor to access “main memory”, including any address translation and cache miss overheads. But some implementations—primarily on mainframes—will not actually force locked pages to be loaded and held resident in main memory. Rather, they will handle locked pages so that accesses to these pages will meet the performance metrics for locked process memory in the implementation. Also, although it is not, for example, the intention that this interface, as specified, be used to lock process memory into “cache”, it is conceivable that an implementation could support a large static RAM memory and define this as “main memory” and use a large[r] dynamic RAM as “backing store”. These interfaces could then be interpreted as supporting the locking of process memory into the static RAM. Support for multiple levels of backing store would require extensions to these interfaces.

Implementations may also use memory locking to guarantee a fixed translation between virtual and physical addresses where such is beneficial to improving determinacy for direct-to/from-process input/output. POSIX.1-2008 does not guarantee to the application that the virtual-to-physical address translations, if such exist, are fixed, because such behavior would not be implementable on all architectures on which implementations of POSIX.1-2008 are expected. But POSIX.1-2008 does mandate that an implementation define, for the benefit of potential users, whether or not locking guarantees fixed translations.

Memory locking is defined with respect to the address space of a process. Only the pages mapped into the address space of a process may be locked by the process, and when the pages are no longer mapped into the address space—for whatever reason—the locks established with respect to that address space are removed. Shared memory areas warrant special mention, as they may be mapped into more than one address space or mapped more than once into the address space of a process; locks may be established on pages within these areas with respect to several of these mappings. In such a case, the lock state of the underlying physical pages is the logical OR of the lock state with respect to each of the mappings. Only when all such locks have been removed are the shared pages considered unlocked.

In recognition of the page granularity of Memory Management Units (MMU), and in order to support locking of ranges of address space, memory locking is defined in terms of “page” granularity. That is, for the interfaces that support an address and size specification for the region to be locked, the address must be on a page boundary, and all pages mapped by the specified range are locked, if valid. This means that the length is implicitly rounded

up to a multiple of the page size. The page size is implementation-defined and is available to applications as a compile-time symbolic constant or at runtime via *sysconf()*.

A “real memory” POSIX.1b implementation that has no MMU could elect not to support these interfaces, returning [ENOSYS]. But an application could easily interpret this as meaning that the implementation would unconditionally page or swap the application when such is not the case. It is the intention of POSIX.1-2008 that such a system could define these interfaces as “NO-OPs”, returning success without actually performing any function except for mandated argument checking.

- Requirements

For realtime applications, memory locking is generally considered to be required as part of application initialization. This locking is performed after an application has been loaded (that is, *exec'd*) and the program remains locked for its entire lifetime. But to support applications that undergo major mode changes where, in one mode, locking is required, but in another it is not, the specified interfaces allow repeated locking and unlocking of memory within the lifetime of a process.

When a realtime application locks its address space, it should not be necessary for the application to then “touch” all of the pages in the address space to guarantee that they are resident or else suffer potential paging delays the first time the page is referenced. Thus, POSIX.1-2008 requires that the pages locked by the specified interfaces be resident when the locking functions return successfully.

Many architectures support system-managed stacks that grow automatically when the current extent of the stack is exceeded. A realtime application has a requirement to be able to “preallocate” sufficient stack space and lock it down so that it will not suffer page faults to grow the stack during critical realtime operation. There was no consensus on a portable way to specify how much stack space is needed, so POSIX.1-2008 supports no specific interface for preallocating stack space. But an application can portably lock down a specific amount of stack space by specifying *MCL\_FUTURE* in a call to *mlockall()* and then calling a dummy function that declares an automatic array of the desired size.

Memory locking for realtime applications is also generally considered to be an “all or nothing” proposition. That is, the entire process, or none, is locked down. But, for applications that have well-defined sections that need to be locked and others that do not, POSIX.1-2008 supports an optional set of interfaces to lock or unlock a range of process addresses. Reasons for locking down a specific range include:

- An asynchronous event handler function that must respond to external events in a deterministic manner such that page faults cannot be tolerated
- An input/output “buffer” area that is the target for direct-to-process I/O, and the overhead of implicit locking and unlocking for each I/O call cannot be tolerated

Finally, locking is generally viewed as an “application-wide” function. That is, the application is globally aware of which regions are locked and which are not over time. This is in contrast to a function that is used temporarily within a “third party” library routine whose function is unknown to the application, and therefore must have no “side-effects”. The specified interfaces, therefore, do not support “lock stacking” or “lock nesting” within a process. But, for pages that are shared between processes or mapped more than once into a process address space, “lock stacking” is essentially mandated by the requirement that unlocking of pages that are mapped by more than one process or more than once by the same process does not affect locks established on the other mappings.

There was some support for “lock stacking” so that locking could be transparently used in functions or opaque modules. But the consensus was not to burden all implementations

120700 with lock stacking (and reference counting), and an implementation option was proposed.  
120701 There were strong objections to the option because applications would have to support  
120702 both options in order to remain portable. The consensus was to eliminate lock stacking  
120703 altogether, primarily through overwhelming support for the System V “m[un]lock[all]”  
120704 interface on which POSIX.1-2008 is now based.

120705 Locks are not inherited across *fork()*s because some implementations implement *fork()* by  
120706 creating new address spaces for the child. In such an implementation, requiring locks to be  
120707 inherited would lead to new situations in which a fork would fail due to the inability of  
120708 the system to lock sufficient memory to lock both the parent and the child. The consensus  
120709 was that there was no benefit to such inheritance. Note that this does not mean that locks  
120710 are removed when, for instance, a thread is created in the same address space.

120711 Similarly, locks are not inherited across *exec* because some implementations implement *exec*  
120712 by unmapping all of the pages in the address space (which, by definition, removes the  
120713 locks on these pages), and maps in pages of the *exec*'d image. In such an implementation,  
120714 requiring locks to be inherited would lead to new situations in which *exec* would fail.  
120715 Reporting this failure would be very cumbersome to detect in time to report to the calling  
120716 process, and no appropriate mechanism exists for informing the *exec*'d process of its status.

120717 It was determined that, if the newly loaded application required locking, it was the  
120718 responsibility of that application to establish the locks. This is also in keeping with the  
120719 general view that it is the responsibility of the application to be aware of all locks that are  
120720 established.

120721 There was one request to allow (not mandate) locks to be inherited across *fork()*, and a  
120722 request for a flag, *MCL\_INHERIT*, that would specify inheritance of memory locks across  
120723 *execs*. Given the difficulties raised by this and the general lack of support for the feature in  
120724 POSIX.1-2008, it was not added. POSIX.1-2008 does not preclude an implementation from  
120725 providing this feature for administrative purposes, such as a “run” command that will  
120726 lock down and execute a specified application. Additionally, the rationale for the objection  
120727 equated *fork()* with creating a thread in the address space. POSIX.1-2008 does not mandate  
120728 releasing locks when creating additional threads in an existing process.

120729 • Standardization Issues

120730 One goal of POSIX.1-2008 is to define a set of primitives that provide the necessary  
120731 functionality for realtime applications, with consideration for the needs of other  
120732 application domains where such were identified, which is based to the extent possible on  
120733 existing industry practice.

120734 The Memory Locking option is required by many realtime applications to tune  
120735 performance. Such a facility is accomplished by placing constraints on the virtual memory  
120736 system to limit paging of time of the process or of critical sections of the process. This  
120737 facility should not be used by most non-realtime applications.

120738 Optional features provided in POSIX.1-2008 allow applications to lock selected address  
120739 ranges with the caveat that the process is responsible for being aware of the page  
120740 granularity of locking and the unnested nature of the locks.

## Mapped Files Functions

The memory mapped files functionality provides a mechanism that allows a process to access files by directly incorporating file data into its address space. Once a file is “mapped” into a process address space, the data can be manipulated by instructions as memory. The use of mapped files can significantly reduce I/O data movement since file data does not have to be copied into process data buffers as in *read()* and *write()*. If more than one process maps a file, its contents are shared among them. This provides a low overhead mechanism by which processes can synchronize and communicate.

- Historical Perspective

Realtime applications have historically been implemented using a collection of cooperating processes or tasks. In early systems, these processes ran on bare hardware (that is, without an operating system) with no memory relocation or protection. The application paradigms that arose from this environment involve the sharing of data between the processes.

When realtime systems were implemented on top of vendor-supplied operating systems, the paradigm or performance benefits of direct access to data by multiple processes was still deemed necessary. As a result, operating systems that claim to support realtime applications must support the shared memory paradigm.

Additionally, a number of realtime systems provide the ability to map specific sections of the physical address space into the address space of a process. This ability is required if an application is to obtain direct access to memory locations that have specific properties (for example, refresh buffers or display devices, dual ported memory locations, DMA target locations). The use of this ability is common enough to warrant some degree of standardization of its interface. This ability overlaps the general paradigm of shared memory in that, in both instances, common global objects are made addressable by individual processes or tasks.

Finally, a number of systems also provide the ability to map process addresses to files. This provides both a general means of sharing persistent objects, and using files in a manner that optimizes memory and swapping space usage.

Simple shared memory is clearly a special case of the more general file mapping capability. In addition, there is relatively widespread agreement and implementation of the file mapping interface. In these systems, many different types of objects can be mapped (for example, files, memory, devices, and so on) using the same mapping interfaces. This approach both minimizes interface proliferation and maximizes the generality of programs using the mapping interfaces.

- Memory Mapped Files Usage

A memory object can be concurrently mapped into the address space of one or more processes. The *mmap()* and *munmap()* functions allow a process to manipulate their address space by mapping portions of memory objects into it and removing them from it. When multiple processes map the same memory object, they can share access to the underlying data. Implementations may restrict the size and alignment of mappings to be on *page*-size boundaries. The page size, in bytes, is the value of the system-configurable variable {PAGESIZE}, typically accessed by calling *sysconf()* with a *name* argument of *\_SC\_PAGESIZE*. If an implementation has no restrictions on size or alignment, it may specify a 1-byte page size.

To map memory, a process first opens a memory object. The *ftruncate()* function can be used to contract or extend the size of the memory object even when the object is currently mapped. If the memory object is extended, the contents of the extended areas are zeros.

120788 After opening a memory object, the application maps the object into its address space  
 120789 using the *mmap()* function call. Once a mapping has been established, it remains mapped  
 120790 until unmapped with *munmap()*, even if the memory object is closed. The *mprotect()*  
 120791 function can be used to change the memory protections initially established by *mmap()*.

120792 A *close()* of the file descriptor, while invalidating the file descriptor itself, does not unmap  
 120793 any mappings established for the memory object. The address space, including all mapped  
 120794 regions, is inherited on *fork()*. The entire address space is unmapped on process  
 120795 termination or by successful calls to any of the *exec* family of functions.

120796 The *msync()* function is used to force mapped file data to permanent storage.

120797 • Effects on Other Functions

120798 With memory mapped files, the operation of the *open()*, *creat()*, and *unlink()* functions are  
 120799 a natural result of using the file system name space to map the global names for memory  
 120800 objects.

120801 The *ftruncate()* function can be used to set the length of a sharable memory object.

120802 The meaning of *stat()* fields other than the size and protection information is undefined on  
 120803 implementations where memory objects are not implemented using regular files. When  
 120804 regular files are used, the times reflect when the implementation updated the file image of  
 120805 the data, not when a process updated the data in memory.

120806 The operations of *fdopen()*, *write()*, *read()*, and *lseek()* were made unspecified for objects  
 120807 opened with *shm\_open()*, so that implementations that did not implement memory objects  
 120808 as regular files would not have to support the operation of these functions on shared  
 120809 memory objects.

120810 The behavior of memory objects with respect to *close()*, *dup()*, *dup2()*, *open()*, *close()*,  
 120811 *fork()*, *\_exit()*, and the *exec* family of functions is the same as the behavior of the existing  
 120812 practice of the *mmap()* function.

120813 A memory object can still be referenced after a close. That is, any mappings made to the  
 120814 file are still in effect, and reads and writes that are made to those mappings are still valid  
 120815 and are shared with other processes that have the same mapping. Likewise, the memory  
 120816 object can still be used if any references remain after its name(s) have been deleted. Any  
 120817 references that remain after a close must not appear to the application as file descriptors.

120818 This is existing practice for *mmap()* and *close()*. In addition, there are already mappings  
 120819 present (text, data, stack) that do not have open file descriptors. The text mapping in  
 120820 particular is considered a reference to the file containing the text. The desire was to treat all  
 120821 mappings by the process uniformly. Also, many modern implementations use *mmap()* to  
 120822 implement shared libraries, and it would not be desirable to keep file descriptors for each  
 120823 of the many libraries an application can use. It was felt there were many other existing  
 120824 programs that used this behavior to free a file descriptor, and thus POSIX.1-2008 could not  
 120825 forbid it and still claim to be using existing practice.

120826 For implementations that implement memory objects using memory only, memory objects  
 120827 will retain the memory allocated to the file after the last close and will use that same  
 120828 memory on the next open. Note that closing the memory object is not the same as deleting  
 120829 the name, since the memory object is still defined in the memory object name space.

120830 The locks of *fcntl()* do not block any read or write operation, including read or write access  
 120831 to shared memory or mapped files. In addition, implementations that only support shared  
 120832 memory objects should not be required to implement record locks. The reference to *fcntl()*  
 120833 is added to make this point explicitly. The other *fcntl()* commands are useful with shared  
 120834 memory objects.

120835 The size of pages that mapping hardware may be able to support may be a configurable  
 120836 value, or it may change based on hardware implementations. The addition of the  
 120837 `_SC_PAGESIZE` parameter to the `sysconf()` function is provided for determining the  
 120838 mapping page size at runtime.

## 120839 Shared Memory Functions

120840 Implementations may support the Shared Memory Objects option independently of memory  
 120841 mapped files. Shared memory objects are named regions of storage that may be independent of  
 120842 the file system and can be mapped into the address space of one or more processes to allow  
 120843 them to share the associated memory.

### 120844 • Requirements

120845 Shared memory is used to share data among several processes, each potentially running at  
 120846 different priority levels, responding to different inputs, or performing separate tasks.  
 120847 Shared memory is not just simply providing common access to data, it is providing the  
 120848 fastest possible communication between the processes. With one memory write operation,  
 120849 a process can pass information to as many processes as have the memory region mapped.

120850 As a result, shared memory provides a mechanism that can be used for all other  
 120851 interprocess communication facilities. It may also be used by an application for  
 120852 implementing more sophisticated mechanisms than semaphores and message queues.

120853 The need for a shared memory interface is obvious for virtual memory systems, where the  
 120854 operating system is directly preventing processes from accessing each other's data.  
 120855 However, in unprotected systems, such as those found in some embedded controllers, a  
 120856 shared memory interface is needed to provide a portable mechanism to allocate a region of  
 120857 memory to be shared and then to communicate the address of that region to other  
 120858 processes.

120859 This, then, provides the minimum functionality that a shared memory interface must have  
 120860 in order to support realtime applications: to allocate and name an object to be mapped into  
 120861 memory for potential sharing (`open()` or `shm_open()`), and to make the memory object  
 120862 available within the address space of a process (`mmap()`). To complete the interface, a  
 120863 mechanism to release the claim of a process on a shared memory object (`munmap()`) is also  
 120864 needed, as well as a mechanism for deleting the name of a sharable object that was  
 120865 previously created (`unlink()` or `shm_unlink()`).

120866 After a mapping has been established, an implementation should not have to provide  
 120867 services to maintain that mapping. All memory writes into that area will appear  
 120868 immediately in the memory mapping of that region by any other processes.

120869 Thus, requirements include:

- 120870 — Support creation of sharable memory objects and the mapping of these objects into  
 120871 the address space of a process.
- 120872 — Sharable memory objects should be accessed by global names accessible from all  
 120873 processes.
- 120874 — Support the mapping of specific sections of physical address space (such as a  
 120875 memory mapped device) into the address space of a process. This should not be  
 120876 done by the process specifying the actual address, but again by an implementation-  
 120877 defined global name (such as a special device name) dedicated to this purpose.
- 120878 — Support the mapping of discrete portions of these memory objects.



- 120879 — Support for minimum hardware configurations that contain no physical media on  
120880 which to store shared memory contents permanently.
- 120881 — The ability to preallocate the entire shared memory region so that minimum  
120882 hardware configurations without virtual memory support can guarantee contiguous  
120883 space.
- 120884 — The maximizing of performance by not requiring functionality that would require  
120885 implementation interaction above creating the shared memory area and returning  
120886 the mapping.

120887 Note that the above requirements do not preclude:

- 120888 — The sharable memory object from being implemented using actual files on an actual  
120889 file system.
- 120890 — The global name that is accessible from all processes being restricted to a file system  
120891 area that is dedicated to handling shared memory.
- 120892 — An implementation not providing implementation-defined global names for the  
120893 purpose of physical address mapping.

#### 120894 • Shared Memory Objects Usage

120895 If the Shared Memory Objects option is supported, a shared memory object may be  
120896 created, or opened if it already exists, with the *shm\_open()* function. If the shared memory  
120897 object is created, it has a length of zero. The *ftruncate()* function can be used to set the size  
120898 of the shared memory object after creation. The *shm\_unlink()* function removes the name  
120899 for a shared memory object created by *shm\_open()*.

#### 120900 • Shared Memory Overview

120901 The shared memory facility defined by POSIX.1-2008 usually results in memory locations  
120902 being added to the address space of the process. The implementation returns the address  
120903 of the new space to the application by means of a pointer. This works well in languages  
120904 like C. However, in languages without pointer types it will not work. In the bindings for  
120905 such a language, either a special COMMON section will need to be defined (which is  
120906 unlikely), or the binding will have to allow existing structures to be mapped. The  
120907 implementation will likely have to place restrictions on the size and alignment of such  
120908 structures or will have to map a suitable region of the address space of the process into the  
120909 memory object, and thus into other processes. These are issues for that particular language  
120910 binding. For POSIX.1-2008, however, the practice will not be forbidden, merely undefined.

120911 Two potentially different name spaces are used for naming objects that may be mapped  
120912 into process address spaces. When using memory mapped files, files may be accessed via  
120913 *open()*. When the Shared Memory Objects option is supported, sharable memory objects  
120914 that might not be files may be accessed via the *shm\_open()* function. These operations are  
120915 not mutually-exclusive.

120916 Some implementations supporting the Shared Memory Objects option may choose to  
120917 implement the shared memory object name space as part of the file system name space.  
120918 There are several reasons for this:

- 120919 — It allows applications to prevent name conflicts by use of the directory structure.
- 120920 — It uses an existing mechanism for accessing global objects and prevents the creation  
120921 of a new mechanism for naming global objects.

120922 In such implementations, memory objects can be implemented using regular files, if that is  
120923 what the implementation chooses. The *shm\_open()* function can be implemented as an

120924 *open()* call in a fixed directory with the `O_CLOEXEC` flag set. The *shm\_unlink()* function  
120925 can be implemented as an *unlink()* call.

120926 On the other hand, it is also expected that small embedded systems that support the  
120927 Shared Memory Objects option may wish to implement shared memory without having  
120928 any file systems present. In this case, the implementations may choose to use a simple  
120929 string valued name space for shared memory regions. The *shm\_open()* function permits  
120930 either type of implementation.

120931 Some implementations have hardware that supports protection of mapped data from  
120932 certain classes of access and some do not. Systems that supply this functionality support  
120933 the memory protection functionality.

120934 Some implementations restrict size, alignment, and protections to be on *page*-size  
120935 boundaries. If an implementation has no restrictions on size or alignment, it may specify a  
120936 1-byte page size. Applications on implementations that do support larger pages must be  
120937 cognizant of the page size since this is the alignment and protection boundary.

120938 Simple embedded implementations may have a 1-byte page size and only support the  
120939 Shared Memory Objects option. This provides simple shared memory between processes  
120940 without requiring mapping hardware.

120941 POSIX.1-2008 specifically allows a memory object to remain referenced after a close  
120942 because that is existing practice for the *mmap()* function.

## 120943 **Typed Memory Functions**

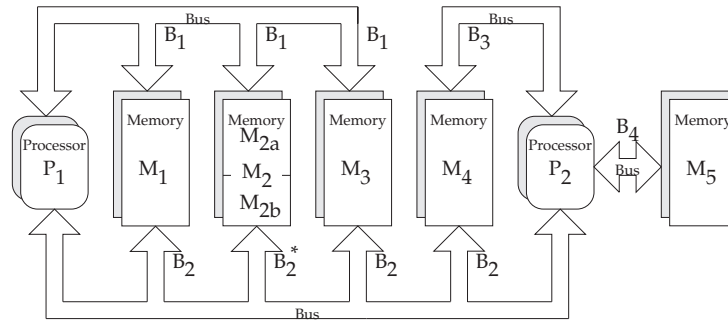
120944 Implementations may support the Typed Memory Objects option without supporting either the  
120945 Shared Memory option or memory mapped files. Types memory objects are pools of specialized  
120946 storage, different from the main memory resource normally used by a processor to hold code  
120947 and data, that can be mapped into the address space of one or more processes.

### 120948 • Model

120949 Realtime systems conforming to one of the POSIX.13 realtime profiles are expected (and  
120950 desired) to be supported on systems with more than one type or pool of memory (for  
120951 example, SRAM, DRAM, ROM, EPROM, EEPROM), where each type or pool of memory  
120952 may be accessible by one or more processors via one or more buses (ports). Memory  
120953 mapped files, shared memory objects, and the language-specific storage allocation  
120954 operators (*malloc()* for the ISO C standard, *new* for ISO Ada) fail to provide application  
120955 program interfaces versatile enough to allow applications to control their utilization of  
120956 such diverse memory resources. The typed memory interfaces *posix\_typed\_mem\_open()*,  
120957 *posix\_mem\_offset()*, *posix\_typed\_mem\_get\_info()*, *mmap()*, and *munmap()* defined herein  
120958 support the model of typed memory described below.

120959 For purposes of this model, a system comprises several processors (for example,  $P_1$  and  
120960  $P_2$ ), several physical memory pools (for example,  $M_1$ ,  $M_2$ ,  $M_{2a}$ ,  $M_{2b}$ ,  $M_3$ ,  $M_4$ , and  $M_5$ ), and  
120961 several buses or “ports” (for example,  $B_1$ ,  $B_2$ ,  $B_3$ , and  $B_4$ ) interconnecting the various  
120962 processors and memory pools in some system-specific way. Notice that some memory  
120963 pools may be contained in others (for example,  $M_{2a}$  and  $M_{2b}$  are contained in  $M_2$ ).

120964 **Figure B-1** (on page 3565) shows an example of such a model. In a system like this, an  
120965 application should be able to perform the following operations:



\* All addresses in pool M<sub>2</sub> (comprising pools M<sub>2a</sub> and M<sub>2b</sub>) accessible via port B<sub>1</sub>.  
 Addresses in pool M<sub>2b</sub> are also accessible via port B<sub>2</sub>.  
 Addresses in pool M<sub>2a</sub> are *not* accessible via port B<sub>2</sub>.

**Figure B-1** Example of a System with Typed Memory

#### — Typed Memory Allocation

An application should be able to allocate memory dynamically from the desired pool using the desired bus, and map it into the address space of a process. For example, processor P<sub>1</sub> can allocate some portion of memory pool M<sub>1</sub> through port B<sub>1</sub>, treating all unmapped subareas of M<sub>1</sub> as a heap-storage resource from which memory may be allocated. This portion of memory is mapped into address space of the process, and subsequently deallocated when unmapped from all processes.

#### — Using the Same Storage Region from Different Buses

An application process with a mapped region of storage that is accessed from one bus should be able to map that same storage area at another address (subject to page size restrictions detailed in *mmap()*), to allow it to be accessed from another bus. For example, processor P<sub>1</sub> may wish to access the same region of memory pool M<sub>2b</sub> both through ports B<sub>1</sub> and B<sub>2</sub>.

#### — Sharing Typed Memory Regions

Several application processes running on the same or different processors may wish to share a particular region of a typed memory pool. Each process or processor may wish to access this region through different buses. For example, processor P<sub>1</sub> may want to share a region of memory pool M<sub>4</sub> with processor P<sub>2</sub>, and they may be required to use buses B<sub>2</sub> and B<sub>3</sub>, respectively, to minimize bus contention. A problem arises here when a process allocates and maps a portion of fragmented memory and then wants to share this region of memory with another process, either in the same processor or different processors. The solution adopted is to allow the first process to find out the memory map (offsets and lengths) of all the different fragments of memory that were mapped into its address space, by repeatedly calling *posix\_mem\_offset()*. Then, this process can pass the offsets and lengths obtained to the second process, which can then map the same memory fragments into its address space.

#### — Contiguous Allocation

The problem of finding the memory map of the different fragments of the memory pool that were mapped into logically contiguous addresses of a given process can be

solved by requesting contiguous allocation. For example, a process in  $P_1$  can allocate 10 Kbytes of physically contiguous memory from  $M_3-B_1$ , and obtain the offset (within pool  $M_3$ ) of this block of memory. Then, it can pass this offset (and the length) to a process in  $P_2$  using some interprocess communication mechanism. The second process can map the same block of memory by using the offset transferred and specifying  $M_3-B_2$ .

#### 121003 — Unallocated Mapping

Any subarea of a memory pool that is mapped to a process, either as the result of an allocation request or an explicit mapping, is normally unavailable for allocation. Special processes such as debuggers, however, may need to map large areas of a typed memory pool, yet leave those areas available for allocation.

Typed memory allocation and mapping has to coexist with storage allocation operators like *malloc()*, but systems are free to choose how to implement this coexistence. For example, it may be system configuration-dependent if all available system memory is made part of one of the typed memory pools or if some part will be restricted to conventional allocation operators. Equally system configuration-dependent may be the availability of operators like *malloc()* to allocate storage from certain typed memory pools. It is not excluded to configure a system such that a given named pool,  $P_1$ , is in turn split into non-overlapping named subpools. For example,  $M_1-B_1$ ,  $M_2-B_1$ , and  $M_3-B_1$  could also be accessed as one common pool  $M_{123}-B_1$ . A call to *malloc()* on  $P_1$  could work on such a larger pool while full optimization of memory usage by  $P_1$  would require typed memory allocation at the subpool level.

#### 121019 • Existing Practice

OS-9 provides for the naming (numbering) and prioritization of memory types by a system administrator. It then provides APIs to request memory allocation of typed (colored) memory by number, and to generate a bus address from a mapped memory address (translate). When requesting colored memory, the user can specify type 0 to signify allocation from the first available type in priority order.

HP-RT presents interfaces to map different kinds of storage regions that are visible through a VME bus, although it does not provide allocation operations. It also provides functions to perform address translation between VME addresses and virtual addresses. It represents a VME-bus unique solution to the general problem.

The PSOS approach is similar (that is, based on a pre-established mapping of bus address ranges to specific memories) with a concept of segments and regions (regions dynamically allocated from a heap which is a special segment). Therefore, PSOS does not fully address the general allocation problem either. PSOS does not have a “process”-based model, but more of a “thread”-only-based model of multi-tasking. So mapping to a process address space is not an issue.

QNX uses the System V approach of opening specially named devices (shared memory segments) and using *mmap()* to then gain access from the process. They do not address allocation directly, but once typed shared memory can be mapped, an “allocation manager” process could be written to handle requests for allocation.

The System V approach also included allocation, implemented by opening yet other special “devices” which allocate, rather than appearing as a whole memory object.

The Orkid realtime kernel interface definition has operations to manage memory “regions” and “pools”, which are areas of memory that may reflect the differing physical nature of the memory. Operations to allocate memory from these regions and pools are also provided.

- Requirements

Existing practice in SVID-derived UNIX systems relies on functionality similar to *mmap()* and its related interfaces to achieve mapping and allocation of typed memory. However, the issue of sharing typed memory (allocated or mapped) and the complication of multiple ports are not addressed in any consistent way by existing UNIX system practice. Part of this functionality is existing practice in specialized realtime operating systems. In order to solidify the capabilities implied by the model above, the following requirements are imposed on the interface:

- Identification of Typed Memory Pools and Ports

All processes (running in all processors) in the system are able to identify a particular (system configured) typed memory pool accessed through a particular (system configured) port by a name. That name is a member of a name space common to all these processes, but need not be the same name space as that containing ordinary pathnames. The association between memory pools/ports and corresponding names is typically established when the system is configured. The “open” operation for typed memory objects should be distinct from the *open()* function, for consistency with other similar services, but implementable on top of *open()*. This implies that the handle for a typed memory object will be a file descriptor.

- Allocation and Mapping of Typed Memory

Once a typed memory object has been identified by a process, it is possible to both map user-selected subareas of that object into process address space and to map system-selected (that is, dynamically allocated) subareas of that object, with user-specified length, into process address space. It is also possible to determine the maximum length of memory allocation that may be requested from a given typed memory object.

- Sharing Typed Memory

Two or more processes are able to share portions of typed memory, either user-selected or dynamically allocated. This requirement applies also to dynamically allocated regions of memory that are composed of several non-contiguous pieces.

- Contiguous Allocation

For dynamic allocation, it is the user’s option whether the system is required to allocate a contiguous subarea within the typed memory object, or whether it is permitted to allocate discontinuous fragments which appear contiguous in the process mapping. Contiguous allocation simplifies the process of sharing allocated typed memory, while discontinuous allocation allows for potentially better recovery of deallocated typed memory.

- Accessing Typed Memory Through Different Ports

Once a subarea of a typed memory object has been mapped, it is possible to determine the location and length corresponding to a user-selected portion of that object within the memory pool. This location and length can then be used to remap that portion of memory for access from another port. If the referenced portion of typed memory was allocated discontinuously, the length thus determined may be shorter than anticipated, and the user code must adapt to the value returned.

- Deallocation

When a previously mapped subarea of typed memory is no longer mapped by any process in the system—as a result of a call or calls to *munmap()*—that subarea

121091 becomes potentially reusable for dynamic allocation; actual reuse of the subarea is a  
 121092 function of the dynamic typed memory allocation policy.

121093 — Unallocated Mapping

121094 It must be possible to map user-selected subareas of a typed memory object without  
 121095 marking that subarea as unavailable for allocation. This option is not the default  
 121096 behavior, and requires appropriate privileges.

121097 • Scenario

121098 The following scenario will serve to clarify the use of the typed memory interfaces.

121099 Process A running on  $P_1$  (see Figure B-1, on page 3565) wants to allocate some memory  
 121100 from memory pool  $M_2$ , and it wants to share this portion of memory with process B  
 121101 running on  $P_2$ . Since  $P_2$  only has access to the lower part of  $M_2$ , both processes will use the  
 121102 memory pool named  $M_{2b}$  which is the part of  $M_2$  that is accessible both from  $P_1$  and  $P_2$ . The  
 121103 operations that both processes need to perform are shown below:

121104 — Allocating Typed Memory

121105 Process A calls *posix\_typed\_mem\_open()* with the name **/typed.m2b-b1** and a *tflag* of  
 121106 POSIX\_TYPED\_MEM\_ALLOCATE to get a file descriptor usable for allocating from  
 121107 pool  $M_{2b}$  accessed through port  $B_1$ . It then calls *mmap()* with this file descriptor  
 121108 requesting a length of 4096 bytes. The system allocates two discontinuous blocks of  
 121109 sizes 1024 and 3072 bytes within  $M_{2b}$ . The *mmap()* function returns a pointer to a  
 121110 4096-byte array in process A's logical address space, mapping the allocated blocks  
 121111 contiguously. Process A can then utilize the array, and store data in it.

121112 — Determining the Location of the Allocated Blocks

121113 Process A can determine the lengths and offsets (relative to  $M_{2b}$ ) of the two blocks  
 121114 allocated, by using the following procedure: First, process A calls *posix\_mem\_offset()*  
 121115 with the address of the first element of the array and length 4096. Upon return, the  
 121116 offset and length (1024 bytes) of the first block are returned. A second call to  
 121117 *posix\_mem\_offset()* is then made using the address of the first element of the array  
 121118 plus 1024 (the length of the first block), and a new length of 4096–1024. If there were  
 121119 more fragments allocated, this procedure could have been continued within a loop  
 121120 until the offsets and lengths of all the blocks were obtained. Notice that this relatively  
 121121 complex procedure can be avoided if contiguous allocation is requested (by opening  
 121122 the typed memory object with the *tflag*  
 121123 POSIX\_TYPED\_MEM\_ALLOCATE\_CONTIG).

121124 — Sharing Data Across Processes

121125 Process A passes the two offset values and lengths obtained from the  
 121126 *posix\_mem\_offset()* calls to process B running on  $P_2$ , via some form of interprocess  
 121127 communication. Process B can gain access to process A's data by calling  
 121128 *posix\_typed\_mem\_open()* with the name **/typed.m2b-b2** and a *tflag* of zero, then using  
 121129 two *mmap()* calls on the resulting file descriptor to map the two subareas of that  
 121130 typed memory object to its own address space.

121131 • Rationale for no *mem\_alloc()* and *mem\_free()*

121132 The standard developers had originally proposed a pair of new flags to *mmap()* which,  
 121133 when applied to a typed memory object descriptor, would cause *mmap()* to allocate  
 121134 dynamically from an unallocated and unmapped area of the typed memory object.  
 121135 Deallocation was similarly accomplished through the use of *munmap()*. This was rejected  
 121136 by the ballot group because it excessively complicated the (already rather complex)

*mmap()* interface and introduced semantics useful only for typed memory, to a function which must also map shared memory and files. They felt that a memory allocator should be built on top of *mmap()* instead of being incorporated within the same interface, much as the ISO C standard libraries build *malloc()* on top of the virtual memory mapping functions *brk()* and *sbrk()*. This would eliminate the complicated semantics involved with unmapping only part of an allocated block of typed memory.

To attempt to achieve ballot group consensus, typed memory allocation and deallocation was first migrated from *mmap()* and *munmap()* to a pair of complementary functions modeled on the ISO C standard *malloc()* and *free()*. The *mem\_alloc()* function specified explicitly the typed memory object (typed memory pool/access port) from which allocation takes place, unlike *malloc()* where the memory pool and port are unspecified. The *mem\_free()* function handled deallocation. These new semantics still met all of the requirements detailed above without modifying the behavior of *mmap()* except to allow it to map specified areas of typed memory objects. An implementation would have been free to implement *mem\_alloc()* and *mem\_free()* over *mmap()*, through *mmap()*, or independently but cooperating with *mmap()*.

The ballot group was queried to see if this was an acceptable alternative, and while there was some agreement that it achieved the goal of removing the complicated semantics of allocation from the *mmap()* interface, several balloters realized that it just created two additional functions that behaved, in great part, like *mmap()*. These balloters proposed an alternative which has been implemented here in place of a separate *mem\_alloc()* and *mem\_free()*. This alternative is based on four specific suggestions:

1. The *posix\_typed\_mem\_open()* function should provide a flag which specifies "allocate on *mmap()*" (otherwise, *mmap()* just maps the underlying object). This allows things roughly similar to */dev/zero* versus */dev/swap*. Two such flags have been implemented, one of which forces contiguous allocation.
2. The *posix\_mem\_offset()* function is acceptable because it can be applied usefully to mapped objects in general. It should return the file descriptor of the underlying object.
3. The *mem\_get\_info()* function in an earlier draft should be renamed *posix\_typed\_mem\_get\_info()* because it is not generally applicable to memory objects. It should probably return the file descriptor's allocation attribute. The renaming of the function has been implemented, but having it return a piece of information which is readily known by an application without this function has been rejected. Its whole purpose is to query the typed memory object for attributes that are not user-specified, but determined by the implementation.
4. There should be no separate *mem\_alloc()* or *mem\_free()* functions. Instead, using *mmap()* on a typed memory object opened with an "allocate on *mmap()*" flag should be used to force allocation. These are precisely the semantics defined in the current draft.

- Rationale for no Typed Memory Access Management

The working group had originally defined an additional interface (and an additional kind of object: typed memory master) to establish and dissolve mappings to typed memory on behalf of devices or processors which were independent of the operating system and had no inherent capability to directly establish mappings on their own. This was to have provided functionality similar to device driver interfaces such as *physio()* and their underlying bus-specific interfaces (for example, *mballoc()*) which serve to set up and break down DMA pathways, and derive mapped addresses for use by hardware devices and processor cards.

121186 The ballot group felt that this was beyond the scope of POSIX.1 and its amendments.  
 121187 Furthermore, the removal of interrupt handling interfaces from a preceding amendment  
 121188 (the IEEE Std 1003.1d-1999) during its balloting process renders these typed memory  
 121189 access management interfaces an incomplete solution to portable device management from  
 121190 a user process; it would be possible to initiate a device transfer to/from typed memory, but  
 121191 impossible to handle the transfer-complete interrupt in a portable way.

121192 To achieve ballot group consensus, all references to typed memory access management  
 121193 capabilities were removed. The concept of portable interfaces from a device driver to both  
 121194 operating system and hardware is being addressed by the Uniform Driver Interface (UDI)  
 121195 industry forum, with formal standardization deferred until proof of concept and industry-  
 121196 wide acceptance and implementation.

#### 121197 B.2.8.4 Process Scheduling

121198 IEEE PASC Interpretation 1003.1 #96 has been applied, adding the *pthread\_setschedprio()*  
 121199 function. This was added since previously there was no way for a thread to lower its own  
 121200 priority without going to the tail of the threads list for its new priority. This capability is  
 121201 necessary to bound the duration of priority inversion encountered by a thread.

121202 The following portion of the rationale presents models, requirements, and standardization  
 121203 issues relevant to process scheduling; see also [Section B.2.9.4](#) (on page 3610).

121204 In an operating system supporting multiple concurrent processes, the system determines the  
 121205 order in which processes execute to meet implementation-defined goals. For time-sharing  
 121206 systems, the goal is to enhance system throughput and promote fairness; the application is  
 121207 provided with little or no control over this sequencing function. While this is acceptable and  
 121208 desirable behavior in a time-sharing system, it is inappropriate in a realtime system; realtime  
 121209 applications must specifically control the execution sequence of their concurrent processes in  
 121210 order to meet externally defined response requirements.

121211 In POSIX.1-2008, the control over process sequencing is provided using a concept of scheduling  
 121212 policies. These policies, described in detail in this section, define the behavior of the system  
 121213 whenever processor resources are to be allocated to competing processes. Only the behavior of  
 121214 the policy is defined; conforming implementations are free to use any mechanism desired to  
 121215 achieve the described behavior.

#### 121216 • Models

121217 In an operating system supporting multiple concurrent processes, the system determines  
 121218 the order in which processes execute and might force long-running processes to yield to  
 121219 other processes at certain intervals. Typically, the scheduling code is executed whenever an  
 121220 event occurs that might alter the process to be executed next.

121221 The simplest scheduling strategy is a “first-in, first-out” (FIFO) dispatcher. Whenever a  
 121222 process becomes runnable, it is placed on the end of a ready list. The process at the front of  
 121223 the ready list is executed until it exits or becomes blocked, at which point it is removed  
 121224 from the list. This scheduling technique is also known as “run-to-completion” or “run-to-  
 121225 block”.

121226 A natural extension to this scheduling technique is the assignment of a “non-migrating  
 121227 priority” to each process. This policy differs from strict FIFO scheduling in only one  
 121228 respect: whenever a process becomes runnable, it is placed at the end of the list of  
 121229 processes runnable at that priority level. When selecting a process to run, the system  
 121230 always selects the first process from the highest priority queue with a runnable process.  
 121231 Thus, when a process becomes unblocked, it will preempt a running process of lower  
 121232 priority without otherwise altering the ready list. Further, if a process elects to alter its



|        |                                                                                                           |
|--------|-----------------------------------------------------------------------------------------------------------|
| 121233 | priority, it is removed from the ready list and reinserted, using its new priority, according             |
| 121234 | to the policy above.                                                                                      |
| 121235 | While the above policy might be considered unfriendly in a time-sharing environment in                    |
| 121236 | which multiple users require more balanced resource allocation, it could be ideal in a                    |
| 121237 | realtime environment for several reasons. The most important of these is that it is                       |
| 121238 | deterministic: the highest-priority process is always run and, among processes of equal                   |
| 121239 | priority, the process that has been runnable for the longest time is executed first. Because of           |
| 121240 | this determinism, cooperating processes can implement more complex scheduling simply                      |
| 121241 | by altering their priority. For instance, if processes at a single priority were to reschedule            |
| 121242 | themselves at fixed time intervals, a time-slice policy would result.                                     |
| 121243 | In a dedicated operating system in which all processes are well-behaved realtime                          |
| 121244 | applications, non-migrating priority scheduling is sufficient. However, many existing                     |
| 121245 | implementations provide for more complex scheduling policies.                                             |
| 121246 | POSIX.1-2008 specifies a linear scheduling model. In this model, every process in the                     |
| 121247 | system has a priority. The system scheduler always dispatches a process that has the                      |
| 121248 | highest (generally the most time-critical) priority among all runnable processes in the                   |
| 121249 | system. As long as there is only one such process, the dispatching policy is trivial. When                |
| 121250 | multiple processes of equal priority are eligible to run, they are ordered according to a                 |
| 121251 | strict run-to-completion (FIFO) policy.                                                                   |
| 121252 | The priority is represented as a positive integer and is inherited from the parent process.               |
| 121253 | For processes running under a fixed priority scheduling policy, the priority is never altered             |
| 121254 | except by an explicit function call.                                                                      |
| 121255 | It was determined arbitrarily that larger integers correspond to “higher priorities”.                     |
| 121256 | Certain implementations might impose restrictions on the priority ranges to which                         |
| 121257 | processes can be assigned. There also can be restrictions on the set of policies to which                 |
| 121258 | processes can be set.                                                                                     |
| 121259 | • Requirements                                                                                            |
| 121260 | Realtime processes require that scheduling be fast and deterministic, and that it guarantees              |
| 121261 | to preempt lower priority processes.                                                                      |
| 121262 | Thus, given the linear scheduling model, realtime processes require that they be run at a                 |
| 121263 | priority that is higher than other processes. Within this framework, realtime processes are               |
| 121264 | free to yield execution resources to each other in a completely portable and                              |
| 121265 | implementation-defined manner.                                                                            |
| 121266 | As there is a generally perceived requirement for processes at the same priority level to                 |
| 121267 | share processor resources more equitably, provisions are made by providing a scheduling                   |
| 121268 | policy (that is, SCHED_RR) intended to provide a timeslice-like facility.                                 |
| 121269 | <b>Note:</b> The following topics assume that low numeric priority implies low scheduling criticality     |
| 121270 | and <i>vice versa</i> .                                                                                   |
| 121271 | • Rationale for New Interface                                                                             |
| 121272 | Realtime applications need to be able to determine when processes will run in relation to                 |
| 121273 | each other. It must be possible to guarantee that a critical process will run whenever it is              |
| 121274 | runnable; that is, whenever it wants to for as long as it needs. SCHED_FIFO satisfies this                |
| 121275 | requirement. Additionally, SCHED_RR was defined to meet a realtime requirement for a                      |
| 121276 | well-defined time-sharing policy for processes at the same priority.                                      |
| 121277 | It would be possible to use the BSD <i>setpriority()</i> and <i>getpriority()</i> functions by redefining |
| 121278 | the meaning of the “nice” parameter according to the scheduling policy currently in use by                |

|        |                                                                                                      |
|--------|------------------------------------------------------------------------------------------------------|
| 121279 | the process. The System V <i>nice()</i> interface was felt to be undesirable for realtime because it |
| 121280 | specifies an adjustment to the “nice” value, rather than setting it to an explicit value.            |
| 121281 | Realtime applications will usually want to set priority to an explicit value. Also, System V         |
| 121282 | <i>nice()</i> does not allow for changing the priority of another process.                           |
| 121283 | With the POSIX.1b interfaces, the traditional “nice” value does not affect the SCHED_FIFO            |
| 121284 | or SCHED_RR scheduling policies. If a “nice” value is supported, it is implementation-               |
| 121285 | defined whether it affects the SCHED_OTHER policy.                                                   |
| 121286 | An important aspect of POSIX.1-2008 is the explicit description of the queuing and                   |
| 121287 | preemption rules. It is critical, to achieve deterministic scheduling, that such rules be            |
| 121288 | stated clearly in POSIX.1-2008.                                                                      |
| 121289 | POSIX.1-2008 does not address the interaction between priority and swapping. The issues              |
| 121290 | involved with swapping and virtual memory paging are extremely implementation-                       |
| 121291 | defined and would be nearly impossible to standardize at this point. The proposed                    |
| 121292 | scheduling paradigm, however, fully describes the scheduling behavior of runnable                    |
| 121293 | processes, of which one criterion is that the working set be resident in memory. Assuming            |
| 121294 | the existence of a portable interface for locking portions of a process in memory, paging            |
| 121295 | behavior need not affect the scheduling of realtime processes.                                       |
| 121296 | POSIX.1-2008 also does not address the priorities of “system” processes. In general, these           |
| 121297 | processes should always execute in low-priority ranges to avoid conflict with other                  |
| 121298 | realtime processes. Implementations should document the priority ranges in which system              |
| 121299 | processes run.                                                                                       |
| 121300 | The default scheduling policy is not defined. The effect of I/O interrupts and other system          |
| 121301 | processing activities is not defined. The temporary lending of priority from one process to          |
| 121302 | another (such as for the purposes of affecting freeing resources) by the system is not               |
| 121303 | addressed. Preemption of resources is not addressed. Restrictions on the ability of a                |
| 121304 | process to affect other processes beyond a certain level (influence levels) is not addressed.        |
| 121305 | The rationale used to justify the simple time-quantum scheduler is that it is common                 |
| 121306 | practice to depend upon this type of scheduling to ensure “fair” distribution of processor           |
| 121307 | resources among portions of the application that must interoperate in a serial fashion. Note         |
| 121308 | that POSIX.1-2008 is silent with respect to the setting of this time quantum, or whether it is       |
| 121309 | a system-wide value or a per-process value, although it appears that the prevailing                  |
| 121310 | realtime practice is for it to be a system-wide value.                                               |
| 121311 | In a system with $N$ processes at a given priority, all processor-bound, in which the time           |
| 121312 | quantum is equal for all processes at a specific priority level, the following assumptions           |
| 121313 | are made of such a scheduling policy:                                                                |
| 121314 | 1. A time quantum $Q$ exists and the current process will own control of the processor               |
| 121315 | for at least a duration of $Q$ and will have the processor for a duration of $Q$ .                   |
| 121316 | 2. The $N$ th process at that priority will control a processor within a duration of $(N-1)$         |
| 121317 | $\times Q$ .                                                                                         |
| 121318 | These assumptions are necessary to provide equal access to the processor and bounded                 |
| 121319 | response from the application.                                                                       |
| 121320 | The assumptions hold for the described scheduling policy only if no system overhead,                 |
| 121321 | such as interrupt servicing, is present. If the interrupt servicing load is non-zero, then one       |
| 121322 | of the two assumptions becomes fallacious, based upon how $Q$ is measured by the system.             |
| 121323 | If $Q$ is measured by clock time, then the assumption that the process obtains a duration $Q$        |
| 121324 | processor time is false if interrupt overhead exists. Indeed, a scenario can be constructed          |
| 121325 | with $N$ processes in which a single process undergoes complete processor starvation if a            |

121326 peripheral device, such as an analog-to-digital converter, generates significant interrupt  
 121327 activity periodically with a period of  $N \times Q$ .

121328 If  $Q$  is measured as actual processor time, then the assumption that the  $N$ th process runs in  
 121329 within the duration  $(N-1) \times Q$  is false.

121330 It should be noted that SCHED\_FIFO suffers from interrupt-based delay as well. However,  
 121331 for SCHED\_FIFO, the implied response of the system is “as soon as possible”, so that the  
 121332 interrupt load for this case is a vendor selection and not a compliance issue.

121333 With this in mind, it is necessary either to complete the definition by including bounds on  
 121334 the interrupt load, or to modify the assumptions that can be made about the scheduling  
 121335 policy.

121336 Since the motivation of inclusion of the policy is common usage, and since current  
 121337 applications do not enjoy the luxury of bounded interrupt load, item (2) above is sufficient  
 121338 to express existing application needs and is less restrictive in the standard definition. No  
 121339 difference in interface is necessary.

121340 In an implementation in which the time quantum is equal for all processes at a specific  
 121341 priority, our assumptions can then be restated as:

- 121342 — A time quantum  $Q$  exists, and a processor-bound process will be rescheduled after a  
 121343 duration of, at most,  $Q$ . Time quantum  $Q$  may be defined in either wall clock time or  
 121344 execution time.
- 121345 — In general, the  $N$ th process of a priority level should wait no longer than  $(N-1) \times Q$   
 121346 time to execute, assuming no processes exist at higher priority levels.
- 121347 — No process should wait indefinitely.

121348 For implementations supporting per-process time quanta, these assumptions can be  
 121349 readily extended.

### 121350 **Sporadic Server Scheduling Policy**

121351 The sporadic server is a mechanism defined for scheduling aperiodic activities in time-critical  
 121352 realtime systems. This mechanism reserves a certain bounded amount of execution capacity for  
 121353 processing aperiodic events at a high priority level. Any aperiodic events that cannot be  
 121354 processed within the bounded amount of execution capacity are executed in the background at a  
 121355 low priority level. Thus, a certain amount of execution capacity can be guaranteed to be  
 121356 available for processing periodic tasks, even under burst conditions in the arrival of aperiodic  
 121357 processing requests (that is, a large number of requests in a short time interval). The sporadic  
 121358 server also simplifies the schedulability analysis of the realtime system, because it allows  
 121359 aperiodic processes or threads to be treated as if they were periodic. The sporadic server was  
 121360 first described by Sprunt, et al.

121361 The key concept of the sporadic server is to provide and limit a certain amount of computation  
 121362 capacity for processing aperiodic events at their assigned normal priority, during a time interval  
 121363 called the “replenishment period”. Once the entity controlled by the sporadic server mechanism  
 121364 is initialized with its period and execution-time budget attributes, it preserves its execution  
 121365 capacity until an aperiodic request arrives. The request will be serviced (if there are no higher  
 121366 priority activities pending) as long as there is execution capacity left. If the request is completed,  
 121367 the actual execution time used to service it is subtracted from the capacity, and a replenishment  
 121368 of this amount of execution time is scheduled to happen one replenishment period after the  
 121369 arrival of the aperiodic request. If the request is not completed, because there is no execution  
 121370 capacity left, then the aperiodic process or thread is assigned a lower background priority. For  
 121371 each portion of consumed execution capacity the execution time used is replenished after one

replenishment period. At the time of replenishment, if the sporadic server was executing at a background priority level, its priority is elevated to the normal level. Other similar replenishment policies have been defined, but the one presented here represents a compromise between efficiency and implementation complexity.

The interface that appears in this section defines a new scheduling policy for threads and processes that behaves according to the rules of the sporadic server mechanism. Scheduling attributes are defined and functions are provided to allow the user to set and get the parameters that control the scheduling behavior of this mechanism, namely the normal and low priority, the replenishment period, the maximum number of pending replenishment operations, and the initial execution-time budget.

- Scheduling Aperiodic Activities

Virtually all realtime applications are required to process aperiodic activities. In many cases, there are tight timing constraints that the response to the aperiodic events must meet. Usual timing requirements imposed on the response to these events are:

- The effects of an aperiodic activity on the response time of lower priority activities must be controllable and predictable.
- The system must provide the fastest possible response time to aperiodic events.
- It must be possible to take advantage of all the available processing bandwidth not needed by time-critical activities to enhance average-case response times to aperiodic events.

Traditional methods for scheduling aperiodic activities are background processing, polling tasks, and direct event execution:

- Background processing consists of assigning a very low priority to the processing of aperiodic events. It utilizes all the available bandwidth in the system that has not been consumed by higher priority threads. However, it is very difficult, or impossible, to meet requirements on average-case response time, because the aperiodic entity has to wait for the execution of all other entities which have higher priority.
- Polling consists of creating a periodic process or thread for servicing aperiodic requests. At regular intervals, the polling entity is started and its services accumulated pending aperiodic requests. If no aperiodic requests are pending, the polling entity suspends itself until its next period. Polling allows the aperiodic requests to be processed at a higher priority level. However, worst and average-case response times of polling entities are a direct function of the polling period, and there is execution overhead for each polling period, even if no event has arrived. If the deadline of the aperiodic activity is short compared to the inter-arrival time, the polling frequency must be increased to guarantee meeting the deadline. For this case, the increase in frequency can dramatically reduce the efficiency of the system and, therefore, its capacity to meet all deadlines. Yet, polling represents a good way to handle a large class of practical problems because it preserves system predictability, and because the amortized overhead drops as load increases.
- Direct event execution consists of executing the aperiodic events at a high fixed-priority level. Typically, the aperiodic event is processed by an interrupt service routine as soon as it arrives. This technique provides predictable response times for aperiodic events, but makes the response times of all lower priority activities completely unpredictable under burst arrival conditions. Therefore, if the density of aperiodic event arrivals is unbounded, it may be a dangerous technique for time-critical systems. Yet, for those cases in which the physics of the system imposes a

bound on the event arrival rate, it is probably the most efficient technique.

- The sporadic server scheduling algorithm combines the predictability of the polling approach with the short response times of the direct event execution. Thus, it allows systems to meet an important class of application requirements that cannot be met by using the traditional approaches. Multiple sporadic servers with different attributes can be applied to the scheduling of multiple classes of aperiodic events, each with different kinds of timing requirements, such as individual deadlines, average response times, and so on. It also has many other interesting applications for realtime, such as scheduling producer/consumer tasks in time-critical systems, limiting the effects of faults on the estimation of task execution-time requirements, and so on.

- Existing Practice

The sporadic server has been used in different kinds of applications, including military avionics, robot control systems, industrial automation systems, and so on. There are examples of many systems that cannot be successfully scheduled using the classic approaches, such as direct event execution, or polling, and are schedulable using a sporadic server scheduler. The sporadic server algorithm itself can successfully schedule all systems scheduled with direct event execution or polling.

The sporadic server scheduling policy has been implemented as a commercial product in the run-time system of the Verdex Ada compiler. There are also many applications that have used a much less efficient application-level sporadic server. These realtime applications would benefit from a sporadic server scheduler implemented at the scheduler level.

- Library-Level *versus* Kernel-Level Implementation

The sporadic server interface described in this section requires the sporadic server policy to be implemented at the same level as the scheduler. This means that the process sporadic server must be implemented at the kernel level and the thread sporadic server policy implemented at the same level as the thread scheduler; that is, kernel or library level.

In an earlier interface for the sporadic server, this mechanism was implementable at a different level than the scheduler. This feature allowed the implementor to choose between an efficient scheduler-level implementation, or a simpler user or library-level implementation. However, the working group considered that this interface made the use of sporadic servers more complex, and that library-level implementations would lack some of the important functionality of the sporadic server, namely the limitation of the actual execution time of aperiodic activities. The working group also felt that the interface described in this chapter does not preclude library-level implementations of threads intended to provide efficient low-overhead scheduling for those threads that are not scheduled under the sporadic server policy.

- Range of Scheduling Priorities

Each of the scheduling policies supported in POSIX.1-2008 has an associated range of priorities. The priority ranges for each policy might or might not overlap with the priority ranges of other policies. For time-critical realtime applications it is usual for periodic and aperiodic activities to be scheduled together in the same processor. Periodic activities will usually be scheduled using the SCHED\_FIFO scheduling policy, while aperiodic activities may be scheduled using SCHED\_SPORADIC. Since the application developer will require complete control over the relative priorities of these activities in order to meet his timing requirements, it would be desirable for the priority ranges of SCHED\_FIFO and SCHED\_SPORADIC to overlap completely. Therefore, although POSIX.1-2008 does not

121468 require any particular relationship between the different priority ranges, it is  
121469 recommended that these two ranges should coincide.

121470 • Dynamically Setting the Sporadic Server Policy

121471 Several members of the working group requested that implementations should not be  
121472 required to support dynamically setting the sporadic server scheduling policy for a thread.  
121473 The reason is that this policy may have a high overhead for library-level implementations  
121474 of threads, and if threads are allowed to dynamically set this policy, this overhead can be  
121475 experienced even if the thread does not use that policy. By disallowing the dynamic setting  
121476 of the sporadic server scheduling policy, these implementations can accomplish efficient  
121477 scheduling for threads using other policies. If a strictly conforming application needs to  
121478 use the sporadic server policy, and is therefore willing to pay the overhead, it must set this  
121479 policy at the time of thread creation.

121480 • Limitation of the Number of Pending Replenishments

121481 The number of simultaneously pending replenishment operations must be limited for each  
121482 sporadic server for two reasons: an unlimited number of replenishment operations would  
121483 need an unlimited number of system resources to store all the pending replenishment  
121484 operations; on the other hand, in some implementations each replenishment operation will  
121485 represent a source of priority inversion (just for the duration of the replenishment  
121486 operation) and thus, the maximum amount of replenishments must be bounded to  
121487 guarantee bounded response times. The way in which the number of replenishments is  
121488 bounded is by lowering the priority of the sporadic server to *sched\_ss\_low\_priority* when  
121489 the number of pending replenishments has reached its limit. In this way, no new  
121490 replenishments are scheduled until the number of pending replenishments decreases.

121491 In the sporadic server scheduling policy defined in POSIX.1-2008, the application can  
121492 specify the maximum number of pending replenishment operations for a single sporadic  
121493 server, by setting the value of the *sched\_ss\_max\_repl* scheduling parameter. This value must  
121494 be between one and {SS\_REPL\_MAX}, which is a maximum limit imposed by the  
121495 implementation. The limit {SS\_REPL\_MAX} must be greater than or equal to  
121496 {\_POSIX\_SS\_REPL\_MAX}, which is defined to be four in POSIX.1-2008. The minimum  
121497 limit of four was chosen so that an application can at least guarantee that four different  
121498 aperiodic events can be processed during each interval of length equal to the  
121499 replenishment period.

121500 B.2.8.5 Clocks and Timers

121501 • Clocks

121502 POSIX.1-2008 and the ISO C standard both define functions for obtaining system time.  
121503 Implicit behind these functions is a mechanism for measuring passage of time. This  
121504 specification makes this mechanism explicit and calls it a clock. The CLOCK\_REALTIME  
121505 clock required by POSIX.1-2008 is a higher resolution version of the clock that maintains  
121506 POSIX.1 system time. This is a “system-wide” clock, in that it is visible to all processes  
121507 and, were it possible for multiple processes to all read the clock at the same time, they  
121508 would see the same value.

121509 An extensible interface was defined, with the ability for implementations to define  
121510 additional clocks. This was done because of the observation that many realtime platforms  
121511 support multiple clocks, and it was desired to fit this model within the standard interface.  
121512 But implementation-defined clocks need not represent actual hardware devices, nor are  
121513 they necessarily system-wide.

- Timers

Two timer types are required for a system to support realtime applications:

1. One-shot

A one-shot timer is a timer that is armed with an initial expiration time, either relative to the current time or at an absolute time (based on some timing base, such as time in seconds and nanoseconds since the Epoch). The timer expires once and then is disarmed. With the specified facilities, this is accomplished by setting the *it\_value* member of the *value* argument to the desired expiration time and the *it\_interval* member to zero.

2. Periodic

A periodic timer is a timer that is armed with an initial expiration time, again either relative or absolute, and a repetition interval. When the initial expiration occurs, the timer is reloaded with the repetition interval and continues counting. With the specified facilities, this is accomplished by setting the *it\_value* member of the *value* argument to the desired initial expiration time and the *it\_interval* member to the desired repetition interval.

For both of these types of timers, the time of the initial timer expiration can be specified in two ways:

1. Relative (to the current time)
2. Absolute

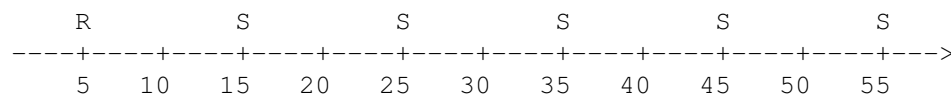
- Examples of Using Realtime Timers

In the diagrams below, *S* indicates a program schedule, *R* shows a schedule method request, and *E* suggests an internal operating system event.

— Periodic Timer: Data Logging

During an experiment, it might be necessary to log realtime data periodically to an internal buffer or to a mass storage device. With a periodic scheduling method, a logging module can be started automatically at fixed time intervals to log the data.

Program schedule is requested every 10 seconds.



[Time (in Seconds)]

To achieve this type of scheduling using the specified facilities, one would allocate a per-process timer based on clock ID `CLOCK_REALTIME`. Then the timer would be armed via a call to `timer_settime()` with the `TIMER_ABSTIME` flag reset, and with an initial expiration value and a repetition interval of 10 seconds.

— One-shot Timer (Relative Time): Device Initialization

In an emission test environment, large sample bags are used to capture the exhaust from a vehicle. The exhaust is purged from these bags before each and every test. With a one-shot timer, a module could initiate the purge function and then suspend itself for a predetermined period of time while the sample bags are prepared.

Program schedule requested 20 seconds after call is issued.







effectively to minimize bottlenecks and delays and maximize facility throughput.

Program schedule requested for 2:00 a.m. and every 15 minutes thereafter.

```

 R S S S S S S
-----+-----+-----+-----+-----+-----+-----+----->
 23:00 23:30 24:00 00:30 01:00 01:30 02:00 02:30 03:00

```

[Time of Day]

To achieve this type of scheduling using the specified facilities, one would allocate a per-process timer based on clock ID `CLOCK_REALTIME`. Then the timer would be armed via a call to `timer_settime()` with `TIMER_ABSTIME` flag set, and with an initial expiration value equal to 2:00 a.m. and a repetition interval equal to 15 minutes.

- Relationship of Timers to Clocks

The relationship between clocks and timers armed with an absolute time is straightforward: a timer expiration signal is requested when the associated clock reaches or exceeds the specified time. The relationship between clocks and timers armed with a relative time (an interval) is less obvious, but not unintuitive. In this case, a timer expiration signal is requested when the specified interval, *as measured by the associated clock*, has passed. For the required `CLOCK_REALTIME` clock, this allows timer expiration signals to be requested at specified “wall clock” times (absolute), or when a specified interval of “realtime” has passed (relative). For an implementation-defined clock—say, a process virtual time clock—timer expirations could be requested when the process has used a specified total amount of virtual time (absolute), or when it has used a specified *additional* amount of virtual time (relative).

The interfaces also allow flexibility in the implementation of the functions. For example, an implementation could convert all absolute times to intervals by subtracting the clock value at the time of the call from the requested expiration time and “counting down” at the supported resolution. Or it could convert all relative times to absolute expiration time by adding in the clock value at the time of the call and comparing the clock value to the expiration time at the supported resolution. Or it might even choose to maintain absolute times as absolute and compare them to the clock value at the supported resolution for absolute timers, and maintain relative times as intervals and count them down at the resolution supported for relative timers. The choice will be driven by efficiency considerations and the underlying hardware or software clock implementation.

- Data Definitions for Clocks and Timers

POSIX.1-2008 uses a time representation capable of supporting nanosecond resolution timers for the following reasons:

- To enable POSIX.1-2008 to represent those computer systems already using nanosecond or submicrosecond resolution clocks.
- To accommodate those per-process timers that might need nanoseconds to specify an absolute value of system-wide clocks, even though the resolution of the per-process timer may only be milliseconds, or *vice versa*.
- Because the number of nanoseconds in a second can be represented in 32 bits.

Time values are represented in the **timespec** structure. The `tv_sec` member is of type **time\_t** so that this member is compatible with time values used by POSIX.1 functions and the ISO C standard. The `tv_nsec` member is a **signed long** in order to simplify and clarify code that decrements or finds differences of time values. Note that because 1 billion (number of nanoseconds per second) is less than half of the value representable by a signed 32-bit

121647 value, it is always possible to add two valid fractional seconds represented as integral  
121648 nanoseconds without overflowing the signed 32-bit value.

121649 A maximum allowable resolution for the CLOCK\_REALTIME clock of 20 ms (1/50  
121650 seconds) was chosen to allow line frequency clocks in European countries to be  
121651 conforming. 60 Hz clocks in the US will also be conforming, as will finer granularity  
121652 clocks, although a Strictly Conforming Application cannot assume a granularity of less  
121653 than 20 ms (1/50 seconds).

121654 The minimum allowable maximum time allowed for the CLOCK\_REALTIME clock and  
121655 the function *nanosleep()*, and timers created with *clock\_id=CLOCK\_REALTIME*, is  
121656 determined by the fact that the *tv\_sec* member is of type **time\_t**.

121657 POSIX.1-2008 specifies that timer expirations must not be delivered early, and *nanosleep()*  
121658 must not return early due to quantization error. POSIX.1-2008 discusses the various  
121659 implementations of *alarm()* in the rationale and states that implementations that do not  
121660 allow alarm signals to occur early are the most appropriate, but refrained from mandating  
121661 this behavior. Because of the importance of predictability to realtime applications,  
121662 POSIX.1-2008 takes a stronger stance.

121663 The standard developers considered using a time representation that differs from  
121664 POSIX.1b in the second 32 bit of the 64-bit value. Whereas POSIX.1b defines this field as a  
121665 fractional second in nanoseconds, the other methodology defines this as a binary fraction  
121666 of one second, with the radix point assumed before the most significant bit.

121667 POSIX.1b is a software, source-level standard and most of the benefits of the alternate  
121668 representation are enjoyed by hardware implementations of clocks and algorithms. It was  
121669 felt that mandating this format for POSIX.1b clocks and timers would unnecessarily  
121670 burden the application developer with writing, possibly non-portable, multiple precision  
121671 arithmetic packages to perform conversion between binary fractions and integral units  
121672 such as nanoseconds, milliseconds, and so on.

## 121673 **Rationale for the Monotonic Clock**

121674 For those applications that use time services to achieve realtime behavior, changing the value of  
121675 the clock on which these services rely may cause erroneous timing behavior. For these  
121676 applications, it is necessary to have a monotonic clock which cannot run backwards, and which  
121677 has a maximum clock jump that is required to be documented by the implementation.  
121678 Additionally, it is desirable (but not required by POSIX.1-2008) that the monotonic clock  
121679 increases its value uniformly. This clock should not be affected by changes to the system time;  
121680 for example, to synchronize the clock with an external source or to account for leap seconds.  
121681 Such changes would cause errors in the measurement of time intervals for those time services  
121682 that use the absolute value of the clock.

121683 One could argue that by defining the behavior of time services when the value of a clock is  
121684 changed, deterministic realtime behavior can be achieved. For example, one could specify that  
121685 relative time services should be unaffected by changes in the value of a clock. However, there are  
121686 time services that are based upon an absolute time, but that are essentially intended as relative  
121687 time services. For example, *pthread\_cond\_timedwait()* uses an absolute time to allow it to wake  
121688 up after the required interval despite spurious wakeups. Although sometimes the  
121689 *pthread\_cond\_timedwait()* timeouts are absolute in nature, there are many occasions in which they  
121690 are relative, and their absolute value is determined from the current time plus a relative time  
121691 interval. In this latter case, if the clock changes while the thread is waiting, the wait interval will  
121692 not be the expected length. If a *pthread\_cond\_timedwait()* function were created that would take a  
121693 relative time, it would not solve the problem because to retain the intended “deadline” a thread  
121694 would need to compensate for latency due to the spurious wakeup, and preemption between

121695 wakeup and the next wait.

121696 The solution is to create a new monotonic clock, whose value does not change except for the  
 121697 regular ticking of the clock, and use this clock for implementing the various relative timeouts  
 121698 that appear in the different POSIX interfaces, as well as allow *pthread\_cond\_timedwait()* to choose  
 121699 this new clock for its timeout. A new *clock\_nanosleep()* function is created to allow an application  
 121700 to take advantage of this newly defined clock. Notice that the monotonic clock may be  
 121701 implemented using the same hardware clock as the system clock.

121702 Relative timeouts for *sigtimedwait()* and *aio\_suspend()* have been redefined to use the monotonic  
 121703 clock, if present. The *alarm()* function has not been redefined, because the same effect but with  
 121704 better resolution can be achieved by creating a timer (for which the appropriate clock may be  
 121705 chosen).

121706 The *pthread\_cond\_timedwait()* function has been treated in a different way, compared to other  
 121707 functions with absolute timeouts, because it is used to wait for an event, and thus it may have a  
 121708 deadline, while the other timeouts are generally used as an error recovery mechanism, and for  
 121709 them the use of the monotonic clock is not so important. Since the desired timeout for the  
 121710 *pthread\_cond\_timedwait()* function may either be a relative interval or an absolute time of day  
 121711 deadline, a new initialization attribute has been created for condition variables to specify the  
 121712 clock that is used for measuring the timeout in a call to *pthread\_cond\_timedwait()*. In this way, if  
 121713 a relative timeout is desired, the monotonic clock will be used; if an absolute deadline is  
 121714 required instead, the *CLOCK\_REALTIME* or another appropriate clock may be used. This  
 121715 capability has not been added to other functions with absolute timeouts because for those  
 121716 functions the expected use of the timeout is mostly to prevent errors, and not so often to meet  
 121717 precise deadlines. As a consequence, the complexity of adding this capability is not justified by  
 121718 its perceived application usage.

121719 The *nanosleep()* function has not been modified with the introduction of the monotonic clock.  
 121720 Instead, a new *clock\_nanosleep()* function has been created, in which the desired clock may be  
 121721 specified in the function call.

#### 121722 • History of Resolution Issues

121723 Due to the shift from relative to absolute timeouts in IEEE Std 1003.1d-1999, the  
 121724 amendments to the *sem\_timedwait()*, *pthread\_mutex\_timedlock()*, *mq\_timedreceive()*, and  
 121725 *mq\_timedsend()* functions of that standard have been removed. Those amendments  
 121726 specified that *CLOCK\_MONOTONIC* would be used for the (relative) timeouts if the  
 121727 Monotonic Clock option was supported.

121728 Having these functions continue to be tied solely to *CLOCK\_MONOTONIC* would not  
 121729 work. Since the absolute value of a time value obtained from *CLOCK\_MONOTONIC* is  
 121730 unspecified, under the absolute timeouts interface, applications would behave differently  
 121731 depending on whether the Monotonic Clock option was supported or not (because the  
 121732 absolute value of the clock would have different meanings in either case).

121733 Two options were considered:

- 121734 1. Leave the current behavior unchanged, which specifies the *CLOCK\_REALTIME*  
 121735 clock for these (absolute) timeouts, to allow portability of applications between  
 121736 implementations supporting or not the Monotonic Clock option.
- 121737 2. Modify these functions in the way that *pthread\_cond\_timedwait()* was modified to  
 121738 allow a choice of clock, so that an application could use *CLOCK\_REALTIME* when  
 121739 it is trying to achieve an absolute timeout and *CLOCK\_MONOTONIC* when it is  
 121740 trying to achieve a relative timeout.

121741 It was decided that the features of *CLOCK\_MONOTONIC* are not as critical to these

functions as they are to `pthread_cond_timedwait()`. The `pthread_cond_timedwait()` function is given a relative timeout; the timeout may represent a deadline for an event. When these functions are given relative timeouts, the timeouts are typically for error recovery purposes and need not be so precise.

Therefore, it was decided that these functions should be tied to `CLOCK_REALTIME` and not complicated by being given a choice of clock.

## Execution Time Monitoring

- Introduction

The main goals of the execution time monitoring facilities defined in this chapter are to measure the execution time of processes and threads and to allow an application to establish CPU time limits for these entities.

The analysis phase of time-critical realtime systems often relies on the measurement of execution times of individual threads or processes to determine whether the timing requirements will be met. Also, performance analysis techniques for soft deadline realtime systems rely heavily on the determination of these execution times. The execution time monitoring functions provide application developers with the ability to measure these execution times online and open the possibility of dynamic execution-time analysis and system reconfiguration, if required.

The second goal of allowing an application to establish execution time limits for individual processes or threads and detecting when they overrun allows program robustness to be increased by enabling online checking of the execution times.

If errors are detected—possibly because of erroneous program constructs, the existence of errors in the analysis phase, or a burst of event arrivals—online detection and recovery is possible in a portable way. This feature can be extremely important for many time-critical applications. Other applications require trapping CPU-time errors as a normal way to exit an algorithm; for instance, some realtime artificial intelligence applications trigger a number of independent inference processes of varying accuracy and speed, limit how long they can run, and pick the best answer available when time runs out. In many periodic systems, overrun processes are simply restarted in the next resource period, after necessary end-of-period actions have been taken. This allows algorithms that are inherently data-dependent to be made predictable.

The interface that appears in this chapter defines a new type of clock, the CPU-time clock, which measures execution time. Each process or thread can invoke the clock and timer functions defined in POSIX.1 to use them. Functions are also provided to access the CPU-time clock of other processes or threads to enable remote monitoring of these clocks. Monitoring of threads of other processes is not supported, since these threads are not visible from outside of their own process with the interfaces defined in POSIX.1.

- Execution Time Monitoring Interface

The clock and timer interface defined in POSIX.1 historically only defined one clock, which measures wall-clock time. The requirements for measuring execution time of processes and threads, and setting limits to their execution time by detecting when they overrun, can be accomplished with that interface if a new kind of clock is defined. These new clocks measure execution time, and one is associated with each process and with each thread. The clock functions currently defined in POSIX.1 can be used to read and set these CPU-time clocks, and timers can be created using these clocks as their timing base. These timers can then be used to send a signal when some specified execution time has been exceeded. The CPU-time clocks of each process or thread can be accessed by using the symbols

121789 CLOCK\_PROCESS\_CPUTIME\_ID or CLOCK\_THREAD\_CPUTIME\_ID.

121790 The clock and timer interface defined in POSIX.1 and extended with the new kind of CPU-  
 121791 time clock would only allow processes or threads to access their own CPU-time clocks.  
 121792 However, many realtime systems require the possibility of monitoring the execution time  
 121793 of processes or threads from independent monitoring entities. In order to allow  
 121794 applications to construct independent monitoring entities that do not require cooperation  
 121795 from or modification of the monitored entities, two functions have been added:  
 121796 *clock\_getcpuclockid()*, for accessing CPU-time clocks of other processes, and  
 121797 *pthread\_getcpuclockid()*, for accessing CPU-time clocks of other threads. These functions  
 121798 return the clock identifier associated with the process or thread specified in the call. These  
 121799 clock IDs can then be used in the rest of the clock function calls.

121800 The clocks accessed through these functions could also be used as a timing base for the  
 121801 creation of timers, thereby allowing independent monitoring entities to limit the CPU time  
 121802 consumed by other entities. However, this possibility would imply additional complexity  
 121803 and overhead because of the need to maintain a timer queue for each process or thread, to  
 121804 store the different expiration times associated with timers created by different processes or  
 121805 threads. The working group decided this additional overhead was not justified by  
 121806 application requirements. Therefore, creation of timers attached to the CPU-time clocks of  
 121807 other processes or threads has been specified as implementation-defined.

#### 121808 • Overhead Considerations

121809 The measurement of execution time may introduce additional overhead in the thread  
 121810 scheduling, because of the need to keep track of the time consumed by each of these  
 121811 entities. In library-level implementations of threads, the efficiency of scheduling could be  
 121812 somehow compromised because of the need to make a kernel call, at each context switch,  
 121813 to read the process CPU-time clock. Consequently, a thread creation attribute called *cpu-  
 121814 clock-requirement* was defined, to allow threads to disconnect their respective CPU-time  
 121815 clocks. However, the Ballot Group considered that this attribute itself introduced some  
 121816 overhead, and that in current implementations it was not worth the effort. Therefore, the  
 121817 attribute was deleted, and thus thread CPU-time clocks are required for all threads if the  
 121818 Thread CPU-Time Clocks option is supported.

#### 121819 • Accuracy of CPU-Time Clocks

121820 The mechanism used to measure the execution time of processes and threads is specified in  
 121821 POSIX.1-2008 as implementation-defined. The reason for this is that both the underlying  
 121822 hardware and the implementation architecture have a very strong influence on the  
 121823 accuracy achievable for measuring CPU time. For some implementations, the specification  
 121824 of strict accuracy requirements would represent very large overheads, or even the  
 121825 impossibility of being implemented.

121826 Since the mechanism for measuring execution time is implementation-defined, realtime  
 121827 applications will be able to take advantage of accurate implementations using a portable  
 121828 interface. Of course, strictly conforming applications cannot rely on any particular degree  
 121829 of accuracy, in the same way as they cannot rely on a very accurate measurement of wall  
 121830 clock time. There will always exist applications whose accuracy or efficiency requirements  
 121831 on the implementation are more rigid than the values defined in POSIX.1-2008 or any  
 121832 other standard.

121833 In any case, there is a minimum set of characteristics that realtime applications would  
 121834 expect from most implementations. One such characteristic is that the sum of all the  
 121835 execution times of all the threads in a process equals the process execution time, when no  
 121836 CPU-time clocks are disabled. This need not always be the case because implementations  
 121837 may differ in how they account for time during context switches. Another characteristic is

that the sum of the execution times of all processes in a system equals the number of processors, multiplied by the elapsed time, assuming that no processor is idle during that elapsed time. However, in some implementations it might not be possible to relate CPU time to elapsed time. For example, in a heterogeneous multi-processor system in which each processor runs at a different speed, an implementation may choose to define each “second” of CPU time to be a certain number of “cycles” that a CPU has executed.

- Existing Practice

Measuring and limiting the execution time of each concurrent activity are common features of most industrial implementations of realtime systems. Almost all critical realtime systems are currently built upon a cyclic executive. With this approach, a regular timer interrupt kicks off the next sequence of computations. It also checks that the current sequence has completed. If it has not, then some error recovery action can be undertaken (or at least an overrun is avoided). Current software engineering principles and the increasing complexity of software are driving application developers to implement these systems on multi-threaded or multi-process operating systems. Therefore, if a POSIX operating system is to be used for this type of application, then it must offer the same level of protection.

Execution time clocks are also common in most UNIX implementations, although these clocks usually have requirements different from those of realtime applications. The POSIX.1 *times()* function supports the measurement of the execution time of the calling process, and its terminated child processes. This execution time is measured in clock ticks and is supplied as two different values with the user and system execution times, respectively. BSD supports the function *getrusage()*, which allows the calling process to get information about the resources used by itself and/or all of its terminated child processes. The resource usage includes user and system CPU time. Some UNIX systems have options to specify high resolution (up to one microsecond) CPU-time clocks using the *times()* or the *getrusage()* functions.

The *times()* and *getrusage()* interfaces do not meet important realtime requirements, such as the possibility of monitoring execution time from a different process or thread, or the possibility of detecting an execution time overrun. The latter requirement is supported in some UNIX implementations that are able to send a signal when the execution time of a process has exceeded some specified value. For example, BSD defines the functions *getitimer()* and *setitimer()*, which can operate either on a realtime clock (wall-clock), or on virtual-time or profile-time clocks which measure CPU time in two different ways. These functions do not support access to the execution time of other processes.

The IBM MVS operating system supports per-process and per-thread execution time clocks. It also supports limiting the execution time of a given process.

Given all this existing practice, the working group considered that the POSIX.1 clocks and timers interface was appropriate to meet most of the requirements that realtime applications have for execution time clocks. Functions were added to get the CPU time clock IDs, and to allow/disallow the thread CPU-time clocks (in order to preserve the efficiency of some implementations of threads).

- Clock Constants

The definition of the manifest constants `CLOCK_PROCESS_CPUTIME_ID` and `CLOCK_THREAD_CPUTIME_ID` allows processes or threads, respectively, to access their own execution-time clocks. However, given a process or thread, access to its own execution-time clock is also possible if the clock ID of this clock is obtained through a call to *clock\_getcpuclockid()* or *pthread\_getcpuclockid()*. Therefore, these constants are not necessary and could be deleted to make the interface simpler. Their existence saves one

121887 system call in the first access to the CPU-time clock of each process or thread. The working  
 121888 group considered this issue and decided to leave the constants in POSIX.1-2008 because  
 121889 they are closer to the POSIX.1b use of clock identifiers.

121890 • Library Implementations of Threads

121891 In library implementations of threads, kernel entities and library threads can coexist. In  
 121892 this case, if the CPU-time clocks are supported, most of the clock and timer functions will  
 121893 need to have two implementations: one in the thread library, and one in the system calls  
 121894 library. The main difference between these two implementations is that the thread library  
 121895 implementation will have to deal with clocks and timers that reside in the thread space,  
 121896 while the kernel implementation will operate on timers and clocks that reside in kernel  
 121897 space. In the library implementation, if the clock ID refers to a clock that resides in the  
 121898 kernel, a kernel call will have to be made. The correct version of the function can be chosen  
 121899 by specifying the appropriate order for the libraries during the link process.

121900 • History of Resolution Issues: Deletion of the *enable* Attribute

121901 In early proposals, consideration was given to inclusion of an attribute called *enable* for  
 121902 CPU-time clocks. This would allow implementations to avoid the overhead of measuring  
 121903 execution time for those processes or threads for which this measurement was not  
 121904 required. However, this is unnecessary since processes are already required to measure  
 121905 execution time by the POSIX.1 *times()* function. Consequently, the *enable* attribute is not  
 121906 present.

121907 **Rationale Relating to Timeouts**

121908 • Requirements for Timeouts

121909 Realtime systems which must operate reliably over extended periods without human  
 121910 intervention are characteristic in embedded applications such as avionics, machine control,  
 121911 and space exploration, as well as more mundane applications such as cable TV, security  
 121912 systems, and plant automation. A multi-tasking paradigm, in which many independent  
 121913 and/or cooperating software functions relinquish the processor(s) while waiting for a  
 121914 specific stimulus, resource, condition, or operation completion, is very useful in producing  
 121915 well engineered programs for such systems. For such systems to be robust and fault-  
 121916 tolerant, expected occurrences that are unduly delayed or that never occur must be  
 121917 detected so that appropriate recovery actions may be taken. This is difficult if there is no  
 121918 way for a task to regain control of a processor once it has relinquished control (blocked)  
 121919 awaiting an occurrence which, perhaps because of corrupted code, hardware malfunction,  
 121920 or latent software bugs, will not happen when expected. Therefore, the common practice  
 121921 in realtime operating systems is to provide a capability to time out such blocking services.  
 121922 Although there are several methods to achieve this already defined by POSIX, none are as  
 121923 reliable or efficient as initiating a timeout simultaneously with initiating a blocking service.  
 121924 This is especially critical in hard-realtime embedded systems because the processors  
 121925 typically have little time reserve, and allowed fault recovery times are measured in  
 121926 milliseconds rather than seconds.

121927 The working group largely agreed that such timeouts were necessary and ought to become  
 121928 part of POSIX.1-2008, particularly vendors of realtime operating systems whose customers  
 121929 had already expressed a strong need for timeouts. There was some resistance to inclusion  
 121930 of timeouts in POSIX.1-2008 because the desired effect, fault tolerance, could, in theory, be  
 121931 achieved using existing facilities and alternative software designs, but there was no  
 121932 compelling evidence that realtime system designers would embrace such designs at the  
 121933 sacrifice of performance and/or simplicity.

- Which Services should be Timed Out?

Originally, the working group considered the prospect of providing timeouts on all blocking services, including those currently existing in POSIX.1, POSIX.1b, and POSIX.1c, and future interfaces to be defined by other working groups, as sort of a general policy. This was rather quickly rejected because of the scope of such a change, and the fact that many of those services would not normally be used in a realtime context. More traditional timesharing solutions to timeout would suffice for most of the POSIX.1 interfaces, while others had asynchronous alternatives which, while more complex to utilize, would be adequate for some realtime and all non-realtime applications.

The list of potential candidates for timeouts was narrowed to the following for further consideration:

- POSIX.1b
  - *sem\_wait()*
  - *mq\_receive()*
  - *mq\_send()*
  - *lio\_listio()*
  - *aio\_suspend()*
  - *sigwait()* (timeout already implemented by *sigtimedwait()*)
- POSIX.1c
  - *pthread\_mutex\_lock()*
  - *pthread\_join()*
  - *pthread\_cond\_wait()*  
(timeout already implemented by *pthread\_cond\_timedwait()*)
- POSIX.1
  - *read()*
  - *write()*

After further review by the working group, the *lio\_listio()*, *read()*, and *write()* functions (all forms of blocking synchronous I/O) were eliminated from the list because of the following:

- Asynchronous alternatives exist
- Timeouts can be implemented, albeit non-portably, in device drivers
- A strong desire not to introduce modifications to POSIX.1 interfaces

The working group ultimately rejected *pthread\_join()* since both that interface and a timed variant of that interface are non-minimal and may be implemented as a function. See below for a library implementation of *pthread\_join()*.

Thus, there was a consensus among the working group members to add timeouts to 4 of the remaining 5 functions (the timeout for *aio\_suspend()* was ultimately added directly to POSIX.1b, while the others were added by POSIX.1d). However, *pthread\_mutex\_lock()* remained contentious.

Many feel that *pthread\_mutex\_lock()* falls into the same class as the other functions; that is, it is desirable to time out a mutex lock because a mutex may fail to be unlocked due to



errant or corrupted code in a critical section (looping or branching outside of the unlock code), and therefore is equally in need of a reliable, simple, and efficient timeout. In fact, since mutexes are intended to guard small critical sections, most `pthread_mutex_lock()` calls would be expected to obtain the lock without blocking nor utilizing any kernel service, even in implementations of threads with global contention scope; the timeout alternative need only be considered after it is determined that the thread must block.

Those opposed to timing out mutexes feel that the very simplicity of the mutex is compromised by adding a timeout semantic, and that to do so is senseless. They claim that if a timed mutex is really deemed useful by a particular application, then it can be constructed from the facilities already in POSIX.1b and POSIX.1c. The following two C-language library implementations of mutex locking with timeout represent the solutions offered (in both implementations, the timeout parameter is specified as absolute time, not relative time as in the proposed POSIX.1c interfaces).

- Spinlock Implementation

```
#include <pthread.h>
#include <time.h>
#include <errno.h>

int pthread_mutex_timedlock(pthread_mutex_t *mutex,
 const struct timespec *timeout)
{
 struct timespec timenow;

 while (pthread_mutex_trylock(mutex) == EBUSY)
 {
 clock_gettime(CLOCK_REALTIME, &timenow);
 if (timespec_cmp(&timenow, timeout) >= 0)
 {
 return ETIMEDOUT;
 }
 pthread_yield();
 }
 return 0;
}
```

The Spinlock implementation is generally unsuitable for any application using priority-based thread scheduling policies such as SCHED\_FIFO or SCHED\_RR, since the mutex could currently be held by a thread of lower priority within the same allocation domain, but since the waiting thread never blocks, only threads of equal or higher priority will ever run, and the mutex cannot be unlocked. Setting priority inheritance or priority ceiling protocol on the mutex does not solve this problem, since the priority of a mutex owning thread is only boosted if higher priority threads are blocked waiting for the mutex; clearly not the case for this spinlock.

- Condition Wait Implementation

```
#include <pthread.h>
#include <time.h>
#include <errno.h>

struct timed_mutex
{
 int locked;
 pthread_mutex_t mutex;
}
```

```

122023 pthread_cond_t cond;
122024 };
122025 typedef struct timed_mutex timed_mutex_t;
122026 int timed_mutex_lock(timed_mutex_t *tm,
122027 const struct timespec *timeout)
122028 {
122029 int timedout=FALSE;
122030 int error_status;
122031
122032 pthread_mutex_lock(&tm->mutex);
122033
122034 while (tm->locked && !timedout)
122035 {
122036 if ((error_status=pthread_cond_timedwait(&tm->cond,
122037 &tm->mutex,
122038 timeout))!=0)
122039 {
122040 if (error_status==ETIMEDOUT) timedout = TRUE;
122041 }
122042 }
122043
122044 if(timedout)
122045 {
122046 pthread_mutex_unlock(&tm->mutex);
122047 return ETIMEDOUT;
122048 }
122049 else
122050 {
122051 tm->locked = TRUE;
122052 pthread_mutex_unlock(&tm->mutex);
122053 return 0;
122054 }
122055 }
122056
122057 void timed_mutex_unlock(timed_mutex_t *tm)
122058 {
122059 pthread_mutex_lock(&tm->mutex); / for case assignment not atomic /
122060 tm->locked = FALSE;
122061 pthread_mutex_unlock(&tm->mutex);
122062 pthread_cond_signal(&tm->cond);
122063 }

```

The Condition Wait implementation effectively substitutes the *pthread\_cond\_timedwait()* function (which is currently timed out) for the desired *pthread\_mutex\_timedlock()*. Since waits on condition variables currently do not include protocols which avoid priority inversion, this method is generally unsuitable for realtime applications because it does not provide the same priority inversion protection as the untimed *pthread\_mutex\_lock()*. Also, for any given implementations of the current mutex and condition variable primitives, this library implementation has a performance cost at least 2.5 times that of the untimed *pthread\_mutex\_lock()* even in the case where the timed mutex is readily locked without blocking (the interfaces required for this case are shown in bold). Even in uniprocessors or where assignment is atomic, at least an additional *pthread\_cond\_signal()* is required. *pthread\_mutex\_timedlock()* could be implemented at effectively no performance penalty in this case because the timeout parameters need only be considered after it is determined that the mutex cannot be locked immediately.

Thus it has not yet been shown that the full semantics of mutex locking with timeout can be efficiently and reliably achieved using existing interfaces. Even if the existence of an acceptable library implementation were proven, it is difficult to justify why the interface itself should not be made portable, especially considering approval for the other four timeouts.

- Rationale for Library Implementation of *pthread\_timedjoin()*

Library implementation of *pthread\_timedjoin()*:

```

/*
 * Construct a thread variety entirely from existing functions
 * with which a join can be done, allowing the join to time out.
 */
#include <pthread.h>
#include <time.h>

struct timed_thread {
 pthread_t t;
 pthread_mutex_t m;
 int exiting;
 pthread_cond_t exit_c;
 void *(*start_routine)(void *arg);
 void *arg;
 void *status;
};

typedef struct timed_thread *timed_thread_t;
static pthread_key_t timed_thread_key;
static pthread_once_t timed_thread_once = PTHREAD_ONCE_INIT;

static void timed_thread_init()
{
 pthread_key_create(&timed_thread_key, NULL);
}

static void *timed_thread_start_routine(void *args)
/*
 * Routine to establish thread-specific data value and run the actual
 * thread start routine which was supplied to timed_thread_create().
 */
{
 timed_thread_t tt = (timed_thread_t) args;

 pthread_once(&timed_thread_once, timed_thread_init);
 pthread_setspecific(timed_thread_key, (void *)tt);
 timed_thread_exit((tt->start_routine)(tt->arg));
}

int timed_thread_create(timed_thread_t ttp, const pthread_attr_t *attr,
 void *(*start_routine)(void *), void *arg)
/*
 * Allocate a thread which can be used with timed_thread_join().
 */
{
 timed_thread_t tt;

```

```

122120 int result;
122121 tt = (timed_thread_t) malloc(sizeof(struct timed_thread));
122122 pthread_mutex_init(&tt->m, NULL);
122123 tt->exiting = FALSE;
122124 pthread_cond_init(&tt->exit_c, NULL);
122125 tt->start_routine = start_routine;
122126 tt->arg = arg;
122127 tt->status = NULL;
122128
122129 if ((result = pthread_create(&tt->t, attr,
122130 timed_thread_start_routine, (void *)tt)) != 0) {
122131 free(tt);
122132 return result;
122133 }
122134 pthread_detach(tt->t);
122135 ttp = tt;
122136 return 0;
122137 }
122138
122139 int timed_thread_join(timed_thread_t tt,
122140 struct timespec *timeout,
122141 void **status)
122142 {
122143 int result;
122144
122145 pthread_mutex_lock(&tt->m);
122146 result = 0;
122147 /*
122148 * Wait until the thread announces that it is exiting,
122149 * or until timeout.
122150 */
122151 while (result == 0 && ! tt->exiting) {
122152 result = pthread_cond_timedwait(&tt->exit_c, &tt->m, timeout);
122153 }
122154 pthread_mutex_unlock(&tt->m);
122155 if (result == 0 && tt->exiting) {
122156 *status = tt->status;
122157 free((void *)tt);
122158 return result;
122159 }
122160 return result;
122161 }
122162
122163 void timed_thread_exit(void *status)
122164 {
122165 timed_thread_t tt;
122166 void *specific;
122167
122168 if ((specific=pthread_getspecific(timed_thread_key)) == NULL) {
122169 /*
122170 * Handle cases which won't happen with correct usage.
122171 */
122172 pthread_exit(NULL);
122173 }
122174 }

```

```

122169 tt = (timed_thread_t) specific;
122170 pthread_mutex_lock(&tt->m);
122171 /*
122172 * Tell a joiner that we're exiting.
122173 */
122174 tt->status = status;
122175 tt->exiting = TRUE;
122176 pthread_cond_signal(&tt->exit_c);
122177 pthread_mutex_unlock(&tt->m);
122178 /*
122179 * Call pthread_exit() to call destructors and really
122180 * exit the thread.
122181 */
122182 pthread_exit(NULL);
122183 }

```

The `pthread_join()` C-language example shown above demonstrates that it is possible, using existing pthread facilities, to construct a variety of thread which allows for joining such a thread, but which allows the join operation to time out. It does this by using a `pthread_cond_timedwait()` to wait for the thread to exit. A **timed\_thread\_t** descriptor structure is used to pass parameters from the creating thread to the created thread, and from the exiting thread to the joining thread. This implementation is roughly equivalent to what a normal `pthread_join()` implementation would do, with the single change being that `pthread_cond_timedwait()` is used in place of a simple `pthread_cond_wait()`.

Since it is possible to implement such a facility entirely from existing pthread interfaces, and with roughly equal efficiency and complexity to an implementation which would be provided directly by a pthreads implementation, it was the consensus of the working group members that any `pthread_timedjoin()` facility would be unnecessary, and should not be provided.

- Form of the Timeout Interfaces

The working group considered a number of alternative ways to add timeouts to blocking services. At first, a system interface which would specify a one-shot or persistent timeout to be applied to subsequent blocking services invoked by the calling process or thread was considered because it allowed all blocking services to be timed out in a uniform manner with a single additional interface; this was rather quickly rejected because it could easily result in the wrong services being timed out.

It was suggested that a timeout value might be specified as an attribute of the object (semaphore, mutex, message queue, and so on), but there was no consensus on this, either on a case-by-case basis or for all timeouts.

Looking at the two existing timeouts for blocking services indicates that the working group members favor a separate interface for the timed version of a function. However, `pthread_cond_timedwait()` utilizes an absolute timeout value while `sigtimedwait()` uses a relative timeout value. The working group members agreed that relative timeout values are appropriate where the timeout mechanism's primary use was to deal with an unexpected or error situation, but they are inappropriate when the timeout must expire at a particular time, or before a specific deadline. For the timeouts being introduced in POSIX.1-2008, the working group considered allowing both relative and absolute timeouts as is done with POSIX.1b timers, but ultimately favored the simpler absolute timeout form.

An absolute time measure can be easily implemented on top of an interface that specifies relative time, by reading the clock, calculating the difference between the current time and

122218 the desired wakeup time, and issuing a relative timeout call. But there is a race condition  
122219 with this approach because the thread could be preempted after reading the clock, but  
122220 before making the timed-out call; in this case, the thread would be awakened later than it  
122221 should and, thus, if the wakeup time represented a deadline, it would miss it.

122222 There is also a race condition when trying to build a relative timeout on top of an interface  
122223 that specifies absolute timeouts. In this case, the clock would have to be read to calculate  
122224 the absolute wakeup time as the sum of the current time plus the relative timeout interval.  
122225 In this case, if the thread is preempted after reading the clock but before making the timed-  
122226 out call, the thread would be awakened earlier than desired.

122227 But the race condition with the absolute timeouts interface is not as bad as the one that  
122228 happens with the relative timeout interface, because there are simple workarounds. For the  
122229 absolute timeouts interface, if the timing requirement is a deadline, the deadline can still  
122230 be met because the thread woke up earlier than the deadline. If the timeout is just used as  
122231 an error recovery mechanism, the precision of timing is not really important. If the timing  
122232 requirement is that between actions A and B a minimum interval of time must elapse, the  
122233 absolute timeout interface can be safely used by reading the clock after action A has been  
122234 started. It could be argued that, since the call with the absolute timeout is atomic from the  
122235 application point of view, it is not possible to read the clock after action A, if this action is  
122236 part of the timed-out call. But looking at the nature of the calls for which timeouts are  
122237 specified (locking a mutex, waiting for a semaphore, waiting for a message, or waiting  
122238 until there is space in a message queue), the timeouts that an application would build on  
122239 these actions would not be triggered by these actions themselves, but by some other  
122240 external action. For example, if waiting for a message to arrive to a message queue, and  
122241 waiting for at least 20 milliseconds, this time interval would start to be counted from some  
122242 event that would trigger both the action that produces the message, as well as the action  
122243 that waits for the message to arrive, and not by the wait-for-message operation itself. In  
122244 this case, the workaround proposed above could be used.

122245 For these reasons, the absolute timeout is preferred over the relative timeout interface.

## 122246 B.2.9 Threads

122247 Threads will normally be more expensive than subroutines (or functions, routines, and so on) if  
122248 specialized hardware support is not provided. Nevertheless, threads should be sufficiently  
122249 efficient to encourage their use as a medium to fine-grained structuring mechanism for  
122250 parallelism in an application. Structuring an application using threads then allows it to take  
122251 immediate advantage of any underlying parallelism available in the host environment. This  
122252 means implementors are encouraged to optimize for fast execution at the possible expense of  
122253 efficient utilization of storage. For example, a common thread creation technique is to cache  
122254 appropriate thread data structures. That is, rather than releasing system resources, the  
122255 implementation retains these resources and reuses them when the program next asks to create a  
122256 new thread. If this reuse of thread resources is to be possible, there has to be very little unique  
122257 state associated with each thread, because any such state has to be reset when the thread is  
122258 reused.

## Thread Creation Attributes

Attributes objects are provided for threads, mutexes, and condition variables as a mechanism to support probable future standardization in these areas without requiring that the interface itself be changed.

Attributes objects provide clean isolation of the configurable aspects of threads. For example, “stack size” is an important attribute of a thread, but it cannot be expressed portably. When porting a threaded program, stack sizes often need to be adjusted. The use of attributes objects can help by allowing the changes to be isolated in a single place, rather than being spread across every instance of thread creation.

Attributes objects can be used to set up *classes* of threads with similar attributes; for example, “threads with large stacks and high priority” or “threads with minimal stacks”. These classes can be defined in a single place and then referenced wherever threads need to be created. Changes to “class” decisions become straightforward, and detailed analysis of each `pthread_create()` call is not required.

The attributes objects are defined as opaque types as an aid to extensibility. If these objects had been specified as structures, adding new attributes would force recompilation of all multi-threaded programs when the attributes objects are extended; this might not be possible if different program components were supplied by different vendors.

Additionally, opaque attributes objects present opportunities for improving performance. Argument validity can be checked once when attributes are set, rather than each time a thread is created. Implementations will often need to cache kernel objects that are expensive to create. Opaque attributes objects provide an efficient mechanism to detect when cached objects become invalid due to attribute changes.

Because assignment is not necessarily defined on a given opaque type, implementation-defined default values cannot be defined in a portable way. The solution to this problem is to allow attribute objects to be initialized dynamically by attributes object initialization functions, so that default values can be supplied automatically by the implementation.

The following proposal was provided as a suggested alternative to the supplied attributes:

1. Maintain the style of passing a parameter formed by the bitwise-inclusive OR of flags to the initialization routines (`pthread_create()`, `pthread_mutex_init()`, `pthread_cond_init()`). The parameter containing the flags should be an opaque type for extensibility. If no flags are set in the parameter, then the objects are created with default characteristics. An implementation may specify implementation-defined flag values and associated behavior.
2. If further specialization of mutexes and condition variables is necessary, implementations may specify additional procedures that operate on the `pthread_mutex_t` and `pthread_cond_t` objects (instead of on attributes objects).

The difficulties with this solution are:

1. A bitmask is not opaque if bits have to be set into bit-vector attributes objects using explicitly-coded bitwise-inclusive OR operations. If the set of options exceeds an `int`, application programmers need to know the location of each bit. If bits are set or read by encapsulation (that is, `get*()` or `set*()` functions), then the bitmask is merely an implementation of attributes objects as currently defined and should not be exposed to the programmer.
2. Many attributes are not Boolean or very small integral values. For example, scheduling policy may be placed in 3 bits or 4 bits, but priority requires 5 bits or more, thereby taking up at least 8 bits out of a possible 16 bits on machines with 16-bit integers. Because of this,

the bitmask can only reasonably control whether particular attributes are set or not, and it cannot serve as the repository of the value itself. The value needs to be specified as a function parameter (which is non-extensible), or by setting a structure field (which is non-opaque), or by *get\**() and *set\**() functions (making the bitmask a redundant addition to the attributes objects).

Stack size is defined as an optional attribute because the very notion of a stack is inherently machine-dependent. Some implementations may not be able to change the size of the stack, for example, and others may not need to because stack pages may be discontinuous and can be allocated and released on demand.

The attribute mechanism has been designed in large measure for extensibility. Future extensions to the attribute mechanism or to any attributes object defined in POSIX.1-2008 have to be done with care so as not to affect binary-compatibility.

Attribute objects, even if allocated by means of dynamic allocation functions such as *malloc*(), may have their size fixed at compile time. This means, for example, a *pthread\_create*() in an implementation with extensions to the **pthread\_attr\_t** cannot look beyond the area that the binary application assumes is valid. This suggests that implementations should maintain a size field in the attributes object, as well as possibly version information, if extensions in different directions (possibly by different vendors) are to be accommodated.

### Thread Implementation Models

There are various thread implementation models. At one end of the spectrum is the “library-thread model”. In such a model, the threads of a process are not visible to the operating system kernel, and the threads are not kernel-scheduled entities. The process is the only kernel-scheduled entity. The process is scheduled onto the processor by the kernel according to the scheduling attributes of the process. The threads are scheduled onto the single kernel-scheduled entity (the process) by the runtime library according to the scheduling attributes of the threads. A problem with this model is that it constrains concurrency. Since there is only one kernel-scheduled entity (namely, the process), only one thread per process can execute at a time. If the thread that is executing blocks on I/O, then the whole process blocks.

At the other end of the spectrum is the “kernel-thread model”. In this model, all threads are visible to the operating system kernel. Thus, all threads are kernel-scheduled entities, and all threads can concurrently execute. The threads are scheduled onto processors by the kernel according to the scheduling attributes of the threads. The drawback to this model is that the creation and management of the threads entails operating system calls, as opposed to subroutine calls, which makes kernel threads heavier weight than library threads.

Hybrids of these two models are common. A hybrid model offers the speed of library threads and the concurrency of kernel threads. In hybrid models, a process has some (relatively small) number of kernel scheduled entities associated with it. It also has a potentially much larger number of library threads associated with it. Some library threads may be bound to kernel-scheduled entities, while the other library threads are multiplexed onto the remaining kernel-scheduled entities. There are two levels of thread scheduling:

1. The runtime library manages the scheduling of (unbound) library threads onto kernel-scheduled entities.
2. The kernel manages the scheduling of kernel-scheduled entities onto processors.

For this reason, a hybrid model is referred to as a two-level threads scheduling model. In this model, the process can have multiple concurrently executing threads; specifically, it can have as many concurrently executing threads as it has kernel-scheduled entities.



## Thread-Specific Data

Many applications require that a certain amount of context be maintained on a per-thread basis across procedure calls. A common example is a multi-threaded library routine that allocates resources from a common pool and maintains an active resource list for each thread. The thread-specific data interface provided to meet these needs may be viewed as a two-dimensional array of values with keys serving as the row index and thread IDs as the column index (although the implementation need not work this way).

- Models

Three possible thread-specific data models were considered:

1. No Explicit Support

A standard thread-specific data interface is not strictly necessary to support applications that require per-thread context. One could, for example, provide a hash function that converted a **pthread\_t** into an integer value that could then be used to index into a global array of per-thread data pointers. This hash function, in conjunction with *pthread\_self()*, would be all the interface required to support a mechanism of this sort. Unfortunately, this technique is cumbersome. It can lead to duplicated code as each set of cooperating modules implements their own per-thread data management schemes. This technique would also require that **pthread\_t** not be an opaque type.

2. Single (**void \***) Pointer

Another technique would be to provide a single word of per-thread storage and a pair of functions to fetch and store the value of this word. The word could then hold a pointer to a block of per-thread memory. The allocation, partitioning, and general use of this memory would be entirely up to the application. Although this method is not as problematic as technique 1, it suffers from interoperability problems. For example, all modules using the per-thread pointer would have to agree on a common usage protocol.

3. Key/Value Mechanism

This method associates an opaque key (for example, stored in a variable of type **pthread\_key\_t**) with each per-thread datum. These keys play the role of identifiers for per-thread data. This technique is the most generic and avoids the problems noted above, albeit at the cost of some complexity.

The primary advantage of the third model is its information hiding properties. Modules using this model are free to create and use their own key(s) independent of all other such usage, whereas the other models require that all modules that use thread-specific context explicitly cooperate with all other such modules. The data-independence provided by the third model is worth the additional interface. Therefore, the third model was chosen.

- Requirements

It is important that it be possible to implement the thread-specific data interface without the use of thread private memory. To do otherwise would increase the weight of each thread, thereby limiting the range of applications for which the threads interfaces provided by POSIX.1-2008 is appropriate.

The values that one binds to the key via *pthread\_setspecific()* may, in fact, be pointers to shared storage locations available to all threads. It is only the key/value bindings that are maintained on a per-thread basis, and these can be kept in any portion of the address space that is reserved for use by the calling thread (for example, on the stack). Thus, no per-

thread MMU state is required to implement the interface. On the other hand, there is nothing in the interface specification to preclude the use of a per-thread MMU state if it is available (for example, the key values returned by *pthread\_key\_create()* could be thread private memory addresses).

- Standardization Issues

Thread-specific data is a requirement for a usable thread interface. The binding described in this section provides a portable thread-specific data mechanism for languages that do not directly support a thread-specific storage class. A binding to POSIX.1-2008 for a language that does include such a storage class need not provide this specific interface.

If a language were to include the notion of thread-specific storage, it would be desirable (but *not* required) to provide an implementation of the pthreads thread-specific data interface based on the language feature. For example, assume that a compiler for a C-like language supports a *private* storage class that provides thread-specific storage. Something similar to the following macros might be used to effect a compatible implementation:

```
#define pthread_key_t private void *
#define pthread_key_create(key) /* no-op */
#define pthread_setspecific(key,value) (key)=(value)
#define pthread_getspecific(key) (key)
```

**Note:** For the sake of clarity, this example ignores destructor functions. A correct implementation would have to support them.

## Barriers

- Background

Barriers are typically used in parallel DO/FOR loops to ensure that all threads have reached a particular stage in a parallel computation before allowing any to proceed to the next stage. Highly efficient implementation is possible on machines which support a “Fetch and Add” operation as described in the referenced Almasi and Gottlieb (1989).

The use of return value PTHREAD\_BARRIER\_SERIAL\_THREAD is shown in the following example:

```
if ((status=pthread_barrier_wait(&barrier)) ==
 PTHREAD_BARRIER_SERIAL_THREAD) {
 ...serial section
}
else if (status != 0) {
 ...error processing
}
status=pthread_barrier_wait(&barrier);
...
```

This behavior allows a serial section of code to be executed by one thread as soon as all threads reach the first barrier. The second barrier prevents the other threads from proceeding until the serial section being executed by the one thread has completed.

Although barriers can be implemented with mutexes and condition variables, the referenced Almasi and Gottlieb (1989) provides ample illustration that such implementations are significantly less efficient than is possible. While the relative efficiency of barriers may well vary by implementation, it is important that they be recognized in the POSIX.1-2008 to facilitate applications portability while providing the necessary freedom to implementors.

- Lack of Timeout Feature

Alternate versions of most blocking routines have been provided to support watchdog timeouts. No alternate interface of this sort has been provided for barrier waits for the following reasons:

- Multiple threads may use different timeout values, some of which may be indefinite. It is not clear which threads should break through the barrier with a timeout error if and when these timeouts expire.
- The barrier may become unusable once a thread breaks out of a `pthread_barrier_wait()` with a timeout error. There is, in general, no way to guarantee the consistency of a barrier's internal data structures once a thread has timed out of a `pthread_barrier_wait()`. Even the inclusion of a special barrier reinitialization function would not help much since it is not clear how this function would affect the behavior of threads that reach the barrier between the original timeout and the call to the reinitialization function.

## Spin Locks

- Background

Spin locks represent an extremely low-level synchronization mechanism suitable primarily for use on shared memory multi-processors. It is typically an atomically modified Boolean value that is set to one when the lock is held and to zero when the lock is freed.

When a caller requests a spin lock that is already held, it typically spins in a loop testing whether the lock has become available. Such spinning wastes processor cycles so the lock should only be held for short durations and not across sleep/block operations. Callers should unlock spin locks before calling sleep operations.

Spin locks are available on a variety of systems. The functions included in POSIX.1-2008 are an attempt to standardize that existing practice.

- Lack of Timeout Feature

Alternate versions of most blocking routines have been provided to support watchdog timeouts. No alternate interface of this sort has been provided for spin locks for the following reasons:

- It is impossible to determine appropriate timeout intervals for spin locks in a portable manner. The amount of time one can expect to spend spin-waiting is inversely proportional to the degree of parallelism provided by the system.
- It can vary from a few cycles when each competing thread is running on its own processor, to an indefinite amount of time when all threads are multiplexed on a single processor (which is why spin locking is not advisable on uniprocessors).
- When used properly, the amount of time the calling thread spends waiting on a spin lock should be considerably less than the time required to set up a corresponding watchdog timer. Since the primary purpose of spin locks is to provide a low-overhead synchronization mechanism for multi-processors, the overhead of a timeout mechanism was deemed unacceptable.

It was also suggested that an additional `count` argument be provided (on the `pthread_spin_lock()` call) in lieu of a true timeout so that a spin lock call could fail gracefully if it was unable to apply the lock after `count` attempts. This idea was rejected because it is not existing practice. Furthermore, the same effect can be obtained with `pthread_spin_trylock()`, as illustrated below:

```

122489 int n = MAX_SPIN;
122490 while (--n >= 0)
122491 {
122492 if (!pthread_spin_try_lock(...))
122493 break;
122494 }
122495 if (n >= 0)
122496 {
122497 /* Successfully acquired the lock */
122498 }
122499 else
122500 {
122501 /* Unable to acquire the lock */
122502 }

```

- *process-shared* Attribute

The initialization functions associated with most POSIX synchronization objects (for example, mutexes, barriers, and read-write locks) take an attributes object with a *process-shared* attribute that specifies whether or not the object is to be shared across processes. In the draft corresponding to the first balloting round, two separate initialization functions are provided for spin locks, however: one for spin locks that were to be shared across processes (*spin\_init()*), and one for locks that were only used by multiple threads within a single process (*pthread\_spin\_init()*). This was done so as to keep the overhead associated with spin waiting to an absolute minimum. However, the balloting group requested that, since the overhead associated to a bit check was small, spin locks should be consistent with the rest of the synchronization primitives, and thus the *process-shared* attribute was introduced for spin locks.

- Spin Locks *versus* Mutexes

It has been suggested that mutexes are an adequate synchronization mechanism and spin locks are not necessary. Locking mechanisms typically must trade off the processor resources consumed while setting up to block the thread and the processor resources consumed by the thread while it is blocked. Spin locks require very little resources to set up the blocking of a thread. Existing practice is to simply loop, repeating the atomic locking operation until the lock is available. While the resources consumed to set up blocking of the thread are low, the thread continues to consume processor resources while it is waiting.

On the other hand, mutexes may be implemented such that the processor resources consumed to block the thread are large relative to a spin lock. After detecting that the mutex lock is not available, the thread must alter its scheduling state, add itself to a set of waiting threads, and, when the lock becomes available again, undo all of this before taking over ownership of the mutex. However, while a thread is blocked by a mutex, no processor resources are consumed.

Therefore, spin locks and mutexes may be implemented to have different characteristics. Spin locks may have lower overall overhead for very short-term blocking, and mutexes may have lower overall overhead when a thread will be blocked for longer periods of time. The presence of both interfaces allows implementations with these two different characteristics, both of which may be useful to a particular application.

It has also been suggested that applications can build their own spin locks from the *pthread\_mutex\_trylock()* function:

122537           while (pthread\_mutex\_trylock(&mutex));

122538           The apparent simplicity of this construct is somewhat deceiving, however. While the actual  
122539           wait is quite efficient, various guarantees on the integrity of mutex objects (for example,  
122540           priority inheritance rules) may add overhead to the successful path of the trylock  
122541           operation that is not required of spin locks. One could, of course, add an attribute to the  
122542           mutex to bypass such overhead, but the very act of finding and testing this attribute  
122543           represents more overhead than is found in the typical spin lock.

122544           The need to hold spin lock overhead to an absolute minimum also makes it impossible to  
122545           provide guarantees against starvation similar to those provided for mutexes or read-write  
122546           locks. The overhead required to implement such guarantees (for example, disabling  
122547           preemption before spinning) may well exceed the overhead of the spin wait itself by many  
122548           orders of magnitude. If a “safe” spin wait seems desirable, it can always be provided  
122549           (albeit at some performance cost) via appropriate mutex attributes.

## 122550           **Robust Mutexes**

122551           Robust mutexes are intended to protect applications that use mutexes to protect data shared  
122552           between different processes. If a process is terminated by a signal while a thread is holding a  
122553           mutex, there is no chance for the process to clean up after it. Waiters for the locked mutex might  
122554           wait indefinitely.

122555           With robust mutexes the problem can be solved: whenever a fatal signal terminates a process,  
122556           current or future waiters of the mutex are notified about this fact. The locking function provides  
122557           notification of this condition through the error condition [EOWNERDEAD]. A thread then has  
122558           the chance to clean up the state protected by the mutex and mark the state as consistent again by  
122559           a call to *pthread\_mutex\_consistent()*.

122560           Pre-existing implementations have used the semantics of robust mutexes for a variety of  
122561           situations, some of them not defined in the standard. Where a normally terminated process (i.e.,  
122562           when one thread calls *exit()*) causes notification of other waiters of robust mutexes if the mutex  
122563           is locked by any thread in the process. This behavior is defined in the standard and makes sense  
122564           because no thread other than the thread calling *exit()* has the chance to clean up its data.

122565           If a thread is terminated by cancellation or if it calls *pthread\_exit()*, the situation is different. In  
122566           both these situations the thread has the chance to clean up after itself by registering appropriate  
122567           cleanup handlers. There is no real reason to demand that other waiters for a robust mutex the  
122568           terminating thread owns are notified. The committee felt that this is actively encouraging bad  
122569           practice because programmers are tempted to rely on the robust mutex semantics instead of  
122570           correctly cleaning up after themselves.

122571           Therefore, the standard does not require notification of other waiters at the time a thread is  
122572           terminated while the process continues to run. The mutex is still recognized as being locked by  
122573           the process (with the thread gone it makes no sense to refer to the thread owning the mutex).  
122574           Therefore, a terminating process will cause notifications about the dead owner to be sent to all  
122575           waiters. This delay in the notification is not required, but programmers cannot rely on prompt  
122576           notification after a thread is terminated.

122577           For the same reason is it not required that an implementation supports robust mutexes that are  
122578           not shared between processes. If a robust mutex is used only within one process, all the cleanup  
122579           can be performed by the threads themselves by registering appropriate cleanup handlers. Fatal  
122580           signals are of no importance in this case because after the signal is delivered there is no thread  
122581           remaining to use the mutex.

122582           Some implementations might choose to support intra-process robust mutexes and they might  
122583           also send notification of a dead owner right after the previous owner died. But applications

122584 must not rely on this. Applications should only use robust mutexes for the purpose of handling  
 122585 fatal signals in situations where inter-process mutexes are in use.

## 122586 **Supported Threads Functions**

122587 On POSIX-conforming systems, the following symbolic constants are always conforming:

122588 `_POSIX_READER_WRITER_LOCKS`  
 122589 `_POSIX_THREADS`

122590 Therefore, the following threads functions are always supported:

|                                                   |                                              |
|---------------------------------------------------|----------------------------------------------|
| 122591 <code>pthread_atfork()</code>              | <code>pthread_mutex_destroy()</code>         |
| 122592 <code>pthread_attr_destroy()</code>        | <code>pthread_mutex_init()</code>            |
| 122593 <code>pthread_attr_getdetachstate()</code> | <code>pthread_mutex_lock()</code>            |
| 122594 <code>pthread_attr_getguardsize()</code>   | <code>pthread_mutex_trylock()</code>         |
| 122595 <code>pthread_attr_getschedparam()</code>  | <code>pthread_mutex_unlock()</code>          |
| 122596 <code>pthread_attr_init()</code>           | <code>pthread_mutexattr_destroy()</code>     |
| 122597 <code>pthread_attr_setdetachstate()</code> | <code>pthread_mutexattr_getpshared()</code>  |
| 122598 <code>pthread_attr_setguardsize()</code>   | <code>pthread_mutexattr_gettype()</code>     |
| 122599 <code>pthread_attr_setschedparam()</code>  | <code>pthread_mutexattr_init()</code>        |
| 122600 <code>pthread_cancel()</code>              | <code>pthread_mutexattr_setpshared()</code>  |
| 122601 <code>pthread_cleanup_pop()</code>         | <code>pthread_mutexattr_settype()</code>     |
| 122602 <code>pthread_cleanup_push()</code>        | <code>pthread_once()</code>                  |
| 122603 <code>pthread_cond_broadcast()</code>      | <code>pthread_rwlock_destroy()</code>        |
| 122604 <code>pthread_cond_destroy()</code>        | <code>pthread_rwlock_init()</code>           |
| 122605 <code>pthread_cond_init()</code>           | <code>pthread_rwlock_rdlock()</code>         |
| 122606 <code>pthread_cond_signal()</code>         | <code>pthread_rwlock_tryrdlock()</code>      |
| 122607 <code>pthread_cond_timedwait()</code>      | <code>pthread_rwlock_trywrlock()</code>      |
| 122608 <code>pthread_cond_wait()</code>           | <code>pthread_rwlock_unlock()</code>         |
| 122609 <code>pthread_condattr_destroy()</code>    | <code>pthread_rwlock_wrlock()</code>         |
| 122610 <code>pthread_condattr_getpshared()</code> | <code>pthread_rwlockattr_destroy()</code>    |
| 122611 <code>pthread_condattr_init()</code>       | <code>pthread_rwlockattr_getpshared()</code> |
| 122612 <code>pthread_condattr_setpshared()</code> | <code>pthread_rwlockattr_init()</code>       |
| 122613 <code>pthread_create()</code>              | <code>pthread_rwlockattr_setpshared()</code> |
| 122614 <code>pthread_detach()</code>              | <code>pthread_self()</code>                  |
| 122615 <code>pthread_equal()</code>               | <code>pthread_setcancelstate()</code>        |
| 122616 <code>pthread_exit()</code>                | <code>pthread_setcanceltype()</code>         |
| 122617 <code>pthread_getconcurrency()</code>      | <code>pthread_setconcurrency()</code>        |
| 122618 <code>pthread_getspecific()</code>         | <code>pthread_setspecific()</code>           |
| 122619 <code>pthread_join()</code>                | <code>pthread_sigmask()</code>               |
| 122620 <code>pthread_key_create()</code>          | <code>pthread_testcancel()</code>            |
| 122621 <code>pthread_key_delete()</code>          | <code>sigwait()</code>                       |
| 122622 <code>pthread_kill()</code>                |                                              |

On POSIX-conforming systems, the symbolic constant `_POSIX_THREAD_SAFE_FUNCTIONS` is always defined. Therefore, the following functions are always supported:

|        |                                 |                                 |
|--------|---------------------------------|---------------------------------|
| 122625 | <code>asctime_r()</code>        | <code>getpwuid_r()</code>       |
| 122626 | <code>ctime_r()</code>          | <code>gmtime_r()</code>         |
| 122627 | <code>flockfile()</code>        | <code>localtime_r()</code>      |
| 122628 | <code>ftrylockfile()</code>     | <code>putc_unlocked()</code>    |
| 122629 | <code>funlockfile()</code>      | <code>putchar_unlocked()</code> |
| 122630 | <code>getc_unlocked()</code>    | <code>rand_r()</code>           |
| 122631 | <code>getchar_unlocked()</code> | <code>readdir_r()</code>        |
| 122632 | <code>getgrgid_r()</code>       | <code>strerror_r()</code>       |
| 122633 | <code>getgrnam_r()</code>       | <code>strtok_r()</code>         |
| 122634 | <code>getpwnam_r()</code>       |                                 |

## 122635 Threads Extensions

The following extensions to the IEEE P1003.1c draft standard are now supported in POSIX.1-2008 as part of the alignment with the Single UNIX Specification:

- 122638 • Extended mutex attribute types
- 122639 • Read-write locks and attributes (also introduced by the IEEE Std 1003.1j-2000 amendment)
- 122640 • Thread concurrency level
- 122641 • Thread stack guard size
- 122642 • Parallel I/O
- 122643 • Robust mutexes

These extensions carefully follow the threads programming model specified in POSIX.1c. As with POSIX.1c, all the new functions return zero if successful; otherwise, an error number is returned to indicate the error.

The concept of attribute objects was introduced in POSIX.1c to allow implementations to extend POSIX.1-2008 without changing the existing interfaces. Attribute objects were defined for threads, mutexes, and condition variables. Attributes objects are defined as implementation-defined opaque types to aid extensibility, and functions are defined to allow attributes to be set or retrieved. This model has been followed when adding the new type attribute of **`pthread_mutexattr_t`** or the new read-write lock attributes object **`pthread_rwlockattr_t`**.

- 122653 • Extended Mutex Attributes

POSIX.1c defines a mutex attributes object as an implementation-defined opaque object of type **`pthread_mutexattr_t`**, and specifies a number of attributes which this object must have and a number of functions which manipulate these attributes. These attributes include *detachstate*, *inheritsched*, *schedparm*, *schedpolicy*, *contentionscope*, *stackaddr*, and *stacksize*.

The System Interfaces volume of POSIX.1-2008 specifies another mutex attribute called *type*. The *type* attribute allows applications to specify the behavior of mutex locking operations in situations where POSIX.1c behavior is undefined. The OSF DCE threads implementation, based on Draft 4 of POSIX.1c, specified a similar attribute. Note that the names of the attributes have changed somewhat from the OSF DCE threads implementation.

The System Interfaces volume of POSIX.1-2008 also extends the specification of the following POSIX.1c functions which manipulate mutexes:

|        |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|--------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 122667 | <code>pthread_mutex_lock()</code>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 122668 | <code>pthread_mutex_trylock()</code>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| 122669 | <code>pthread_mutex_unlock()</code>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 122670 | to take account of the new mutex attribute type and to specify behavior which was declared as undefined in POSIX.1c. How a calling thread acquires or releases a mutex now depends upon the mutex <i>type</i> attribute.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| 122671 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 122672 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 122673 | The <i>type</i> attribute can have the following values:                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| 122674 | PTHREAD_MUTEX_NORMAL                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| 122675 | Basic mutex with no specific error checking built in. Does not report a deadlock error.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 122676 | PTHREAD_MUTEX_RECURSIVE                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 122677 | Allows any thread to recursively lock a mutex. The mutex must be unlocked an equal                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 122678 | number of times to release the mutex.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 122679 | PTHREAD_MUTEX_ERRORCHECK                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| 122680 | Detects and reports simple usage errors; that is, an attempt to unlock a mutex that is                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| 122681 | not locked by the calling thread or that is not locked at all, or an attempt to relock a                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| 122682 | mutex the thread already owns.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 122683 | PTHREAD_MUTEX_DEFAULT                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 122684 | The default mutex type. May be mapped to any of the above mutex types or may be                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 122685 | an implementation-defined type.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 122686 | <i>Normal</i> mutexes do not detect deadlock conditions; for example, a thread will hang if it tries to relock a normal mutex that it already owns. Attempting to unlock a mutex locked by another thread, or unlocking an unlocked mutex, results in undefined behavior. Normal mutexes will usually be the fastest type of mutex available on a platform but provide the least error checking.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 122687 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 122688 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 122689 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 122690 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 122691 | <i>Recursive</i> mutexes are useful for converting old code where it is difficult to establish clear boundaries of synchronization. A thread can relock a recursive mutex without first unlocking it. The relocking deadlock which can occur with normal mutexes cannot occur with this type of mutex. However, multiple locks of a recursive mutex require the same number of unlocks to release the mutex before another thread can acquire the mutex. Furthermore, this type of mutex maintains the concept of an owner. Thus, a thread attempting to unlock a recursive mutex which another thread has locked returns with an error. A thread attempting to unlock a recursive mutex that is not locked returns with an error. Never use a recursive mutex with condition variables because the implicit unlock performed by <code>pthread_cond_wait()</code> or <code>pthread_cond_timedwait()</code> will not actually release the mutex if it had been locked multiple times. |
| 122692 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 122693 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 122694 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 122695 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 122696 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 122697 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 122698 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 122699 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 122700 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 122701 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 122702 | <i>Errorcheck</i> mutexes provide error checking and are useful primarily as a debugging aid. A thread attempting to relock an errorcheck mutex without first unlocking it returns with an error. Again, this type of mutex maintains the concept of an owner. Thus, a thread attempting to unlock an errorcheck mutex which another thread has locked returns with an error. A thread attempting to unlock an errorcheck mutex that is not locked also returns with an error. It should be noted that errorcheck mutexes will almost always be much slower than normal mutexes due to the extra state checks performed.                                                                                                                                                                                                                                                                                                                                                             |
| 122703 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 122704 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 122705 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 122706 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 122707 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 122708 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 122709 | The default mutex type provides implementation-defined error checking. The default mutex may be mapped to one of the other defined types or may be something entirely different. This enables each vendor to provide the mutex semantics which the vendor feels will be most useful to their target users. Most vendors will probably choose to make normal mutexes the default so as to give applications the benefit of the fastest type of                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 122710 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 122711 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 122712 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 122713 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |



mutexes available on their platform. Check your implementation's documentation.

An application developer can use any of the mutex types almost interchangeably as long as the application does not depend upon the implementation detecting (or failing to detect) any particular errors. Note that a recursive mutex can be used with condition variable waits as long as the application never recursively locks the mutex.

Two functions are provided for manipulating the *type* attribute of a mutex attributes object. This attribute is set or returned in the *type* parameter of these functions. The `pthread_mutexattr_settype()` function is used to set a specific type value while `pthread_mutexattr_gettype()` is used to return the type of the mutex. Setting the *type* attribute of a mutex attributes object affects only mutexes initialized using that mutex attributes object. Changing the *type* attribute does not affect mutexes previously initialized using that mutex attributes object.

- Read-Write Locks and Attributes

The read-write locks introduced have been harmonized with those in IEEE Std 1003.1j-2000; see also [Section B.2.9.6](#) (on page 3618).

Read-write locks (also known as reader-writer locks) allow a thread to exclusively lock some shared data while updating that data, or allow any number of threads to have simultaneous read-only access to the data.

Unlike a mutex, a read-write lock distinguishes between reading data and writing data. A mutex excludes all other threads. A read-write lock allows other threads access to the data, providing no thread is modifying the data. Thus, a read-write lock is less primitive than either a mutex-condition variable pair or a semaphore.

Application developers should consider using a read-write lock rather than a mutex to protect data that is frequently referenced but seldom modified. Most threads (readers) will be able to read the data without waiting and will only have to block when some other thread (a writer) is in the process of modifying the data. Conversely a thread that wants to change the data is forced to wait until there are no readers. This type of lock is often used to facilitate parallel access to data on multi-processor platforms or to avoid context switches on single processor platforms where multiple threads access the same data.

If a read-write lock becomes unlocked and there are multiple threads waiting to acquire the write lock, the implementation's scheduling policy determines which thread acquires the read-write lock for writing. If there are multiple threads blocked on a read-write lock for both read locks and write locks, it is unspecified whether the readers or a writer acquire the lock first. However, for performance reasons, implementations often favor writers over readers to avoid potential writer starvation.

A read-write lock object is an implementation-defined opaque object of type **pthread\_rwlock\_t** as defined in `<pthread.h>`. There are two different sorts of locks associated with a read-write lock: a read lock and a write lock.

The `pthread_rwlockattr_init()` function initializes a read-write lock attributes object with the default value for all the attributes defined in the implementation. After a read-write lock attributes object has been used to initialize one or more read-write locks, changes to the read-write lock attributes object, including destruction, do not affect previously initialized read-write locks.

Implementations must provide at least the read-write lock attribute *process-shared*. This attribute can have the following values:

|        |                                                                                                                      |
|--------|----------------------------------------------------------------------------------------------------------------------|
| 122759 | PTHREAD_PROCESS_SHARED                                                                                               |
| 122760 | Any thread of any process that has access to the memory where the read-write lock                                    |
| 122761 | resides can manipulate the read-write lock.                                                                          |
| 122762 | PTHREAD_PROCESS_PRIVATE                                                                                              |
| 122763 | Only threads created within the same process as the thread that initialized the read-                                |
| 122764 | write lock can manipulate the read-write lock. This is the default value.                                            |
| 122765 | The <i>pthread_rwlockattr_setpshared()</i> function is used to set the <i>process-shared</i> attribute of an         |
| 122766 | initialized read-write lock attributes object while the function                                                     |
| 122767 | <i>pthread_rwlockattr_getpshared()</i> obtains the current value of the <i>process-shared</i> attribute.             |
| 122768 | A read-write lock attributes object is destroyed using the <i>pthread_rwlockattr_destroy()</i>                       |
| 122769 | function. The effect of subsequent use of the read-write lock attributes object is undefined.                        |
| 122770 | A thread creates a read-write lock using the <i>pthread_rwlock_init()</i> function. The attributes                   |
| 122771 | of the read-write lock can be specified by the application developer; otherwise, the default                         |
| 122772 | implementation-defined read-write lock attributes are used if the pointer to the read-write                          |
| 122773 | lock attributes object is NULL. In cases where the default attributes are appropriate, the                           |
| 122774 | PTHREAD_RWLOCK_INITIALIZER macro can be used to initialize read-write locks.                                         |
| 122775 | A thread which wants to apply a read lock to the read-write lock can use either                                      |
| 122776 | <i>pthread_rwlock_rdlock()</i> or <i>pthread_rwlock_tryrdlock()</i> . If <i>pthread_rwlock_rdlock()</i> is used, the |
| 122777 | thread acquires a read lock if a writer does not hold the write lock and there are no writers                        |
| 122778 | blocked on the write lock. If a read lock is not acquired, the calling thread blocks until it                        |
| 122779 | can acquire a lock. However, if <i>pthread_rwlock_tryrdlock()</i> is used, the function returns                      |
| 122780 | immediately with the error [EBUSY] if any thread holds a write lock or there are blocked                             |
| 122781 | writers waiting for the write lock.                                                                                  |
| 122782 | A thread which wants to apply a write lock to the read-write lock can use either of two                              |
| 122783 | functions: <i>pthread_rwlock_wrlock()</i> or <i>pthread_rwlock_trywrlock()</i> . If <i>pthread_rwlock_wrlock()</i>   |
| 122784 | is used, the thread acquires the write lock if no other reader or writer threads hold the                            |
| 122785 | read-write lock. If the write lock is not acquired, the thread blocks until it can acquire the                       |
| 122786 | write lock. However, if <i>pthread_rwlock_trywrlock()</i> is used, the function returns                              |
| 122787 | immediately with the error [EBUSY] if any thread is holding either a read or a write lock.                           |
| 122788 | The <i>pthread_rwlock_unlock()</i> function is used to unlock a read-write lock object held by the                   |
| 122789 | calling thread. Results are undefined if the read-write lock is not held by the calling thread.                      |
| 122790 | If there are other read locks currently held on the read-write lock object, the read-write                           |
| 122791 | lock object remains in the read locked state but without the current thread as one of its                            |
| 122792 | owners. If this function releases the last read lock for this read-write lock object, the read-                      |
| 122793 | write lock object is put in the unlocked read state. If this function is called to release a write                   |
| 122794 | lock for this read-write lock object, the read-write lock object is put in the unlocked state.                       |
| 122795 | • Thread Concurrency Level                                                                                           |
| 122796 | On threads implementations that multiplex user threads onto a smaller set of kernel                                  |
| 122797 | execution entities, the system attempts to create a reasonable number of kernel execution                            |
| 122798 | entities for the application upon application startup.                                                               |
| 122799 | On some implementations, these kernel entities are retained by user threads that block in                            |
| 122800 | the kernel. Other implementations do not <i>timeslice</i> user threads so that multiple compute-                     |
| 122801 | bound user threads can share a kernel thread. On such implementations, some                                          |
| 122802 | applications may use up all the available kernel execution entities before their user-space                          |
| 122803 | threads are used up. The process may be left with user threads capable of doing work for                             |
| 122804 | the application but with no way to schedule them.                                                                    |
| 122805 | The <i>pthread_setconcurrency()</i> function enables an application to request more kernel                           |

entities; that is, specify a desired concurrency level. However, this function merely provides a hint to the implementation. The implementation is free to ignore this request or to provide some other number of kernel entities. If an implementation does not multiplex user threads onto a smaller number of kernel execution entities, the *pthread\_setconcurrency()* function has no effect.

The *pthread\_setconcurrency()* function may also have an effect on implementations where the kernel mode and user mode schedulers cooperate to ensure that ready user threads are not prevented from running by other threads blocked in the kernel.

The *pthread\_getconcurrency()* function always returns the value set by a previous call to *pthread\_setconcurrency()*. However, if *pthread\_setconcurrency()* was not previously called, this function returns zero to indicate that the threads implementation is maintaining the concurrency level.

- Thread Stack Guard Size

DCE threads introduced the concept of a “thread stack guard size”. Most thread implementations add a region of protected memory to a thread’s stack, commonly known as a “guard region”, as a safety measure to prevent stack pointer overflow in one thread from corrupting the contents of another thread’s stack. The default size of the guard regions attribute is {PAGESIZE} bytes and is implementation-defined.

Some application developers may wish to change the stack guard size. When an application creates a large number of threads, the extra page allocated for each stack may strain system resources. In addition to the extra page of memory, the kernel’s memory manager has to keep track of the different protections on adjoining pages. When this is a problem, the application developer may request a guard size of 0 bytes to conserve system resources by eliminating stack overflow protection.

Conversely an application that allocates large data structures such as arrays on the stack may wish to increase the default guard size in order to detect stack overflow. If a thread allocates two pages for a data array, a single guard page provides little protection against thread stack overflows since the thread can corrupt adjoining memory beyond the guard page.

The System Interfaces volume of POSIX.1-2008 defines a new attribute of a thread attributes object; that is, the *guardsize* attribute which allows applications to specify the size of the guard region of a thread’s stack.

Two functions are provided for manipulating a thread’s stack guard size. The *pthread\_attr\_setguardsize()* function sets the thread *guardsize* attribute, and the *pthread\_attr\_getguardsize()* function retrieves the current value.

An implementation may round up the requested guard size to a multiple of the configurable system variable {PAGESIZE}. In this case, *pthread\_attr\_getguardsize()* returns the guard size specified by the previous *pthread\_attr\_setguardsize()* function call and not the rounded up value.

If an application is managing its own thread stacks using the *stackaddr* attribute, the *guardsize* attribute is ignored and no stack overflow protection is provided. In this case, it is the responsibility of the application to manage stack overflow along with stack allocation.

- Parallel I/O

Suppose two or more threads independently issue read requests on the same file. To read specific data from a file, a thread must first call *lseek()* to seek to the proper offset in the file, and then call *read()* to retrieve the required data. If more than one thread does this at the same time, the first thread may complete its seek call, but before it gets a chance to

122853 issue its read call a second thread may complete its seek call, resulting in the first thread  
 122854 accessing incorrect data when it issues its read call. One workaround is to lock the file  
 122855 descriptor while seeking and reading or writing, but this reduces parallelism and adds  
 122856 overhead.

122857 Instead, the System Interfaces volume of POSIX.1-2008 provides two functions to make  
 122858 seek/read and seek/write operations atomic. The file descriptor's current offset is  
 122859 unchanged, thus allowing multiple read and write operations to proceed in parallel. This  
 122860 improves the I/O performance of threaded applications. The *pread()* function is used to do  
 122861 an atomic read of data from a file into a buffer. Conversely, the *pwrite()* function does an  
 122862 atomic write of data from a buffer to a file.

#### 122863 B.2.9.1 Thread-Safety

122864 All functions required by POSIX.1-2008 need to be thread-safe. Implementations have to  
 122865 provide internal synchronization when necessary in order to achieve this goal. In certain cases—  
 122866 for example, most floating-point implementations—context switch code may have to manage  
 122867 the writable shared state.

122868 While a read from a pipe of `[PIPE_MAX]*2` bytes may not generate a single atomic and thread-  
 122869 safe stream of bytes, it should generate “several” (individually atomic) thread-safe streams of  
 122870 bytes. Similarly, while reading from a terminal device may not generate a single atomic and  
 122871 thread-safe stream of bytes, it should generate some finite number of (individually atomic) and  
 122872 thread-safe streams of bytes. That is, concurrent calls to read for a pipe, FIFO, or terminal device  
 122873 are not allowed to result in corrupting the stream of bytes or other internal data. However,  
 122874 *read()*, in these cases, is not required to return a single contiguous and atomic stream of bytes.

122875 It is not required that all functions provided by POSIX.1-2008 be either async-cancel-safe or  
 122876 async-signal-safe.

122877 As it turns out, some functions are inherently not thread-safe; that is, their interface  
 122878 specifications preclude async-signal-safety. For example, some functions (such as *asctime()*)  
 122879 return a pointer to a result stored in memory space allocated by the function on a per-process  
 122880 basis. Such a function is not thread-safe, because its result can be overwritten by successive  
 122881 invocations. Other functions, while not inherently non-thread-safe, may be implemented in  
 122882 ways that lead to them not being thread-safe. For example, some functions (such as *rand()*) store  
 122883 state information (such as a seed value, which survives multiple function invocations) in  
 122884 memory space allocated by the function on a per-process basis. The implementation of such a  
 122885 function is not thread-safe if the implementation fails to synchronize invocations of the function  
 122886 and thus fails to protect the state information. The problem is that when the state information is  
 122887 not protected, concurrent invocations can interfere with one another (for example, applications  
 122888 using *rand()* may see the same seed value).

#### 122889 Thread-Safety and Locking of Existing Functions

122890 Originally, POSIX.1 was not designed to work in a multi-threaded environment, and some  
 122891 implementations of some existing functions will not work properly when executed concurrently.  
 122892 To provide routines that will work correctly in an environment with threads (“thread-safe”), two  
 122893 problems need to be solved:

- 122894 1. Routines that maintain or return pointers to static areas internal to the routine (which  
 122895 may now be shared) need to be modified. The routines *ttyname()* and *localtime()* are  
 122896 examples.
- 122897 2. Routines that access data space shared by more than one thread need to be modified. The  
 122898 *malloc()* function and the *stdio* family routines are examples.

There are a variety of constraints on these changes. The first is compatibility with the existing versions of these functions—non-thread-safe functions will continue to be in use for some time, as the original interfaces are used by existing code. Another is that the new thread-safe versions of these functions represent as small a change as possible over the familiar interfaces provided by the existing non-thread-safe versions. The new interfaces should be independent of any particular threads implementation. In particular, they should be thread-safe without depending on explicit thread-specific memory. Finally, there should be minimal performance penalty due to the changes made to the functions.

It is intended that the list of functions from POSIX.1 that cannot be made thread-safe and for which corrected versions are provided be complete.

#### *Thread-Safety and Locking Solutions*

Many of the POSIX.1 functions were thread-safe and did not change at all. However, some functions (for example, the math functions typically found in **libm**) are not thread-safe because of writable shared global state. For instance, in IEEE Std 754-1985 floating-point implementations, the computation modes and flags are global and shared.

Some functions are not thread-safe because a particular implementation is not reentrant, typically because of a non-essential use of static storage. These require only a new implementation.

Thread-safe libraries are useful in a wide range of parallel (and asynchronous) programming environments, not just within pthreads. In order to be used outside the context of pthreads, however, such libraries still have to use some synchronization method. These could either be independent of the pthread synchronization operations, or they could be a subset of the pthread interfaces. Either method results in thread-safe library implementations that can be used without the rest of pthreads.

Some functions, such as the *stdio* family interface and dynamic memory allocation functions such as *malloc()*, are inter-dependent routines that share resources (for example, buffers) across related calls. These require synchronization to work correctly, but they do not require any change to their external (user-visible) interfaces.

In some cases, such as *getc()* and *putc()*, adding synchronization is likely to create an unacceptable performance impact. In this case, slower thread-safe synchronized functions are to be provided, but the original, faster (but unsafe) functions (which may be implemented as macros) are retained under new names. Some additional special-purpose synchronization facilities are necessary for these macros to be usable in multi-threaded programs. This also requires changes in **<stdio.h>**.

The other common reason that functions are unsafe is that they return a pointer to static storage, making the functions non-thread-safe. This has to be changed, and there are three natural choices:

1. Return a pointer to thread-specific storage

This could incur a severe performance penalty on those architectures with a costly implementation of the thread-specific data interface.

A variation on this technique is to use *malloc()* to allocate storage for the function output and return a pointer to this storage. This technique may also have an undesirable performance impact, however, and a simplistic implementation requires that the user program explicitly free the storage object when it is no longer needed. This technique is used by some existing POSIX.1 functions. With careful implementation for infrequently used functions, there may be little or no performance or storage penalty, and the maintenance of already-standardized interfaces is a significant benefit.

## 2. Return the actual value computed by the function

This technique can only be used with functions that return pointers to structures—routines that return character strings would have to wrap their output in an enclosing structure in order to return the output on the stack. There is also a negative performance impact inherent in this solution in that the output value has to be copied twice before it can be used by the calling function: once from the called routine's local buffers to the top of the stack, then from the top of the stack to the assignment target. Finally, many older compilers cannot support this technique due to a historical tendency to use internal static buffers to deliver the results of structure-valued functions.

## 3. Have the caller pass the address of a buffer to contain the computed value

The only disadvantage of this approach is that extra arguments have to be provided by the calling program. It represents the most efficient solution to the problem, however, and, unlike the *malloc()* technique, it is semantically clear.

There are some routines (often groups of related routines) whose interfaces are inherently non-thread-safe because they communicate across multiple function invocations by means of static memory locations. The solution is to redesign the calls so that they are thread-safe, typically by passing the needed data as extra parameters. Unfortunately, this may require major changes to the interface as well.

A floating-point implementation using IEEE Std 754-1985 is a case in point. A less problematic example is the *rand48* family of pseudo-random number generators. The functions *getgrgid()*, *getgrnam()*, *getpwnam()*, and *getpwuid()* are another such case.

The problems with *errno* are discussed in [Alternative Solutions for Per-Thread \*errno\*](#) (on page 3534).

Some functions can be thread-safe or not, depending on their arguments. These include the *tmpnam()* and *ctermid()* functions. These functions have pointers to character strings as arguments. If the pointers are not NULL, the functions store their results in the character string; however, if the pointers are NULL, the functions store their results in an area that may be static and thus subject to overwriting by successive calls. These should only be called by multi-thread applications when their arguments are non-NULL.

### *Asynchronous Safety and Thread-Safety*

A floating-point implementation has many modes that effect rounding and other aspects of computation. Functions in some math library implementations may change the computation modes for the duration of a function call. If such a function call is interrupted by a signal or cancellation, the floating-point state is not required to be protected.

There is a significant cost to make floating-point operations async-cancel-safe or async-signal-safe; accordingly, neither form of async safety is required.

### *Functions Returning Pointers to Static Storage*

For those functions that are not thread-safe because they return values in fixed size statically allocated structures, alternate “\_r” forms are provided that pass a pointer to an explicit result structure. Those that return pointers into library-allocated buffers have forms provided with explicit buffer and length parameters.

For functions that return pointers to library-allocated buffers, it makes sense to provide “\_r” versions that allow the application control over allocation of the storage in which results are returned. This allows the state used by these functions to be managed on an application-specific basis, supporting per-thread, per-process, or other application-specific sharing relationships.

Early proposals had provided “\_r” versions for functions that returned pointers to variable-size

122992 buffers without providing a means for determining the required buffer size. This would have  
 122993 made using such functions exceedingly clumsy, potentially requiring iteratively calling them  
 122994 with increasingly larger guesses for the amount of storage required. Hence, *sysconf()* variables  
 122995 have been provided for such functions that return the maximum required buffer size.

122996 Thus, the rule that has been followed by POSIX.1-2008 when adapting single-threaded non-  
 122997 thread-safe functions is as follows: all functions returning pointers to library-allocated storage  
 122998 should have “\_r” versions provided, allowing the application control over the storage  
 122999 allocation. Those with variable-sized return values accept both a buffer address and a length  
 123000 parameter. The *sysconf()* variables are provided to supply the appropriate buffer sizes when  
 123001 required. Implementors are encouraged to apply the same rule when adapting their own  
 123002 existing functions to a pthreads environment.

#### 123003 B.2.9.2 Thread IDs

123004 Separate applications should communicate through well-defined interfaces and should not  
 123005 depend on each other’s implementation. For example, if a programmer decides to rewrite the  
 123006 *sort* utility using multiple threads, it should be easy to do this so that the interface to the *sort*  
 123007 utility does not change. Consider that if the user causes SIGINT to be generated while the *sort*  
 123008 utility is running, keeping the same interface means that the entire *sort* utility is killed, not just  
 123009 one of its threads. As another example, consider a realtime application that manages a reactor.  
 123010 Such an application may wish to allow other applications to control the priority at which it  
 123011 watches the control rods. One technique to accomplish this is to write the ID of the thread  
 123012 watching the control rods into a file and allow other programs to change the priority of that  
 123013 thread as they see fit. A simpler technique is to have the reactor process accept IPCs  
 123014 (Interprocess Communication messages) from other processes, telling it at a semantic level what  
 123015 priority the program should assign to watching the control rods. This allows the programmer  
 123016 greater flexibility in the implementation. For example, the programmer can change the  
 123017 implementation from having one thread per rod to having one thread watching all of the rods  
 123018 without changing the interface. Having threads live inside the process means that the  
 123019 implementation of a process is invisible to outside processes (excepting debuggers and system  
 123020 management tools).

123021 Threads do not provide a protection boundary. Every thread model allows threads to share  
 123022 memory with other threads and encourages this sharing to be widespread. This means that one  
 123023 thread can wipe out memory that is needed for the correct functioning of other threads that are  
 123024 sharing its memory. Consequently, providing each thread with its own user and/or group IDs  
 123025 would not provide a protection boundary between threads sharing memory.

123026 Some applications make the assumption that the implementation can always detect invalid uses  
 123027 of thread IDs of type **pthread\_t**. This is an invalid assumption. Specifically, if **pthread\_t** is  
 123028 defined as a pointer type, no access check needs to be performed before using the ID.

123029 As with other interfaces that take pointer parameters, the outcome of passing an invalid  
 123030 parameter can result in an invalid memory reference or an attempt to access an undefined  
 123031 portion of a memory object, cause signals to be sent (SIGSEGV or SIGBUS) and possible  
 123032 termination of the process. This is a similar case to passing an invalid buffer pointer to *read()*.  
 123033 Some implementations might implement *read()* as a system call and set an [EFAULT] error  
 123034 condition. Other implementations might contain parts of *read()* at user level and the first  
 123035 attempt to access data at an invalid reference will cause a signal to be sent instead.

123036 If an implementation detects use of a thread ID after the end of its lifetime, it is recommended  
 123037 that the function should fail and report an [ESRCH] error. This does not imply that  
 123038 implementations are required to return in this case. It is legitimate behavior to send an “invalid  
 123039 memory reference” signal (SIGSEGV or SIGBUS). It is the application’s responsibility to use only

123040 valid thread IDs and to keep track of the lifetime of the underlying threads.

123041 B.2.9.3 *Thread Mutexes*

123042 There is no additional rationale provided for this section.

123043 B.2.9.4 *Thread Scheduling*

123044 • Scheduling Implementation Models

123045 The following scheduling implementation models are presented in terms of threads and  
123046 “kernel entities”. This is to simplify exposition of the models, and it does not imply that  
123047 an implementation actually has an identifiable “kernel entity”.

123048 A kernel entity is not defined beyond the fact that it has scheduling attributes that are used  
123049 to resolve contention with other kernel entities for execution resources. A kernel entity  
123050 may be thought of as an envelope that holds a thread or a separate kernel thread. It is not a  
123051 conventional process, although it shares with the process the attribute that it has a single  
123052 thread of control; it does not necessarily imply an address space, open files, and so on. It is  
123053 better thought of as a primitive facility upon which conventional processes and threads  
123054 may be constructed.

123055 — System Thread Scheduling Model

123056 This model consists of one thread per kernel entity. The kernel entity is solely  
123057 responsible for scheduling thread execution on one or more processors. This model  
123058 schedules all threads against all other threads in the system using the scheduling  
123059 attributes of the thread.

123060 — Process Scheduling Model

123061 A generalized process scheduling model consists of two levels of scheduling. A  
123062 threads library creates a pool of kernel entities, as required, and schedules threads to  
123063 run on them using the scheduling attributes of the threads. Typically, the size of the  
123064 pool is a function of the simultaneously runnable threads, not the total number of  
123065 threads. The kernel then schedules the kernel entities onto processors according to  
123066 their scheduling attributes, which are managed by the threads library. This set model  
123067 potentially allows a wide range of mappings between threads and kernel entities.

123068 • System and Process Scheduling Model Performance

123069 There are a number of important implications on the performance of applications using  
123070 these scheduling models. The process scheduling model potentially provides lower  
123071 overhead for making scheduling decisions, since there is no need to access kernel-level  
123072 information or functions and the set of schedulable entities is smaller (only the threads  
123073 within the process).

123074 On the other hand, since the kernel is also making scheduling decisions regarding the  
123075 system resources under its control (for example, CPU(s), I/O devices, memory), decisions  
123076 that do not take thread scheduling parameters into account can result in unspecified  
123077 delays for realtime application threads, causing them to miss maximum response time  
123078 limits.

123079 • Rate Monotonic Scheduling

123080 Rate monotonic scheduling was considered, but rejected for standardization in the context  
123081 of pthreads. A sporadic server policy is included.



- Scheduling Options

In POSIX.1-2008, the basic thread scheduling functions are defined under the threads functionality, so that they are required of all threads implementations. However, there are no specific scheduling policies required by this functionality to allow for conforming thread implementations that are not targeted to realtime applications.

Specific standard scheduling policies are defined to be under the Thread Execution Scheduling option, and they are specifically designed to support realtime applications by providing predictable resource-sharing sequences. The name of this option was chosen to emphasize that this functionality is defined as appropriate for realtime applications that require simple priority-based scheduling.

It is recognized that these policies are not necessarily satisfactory for some multi-processor implementations, and work is ongoing to address a wider range of scheduling behaviors. The interfaces have been chosen to create abundant opportunity for future scheduling policies to be implemented and standardized based on this interface. In order to standardize a new scheduling policy, all that is required (from the standpoint of thread scheduling attributes) is to define a new policy name, new members of the thread attributes object, and functions to set these members when the scheduling policy is equal to the new value.

### **Scheduling Contention Scope**

In order to accommodate the requirement for realtime response, each thread has a scheduling contention scope attribute. Threads with a system scheduling contention scope have to be scheduled with respect to all other threads in the system. These threads are usually bound to a single kernel entity that reflects their scheduling attributes and are directly scheduled by the kernel.

Threads with a process scheduling contention scope need be scheduled only with respect to the other threads in the process. These threads may be scheduled within the process onto a pool of kernel entities. The implementation is also free to bind these threads directly to kernel entities and let them be scheduled by the kernel. Process scheduling contention scope allows the implementation the most flexibility and is the default if both contention scopes are supported and none is specified.

Thus, the choice by implementors to provide one or the other (or both) of these scheduling models is driven by the need of their supported application domains for worst-case (that is, realtime) response, or average-case (non-realtime) response.

### **Scheduling Allocation Domain**

The SCHED\_FIFO and SCHED\_RR scheduling policies take on different characteristics on a multi-processor. Other scheduling policies are also subject to changed behavior when executed on a multi-processor. The concept of scheduling allocation domain determines the set of processors on which the threads of an application may run. By considering the application's processor scheduling allocation domain for its threads, scheduling policies can be defined in terms of their behavior for varying processor scheduling allocation domain values. It is conceivable that not all scheduling allocation domain sizes make sense for all scheduling policies on all implementations. The concept of scheduling allocation domain, however, is a useful tool for the description of multi-processor scheduling policies.

The "process control" approach to scheduling obtains significant performance advantages from dynamic scheduling allocation domain sizes when it is applicable.

Non-Uniform Memory Access (NUMA) multi-processors may use a system scheduling structure

that involves reassignment of threads among scheduling allocation domains. In NUMA machines, a natural model of scheduling is to match scheduling allocation domains to clusters of processors. Load balancing in such an environment requires changing the scheduling allocation domain to which a thread is assigned.

### Scheduling Documentation

Implementation-provided scheduling policies need to be completely documented in order to be useful. This documentation includes a description of the attributes required for the policy, the scheduling interaction of threads running under this policy and all other supported policies, and the effects of all possible values for processor scheduling allocation domain. Note that for the implementor wishing to be minimally-compliant, it is (minimally) acceptable to define the behavior as undefined.

### Scheduling Contention Scope Attribute

The scheduling contention scope defines how threads compete for resources. Within POSIX.1-2008, scheduling contention scope is used to describe only how threads are scheduled in relation to one another in the system. That is, either they are scheduled against all other threads in the system ("system scope") or only against those threads in the process ("process scope"). In fact, scheduling contention scope may apply to additional resources, including virtual timers and profiling, which are not currently considered by POSIX.1-2008.

### Mixed Scopes

If only one scheduling contention scope is supported, the scheduling decision is straightforward. To perform the processor scheduling decision in a mixed scope environment, it is necessary to map the scheduling attributes of the thread with process-wide contention scope to the same attribute space as the thread with system-wide contention scope.

Since a conforming implementation has to support one and may support both scopes, it is useful to discuss the effects of such choices with respect to example applications. If an implementation supports both scopes, mixing scopes provides a means of better managing system-level (that is, kernel-level) and library-level resources. In general, threads with system scope will require the resources of a separate kernel entity in order to guarantee the scheduling semantics. On the other hand, threads with process scope can share the resources of a kernel entity while maintaining the scheduling semantics.

The application is free to create threads with dedicated kernel resources, and other threads that multiplex kernel resources. Consider the example of a window server. The server allocates two threads per widget: one thread manages the widget user interface (including drawing), while the other thread takes any required application action. This allows the widget to be "active" while the application is computing. A screen image may be built from thousands of widgets. If each of these threads had been created with system scope, then most of the kernel-level resources might be wasted, since only a few widgets are active at any one time. In addition, mixed scope is particularly useful in a window server where one thread with high priority and system scope handles the mouse so that it tracks well. As another example, consider a database server. For each of the hundreds or thousands of clients supported by a large server, an equivalent number of threads will have to be created. If each of these threads were system scope, the consequences would be the same as for the window server example above. However, the server could be constructed so that actual retrieval of data is done by several dedicated threads. Dedicated threads that do work for all clients frequently justify the added expense of system scope. If it were not permissible to mix system and process threads in the same process, this type of solution would not be possible.

## Dynamic Thread Scheduling Parameters Access

In many time-constrained applications, there is no need to change the scheduling attributes dynamically during thread or process execution, since the general use of these attributes is to reflect directly the time constraints of the application. Since these time constraints are generally imposed to meet higher-level system requirements, such as accuracy or availability, they frequently should remain unchanged during application execution.

However, there are important situations in which the scheduling attributes should be changed. Generally, this will occur when external environmental conditions exist in which the time constraints change. Consider, for example, a space vehicle major mode change, such as the change from ascent to descent mode, or the change from the space environment to the atmospheric environment. In such cases, the frequency with which many of the sensors or actuators need to be read or written will change, which will necessitate a priority change. In other cases, even the existence of a time constraint might be temporary, necessitating not just a priority change, but also a policy change for ongoing threads or processes. For this reason, it is critical that the interface should provide functions to change the scheduling parameters dynamically, but, as with many of the other realtime functions, it is important that applications use them properly to avoid the possibility of unnecessarily degrading performance.

In providing functions for dynamically changing the scheduling behavior of threads, there were two options: provide functions to get and set the individual scheduling parameters of threads, or provide a single interface to get and set all the scheduling parameters for a given thread simultaneously. Both approaches have merit. Access functions for individual parameters allow simpler control of thread scheduling for simple thread scheduling parameters. However, a single function for setting all the parameters for a given scheduling policy is required when first setting that scheduling policy. Since the single all-encompassing functions are required, it was decided to leave the interface as minimal as possible. Note that simpler functions (such as *pthread\_setprio()* for threads running under the priority-based schedulers) can be easily defined in terms of the all-encompassing functions.

If the *pthread\_setschedparam()* function executes successfully, it will have set all of the scheduling parameter values indicated in *param*; otherwise, none of the scheduling parameters will have been modified. This is necessary to ensure that the scheduling of this and all other threads continues to be consistent in the presence of an erroneous scheduling parameter.

The [EPERM] error value is included in the list of possible *pthread\_setschedparam()* error returns as a reflection of the fact that the ability to change scheduling parameters increases risks to the implementation and application performance if the scheduling parameters are changed improperly. For this reason, and based on some existing practice, it was felt that some implementations would probably choose to define specific permissions for changing either a thread's own or another thread's scheduling parameters. POSIX.1-2008 does not include portable methods for setting or retrieving permissions, so any such use of permissions is completely unspecified.

## Mutex Initialization Scheduling Attributes

In a priority-driven environment, a direct use of traditional primitives like mutexes and condition variables can lead to unbounded priority inversion, where a higher priority thread can be blocked by a lower priority thread, or set of threads, for an unbounded duration of time. As a result, it becomes impossible to guarantee thread deadlines. Priority inversion can be bounded and minimized by the use of priority inheritance protocols. This allows thread deadlines to be guaranteed even in the presence of synchronization requirements.

Two useful but simple members of the family of priority inheritance protocols are the basic priority inheritance protocol and the priority ceiling protocol emulation. Under the Basic

123222 Priority Inheritance protocol (governed by the Non-Robust Mutex Priority Inheritance option), a  
 123223 thread that is blocking higher priority threads executes at the priority of the highest priority  
 123224 thread that it blocks. This simple mechanism allows priority inversion to be bounded by the  
 123225 duration of critical sections and makes timing analysis possible.

123226 Under the Priority Ceiling Protocol Emulation protocol (governed by the Thread Priority  
 123227 Protection option), each mutex has a priority ceiling, usually defined as the priority of the  
 123228 highest priority thread that can lock the mutex. When a thread is executing inside critical  
 123229 sections, its priority is unconditionally increased to the highest of the priority ceilings of all the  
 123230 mutexes owned by the thread. This protocol has two very desirable properties in uni-processor  
 123231 systems. First, a thread can be blocked by a lower priority thread for at most the duration of one  
 123232 single critical section. Furthermore, when the protocol is correctly used in a single processor, and  
 123233 if threads do not become blocked while owning mutexes, mutual deadlocks are prevented.

123234 The priority ceiling emulation can be extended to multiple processor environments, in which  
 123235 case the values of the priority ceilings will be assigned depending on the kind of mutex that is  
 123236 being used: local to only one processor, or global, shared by several processors. Local priority  
 123237 ceilings will be assigned the usual way, equal to the priority of the highest priority thread that  
 123238 may lock that mutex. Global priority ceilings will usually be assigned a priority level higher  
 123239 than all the priorities assigned to any of the threads that reside in the involved processors to  
 123240 avoid the effect called remote blocking.

#### 123241 **Change the Priority Ceiling of a Mutex**

123242 In order for the priority protect protocol to exhibit its desired properties of bounding priority  
 123243 inversion and avoidance of deadlock, it is critical that the ceiling priority of a mutex be the same  
 123244 as the priority of the highest thread that can ever hold it, or higher. Thus, if the priorities of the  
 123245 threads using such mutexes never change dynamically, there is no need ever to change the  
 123246 priority ceiling of a mutex.

123247 However, if a major system mode change results in an altered response time requirement for one  
 123248 or more application threads, their priority has to change to reflect it. It will occasionally be the  
 123249 case that the priority ceilings of mutexes held also need to change. While changing priority  
 123250 ceilings should generally be avoided, it is important that POSIX.1-2008 provide these interfaces  
 123251 for those cases in which it is necessary.

#### 123252 *B.2.9.5 Thread Cancellation*

123253 Many existing threads packages have facilities for canceling an operation or canceling a thread.  
 123254 These facilities are used for implementing user requests (such as the CANCEL button in a  
 123255 window-based application), for implementing OR parallelism (for example, telling the other  
 123256 threads to stop working once one thread has found a forced mate in a parallel chess program), or  
 123257 for implementing the ABORT mechanism in Ada.

123258 POSIX programs traditionally have used the signal mechanism combined with either *longjmp()*  
 123259 or polling to cancel operations. Many POSIX programmers have trouble using these facilities to  
 123260 solve their problems efficiently in a single-threaded process. With the introduction of threads,  
 123261 these solutions become even more difficult to use.

123262 The main issues with implementing a cancellation facility are specifying the operation to be  
 123263 canceled, cleanly releasing any resources allocated to that operation, controlling when the target  
 123264 notices that it has been canceled, and defining the interaction between asynchronous signals and  
 123265 cancellation.

### Specifying the Operation to Cancel

Consider a thread that calls through five distinct levels of program abstraction and then, inside the lowest-level abstraction, calls a function that suspends the thread. (An abstraction boundary is a layer at which the client of the abstraction sees only the service being provided and can remain ignorant of the implementation. Abstractions are often layered, each level of abstraction being a client of the lower-level abstraction and implementing a higher-level abstraction.) Depending on the semantics of each abstraction, one could imagine wanting to cancel only the call that causes suspension, only the bottom two levels, or the operation being done by the entire thread. Canceling operations at a finer grain than the entire thread is difficult because threads are active and they may be run in parallel on a multi-processor. By the time one thread can make a request to cancel an operation, the thread performing the operation may have completed that operation and gone on to start another operation whose cancellation is not desired. Thread IDs are not reused until the thread has exited, and either it was created with the *Attr detachstate* attribute set to *PTHREAD\_CREATE\_DETACHED* or the *pthread\_join()* or *pthread\_detach()* function has been called for that thread. Consequently, a thread cancellation will never be misdirected when the thread terminates. For these reasons, the canceling of operations is done at the granularity of the thread. Threads are designed to be inexpensive enough so that a separate thread may be created to perform each separately cancelable operation; for example, each possibly long running user request.

For cancellation to be used in existing code, cancellation scopes and handlers will have to be established for code that needs to release resources upon cancellation, so that it follows the programming discipline described in the text.

### A Special Signal Versus a Special Interface

Two different mechanisms were considered for providing the cancellation interfaces. The first was to provide an interface to direct signals at a thread and then to define a special signal that had the required semantics. The other alternative was to use a special interface that delivered the correct semantics to the target thread.

The solution using signals produced a number of problems. It required the implementation to provide cancellation in terms of signals whereas a perfectly valid (and possibly more efficient) implementation could have both layered on a low-level set of primitives. There were so many exceptions to the special signal (it cannot be used with *kill()*, no POSIX.1 interfaces can be used with it) that it was clearly not a valid signal. Its semantics on delivery were also completely different from any existing POSIX.1 signal. As such, a special interface that did not mandate the implementation and did not confuse the semantics of signals and cancellation was felt to be the better solution.

### Races Between Cancellation and Resuming Execution

Due to the nature of cancellation, there is generally no synchronization between the thread requesting the cancellation of a blocked thread and events that may cause that thread to resume execution. For this reason, and because excess serialization hurts performance, when both an event that a thread is waiting for has occurred and a cancellation request has been made and cancellation is enabled, POSIX.1-2008 explicitly allows the implementation to choose between returning from the blocking call or acting on the cancellation request.

## Interaction of Cancellation with Asynchronous Signals

A typical use of cancellation is to acquire a lock on some resource and to establish a cancellation cleanup handler for releasing the resource when and if the thread is canceled.

A correct and complete implementation of cancellation in the presence of asynchronous signals requires considerable care. An implementation has to push a cancellation cleanup handler on the cancellation cleanup stack while maintaining the integrity of the stack data structure. If an asynchronously-generated signal is posted to the thread during a stack operation, the signal handler cannot manipulate the cancellation cleanup stack. As a consequence, asynchronous signal handlers may not cancel threads or otherwise manipulate the cancellation state of a thread. Threads may, of course, be canceled by another thread that used a *sigwait()* function to wait synchronously for an asynchronous signal.

In order for cancellation to function correctly, it is required that asynchronous signal handlers not change the cancellation state. This requires that some elements of existing practice, such as using *longjmp()* to exit from an asynchronous signal handler implicitly, be prohibited in cases where the integrity of the cancellation state of the interrupt thread cannot be ensured.

## Thread Cancellation Overview

- Cancelability States

The three possible cancelability states (disabled, deferred, and asynchronous) are encoded into two separate bits ((disable, enable) and (deferred, asynchronous)) to allow them to be changed and restored independently. For instance, short code sequences that will not block sometimes disable cancelability on entry and restore the previous state upon exit. Likewise, long or unbounded code sequences containing no convenient explicit cancellation points will sometimes set the cancelability type to asynchronous on entry and restore the previous value upon exit.

- Cancellation Points

Cancellation points are points inside of certain functions where a thread has to act on any pending cancellation request when cancelability is enabled. For functions in the “shall occur” list, a cancellation check must be performed on every call regardless of whether, absent the cancellation, the call would have blocked. For functions in the “may occur” list, a cancellation check may be performed on some calls but not others; i.e., whether or not a cancellation point occurs when one of these functions is being executed can depend on current conditions.

The idea was considered of allowing implementations to define whether blocking calls such as *read()* should be cancellation points. It was decided that it would adversely affect the design of conforming applications if blocking calls were not cancellation points because threads could be left blocked in an uncancelable state.

There are several important blocking routines that are specifically not made cancellation points:

— *pthread\_mutex\_lock()*

If *pthread\_mutex\_lock()* were a cancellation point, every routine that called it would also become a cancellation point (that is, any routine that touched shared state would automatically become a cancellation point). For example, *malloc()*, *free()*, and *rand()* would become cancellation points under this scheme. Having too many cancellation points makes programming very difficult, leading to either much disabling and restoring of cancelability or much difficulty in trying to arrange for reliable cleanup at every possible place.

123354 Since `pthread_mutex_lock()` is not a cancellation point, threads could result in being  
 123355 blocked uninterruptibly for long periods of time if mutexes were used as a general  
 123356 synchronization mechanism. As this is normally not acceptable, mutexes should only  
 123357 be used to protect resources that are held for small fixed lengths of time where not  
 123358 being able to be canceled will not be a problem. Resources that need to be held  
 123359 exclusively for long periods of time should be protected with condition variables.

123360 — `pthread_barrier_wait()`

123361 Canceling a barrier wait will render a barrier unusable. Similar to a barrier timeout  
 123362 (which the standard developers rejected), there is no way to guarantee the  
 123363 consistency of a barrier's internal data structures if a barrier wait is canceled.

123364 — `pthread_spin_lock()`

123365 As with mutexes, spin locks should only be used to protect resources that are held for  
 123366 small fixed lengths of time where not being cancelable will not be a problem.

123367 Every library routine should specify whether or not it includes any cancellation points.  
 123368 Typically, only those routines that may block or compute indefinitely need to include  
 123369 cancellation points.

123370 Correctly coded routines only reach cancellation points after having set up a cancellation  
 123371 cleanup handler to restore invariants if the thread is canceled at that point. Being  
 123372 cancelable only at specified cancellation points allows programmers to keep track of  
 123373 actions needed in a cancellation cleanup handler more easily. A thread should only be  
 123374 made asynchronously cancelable when it is not in the process of acquiring or releasing  
 123375 resources or otherwise in a state from which it would be difficult or impossible to recover.

#### 123376 • Thread Cancellation Cleanup Handlers

123377 The cancellation cleanup handlers provide a portable mechanism, easy to implement, for  
 123378 releasing resources and restoring invariants. They are easier to use than signal handlers  
 123379 because they provide a stack of cancellation cleanup handlers rather than a single handler,  
 123380 and because they have an argument that can be used to pass context information to the  
 123381 handler.

123382 The alternative to providing these simple cancellation cleanup handlers (whose only use is  
 123383 for cleaning up when a thread is canceled) is to define a general exception package that  
 123384 could be used for handling and cleaning up after hardware traps and software-detected  
 123385 errors. This was too far removed from the charter of providing threads to handle  
 123386 asynchrony. However, it is an explicit goal of POSIX.1-2008 to be compatible with existing  
 123387 exception facilities and languages having exceptions.

123388 The interaction of this facility and other procedure-based or language-level exception  
 123389 facilities is unspecified in this version of POSIX.1-2008. However, it is intended that it be  
 123390 possible for an implementation to define the relationship between these cancellation  
 123391 cleanup handlers and Ada, C++, or other language-level exception handling facilities.

123392 It was suggested that the cancellation cleanup handlers should also be called when the  
 123393 process exits or calls the `exec` function. This was rejected partly due to the performance  
 123394 problem caused by having to call the cancellation cleanup handlers of every thread before  
 123395 the operation could continue. The other reason was that the only state expected to be  
 123396 cleaned up by the cancellation cleanup handlers would be the intraprocess state. Any  
 123397 handlers that are to clean up the interprocess state would be registered with `atexit()`. There  
 123398 is the orthogonal problem that the `exec` functions do not honor the `atexit()` handlers, but  
 123399 resolving this is beyond the scope of POSIX.1-2008.

- Async-Cancel Safety

A function is said to be async-cancel-safe if it is written in such a way that entering the function with asynchronous cancelability enabled will not cause any invariants to be violated, even if a cancellation request is delivered at any arbitrary instruction. Functions that are async-cancel-safe are often written in such a way that they need to acquire no resources for their operation and the visible variables that they may write are strictly limited.

Any routine that gets a resource as a side-effect cannot be made async-cancel-safe (for example, *malloc()*). If such a routine were called with asynchronous cancelability enabled, it might acquire the resource successfully, but as it was returning to the client, it could act on a cancellation request. In such a case, the application would have no way of knowing whether the resource was acquired or not.

Indeed, because many interesting routines cannot be made async-cancel-safe, most library routines in general are not async-cancel-safe. Every library routine should specify whether or not it is async-cancel safe so that programmers know which routines can be called from code that is asynchronously cancelable.

IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/8 is applied, adding the *pselect()* function to the list of functions with cancellation points.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/5 is applied, adding the *fdatasync()* function into the table of functions that shall have cancellation points.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/6 is applied, adding the numerous functions into the table of functions that may have cancellation points.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/7 is applied, clarifying the requirements in Thread Cancellation Cleanup Handlers.

#### B.2.9.6 Thread Read-Write Locks

##### Background

Read-write locks are often used to allow parallel access to data on multi-processors, to avoid context switches on uni-processors when multiple threads access the same data, and to protect data structures that are frequently accessed (that is, read) but rarely updated (that is, written). The in-core representation of a file system directory is a good example of such a data structure. One would like to achieve as much concurrency as possible when searching directories, but limit concurrent access when adding or deleting files.

Although read-write locks can be implemented with mutexes and condition variables, such implementations are significantly less efficient than is possible. Therefore, this synchronization primitive is included in POSIX.1-2008 for the purpose of allowing more efficient implementations in multi-processor systems.

##### Queuing of Waiting Threads

The *pthread\_rwlock\_unlock()* function description states that one writer or one or more readers must acquire the lock if it is no longer held by any thread as a result of the call. However, the function does not specify which thread(s) acquire the lock, unless the Thread Execution Scheduling option is supported.

The standard developers considered the issue of scheduling with respect to the queuing of threads blocked on a read-write lock. The question turned out to be whether POSIX.1-2008 should require priority scheduling of read-write locks for threads whose execution scheduling



123444 policy is priority-based (for example, SCHED\_FIFO or SCHED\_RR). There are tradeoffs  
 123445 between priority scheduling, the amount of concurrency achievable among readers, and the  
 123446 prevention of writer and/or reader starvation.

123447 For example, suppose one or more readers hold a read-write lock and the following threads  
 123448 request the lock in the listed order:

123449 pthread\_rwlock\_wrlock() - Low priority thread writer\_a  
 123450 pthread\_rwlock\_rdlock() - High priority thread reader\_a  
 123451 pthread\_rwlock\_rdlock() - High priority thread reader\_b  
 123452 pthread\_rwlock\_rdlock() - High priority thread reader\_c

123453 When the lock becomes available, should *writer\_a* block the high priority readers? Or, suppose a  
 123454 read-write lock becomes available and the following are queued:

123455 pthread\_rwlock\_rdlock() - Low priority thread reader\_a  
 123456 pthread\_rwlock\_rdlock() - Low priority thread reader\_b  
 123457 pthread\_rwlock\_rdlock() - Low priority thread reader\_c  
 123458 pthread\_rwlock\_wrlock() - Medium priority thread writer\_a  
 123459 pthread\_rwlock\_rdlock() - High priority thread reader\_d

123460 If priority scheduling is applied then *reader\_d* would acquire the lock and *writer\_a* would block  
 123461 the remaining readers. But should the remaining readers also acquire the lock to increase  
 123462 concurrency? The solution adopted takes into account that when the Thread Execution  
 123463 Scheduling option is supported, high priority threads may in fact starve low priority threads  
 123464 (the application developer is responsible in this case for designing the system in such a way that  
 123465 this starvation is avoided). Therefore, POSIX.1-2008 specifies that high priority readers take  
 123466 precedence over lower priority writers. However, to prevent writer starvation from threads of  
 123467 the same or lower priority, writers take precedence over readers of the same or lower priority.

123468 Priority inheritance mechanisms are non-trivial in the context of read-write locks. When a high  
 123469 priority writer is forced to wait for multiple readers, for example, it is not clear which subset of  
 123470 the readers should inherit the writer's priority. Furthermore, the internal data structures that  
 123471 record the inheritance must be accessible to all readers, and this implies some sort of  
 123472 serialization that could negate any gain in parallelism achieved through the use of multiple  
 123473 readers in the first place. Finally, existing practice does not support the use of priority  
 123474 inheritance for read-write locks. Therefore, no specification of priority inheritance or priority  
 123475 ceiling is attempted. If reliable priority-scheduled synchronization is absolutely required, it can  
 123476 always be obtained through the use of mutexes.

## 123477 **Comparison to fcntl() Locks**

123478 The read-write locks and the *fcntl()* locks in POSIX.1-2008 share a common goal: increasing  
 123479 concurrency among readers, thus increasing throughput and decreasing delay.

123480 However, the read-write locks have two features not present in the *fcntl()* locks. First, under  
 123481 priority scheduling, read-write locks are granted in priority order. Second, also under priority  
 123482 scheduling, writer starvation is prevented by giving writers preference over readers of equal or  
 123483 lower priority.

123484 Also, read-write locks can be used in systems lacking a file system, such as those conforming to  
 123485 the minimal realtime system profile of IEEE Std 1003.13-1998.

## 123486 History of Resolution Issues

123487 Based upon some balloting objections, early drafts specified the behavior of threads waiting on a  
 123488 read-write lock during the execution of a signal handler, as if the thread had not called the lock  
 123489 operation. However, this specified behavior would require implementations to establish  
 123490 internal signal handlers even though this situation would be rare, or never happen for many  
 123491 programs. This would introduce an unacceptable performance hit in comparison to the little  
 123492 additional functionality gained. Therefore, the behavior of read-write locks and signals was  
 123493 reverted back to its previous mutex-like specification.

### 123494 B.2.9.7 Thread Interactions with Regular File Operations

123495 There is no additional rationale provided for this section.

### 123496 B.2.9.8 Use of Application-Managed Thread Stacks

123497 IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/8 is applied, adding this new section. It  
 123498 was added to make it clear that the current standard does not allow an application to determine  
 123499 when a stack can be reclaimed. This may be addressed in a future version.

## 123500 B.2.10 Sockets

123501 The base document for the sockets interfaces in POSIX.1-2008 is the XNS, Issue 5.2 specification.  
 123502 This was primarily chosen as it aligns with IPv6. Additional material has been added from  
 123503 IEEE Std 1003.1g-2000, notably socket concepts, raw sockets, the *pselect()* function, the  
 123504 *socketmark()* function, and the `<sys/select.h>` header.

### 123505 B.2.10.1 Address Families

123506 There is no additional rationale provided for this section.

### 123507 B.2.10.2 Addressing

123508 There is no additional rationale provided for this section.

### 123509 B.2.10.3 Protocols

123510 There is no additional rationale provided for this section.

### 123511 B.2.10.4 Routing

123512 There is no additional rationale provided for this section.

### 123513 B.2.10.5 Interfaces

123514 There is no additional rationale provided for this section.

123515 B.2.10.6 *Socket Types*

123516 The type **socklen\_t** was invented to cover the range of implementations seen in the field. The  
123517 intent of **socklen\_t** is to be the type for all lengths that are naturally bounded in size; that is, that  
123518 they are the length of a buffer which cannot sensibly become of massive size: network addresses,  
123519 host names, string representations of these, ancillary data, control messages, and socket options  
123520 are examples. Truly boundless sizes are represented by **size\_t** as in *read()*, *write()*, and so on.

123521 All **socklen\_t** types were originally (in BSD UNIX) of type **int**. During the development of  
123522 POSIX.1-2008, it was decided to change all buffer lengths to **size\_t**, which appears at face value  
123523 to make sense. When dual mode 32/64-bit systems came along, this choice unnecessarily  
123524 complicated system interfaces because **size\_t** (with **long**) was a different size under ILP32 and  
123525 LP64 models. Reverting to **int** would have happened except that some implementations had  
123526 already shipped 64-bit-only interfaces. The compromise was a type which could be defined to be  
123527 any size by the implementation: **socklen\_t**.

123528 B.2.10.7 *Socket I/O Mode*

123529 There is no additional rationale provided for this section.

123530 B.2.10.8 *Socket Owner*

123531 There is no additional rationale provided for this section.

123532 B.2.10.9 *Socket Queue Limits*

123533 There is no additional rationale provided for this section.

123534 B.2.10.10 *Pending Error*

123535 There is no additional rationale provided for this section.

123536 B.2.10.11 *Socket Receive Queue*

123537 There is no additional rationale provided for this section.

123538 B.2.10.12 *Socket Out-of-Band Data State*

123539 There is no additional rationale provided for this section.

123540 B.2.10.13 *Connection Indication Queue*

123541 There is no additional rationale provided for this section.

123542 B.2.10.14 *Signals*

123543 There is no additional rationale provided for this section.

123544 *B.2.10.15 Asynchronous Errors*

123545 There is no additional rationale provided for this section.

123546 *B.2.10.16 Use of Options*

123547 There is no additional rationale provided for this section.

123548 *B.2.10.17 Use of Sockets for Local UNIX Connections*

123549 There is no additional rationale provided for this section.

123550 *B.2.10.18 Use of Sockets over Internet Protocols*

123551 A raw socket allows privileged users direct access to a protocol; for example, raw access to the  
123552 IP and ICMP protocols is possible through raw sockets. Raw sockets are intended for  
123553 knowledgeable applications that wish to take advantage of some protocol feature not directly  
123554 accessible through the other sockets interfaces.

123555 *B.2.10.19 Use of Sockets over Internet Protocols Based on IPv4*

123556 There is no additional rationale provided for this section.

123557 *B.2.10.20 Use of Sockets over Internet Protocols Based on IPv6*

123558 The Open Group Base Resolution bwg2001-012 is applied, clarifying that IPv6 implementations  
123559 are required to support use of AF\_INET6 sockets over IPv4.

123560 **B.2.11 Tracing**

123561 The organization of the tracing rationale differs from the traditional rationale in that this tracing  
123562 rationale text is written against the trace interface as a whole, rather than against the individual  
123563 components of the trace interface or the normative section in which those components are  
123564 defined. Therefore the sections below do not parallel the sections of normative text in  
123565 POSIX.1-2008.

123566 *B.2.11.1 Objectives*

123567 The intended uses of tracing are application-system debugging during system development, as a  
123568 “flight recorder” for maintenance of fielded systems, and as a performance measurement tool.  
123569 In all of these intended uses, the vendor-supplied computer system and its software are, for this  
123570 discussion, assumed error-free; the intent being to debug the user-written and/or third-party  
123571 application code, and their interactions. Clearly, problems with the vendor-supplied system and  
123572 its software will be uncovered from time to time, but this is a byproduct of the primary activity,  
123573 debugging user code.

123574 Another need for defining a trace interface in POSIX stems from the objective to provide an  
123575 efficient portable way to perform benchmarks. Existing practice shows that such interfaces are  
123576 commonly used in a variety of systems but with little commonality. As part of the benchmarking  
123577 needs, two aspects within the trace interface must be considered.

123578 The first, and perhaps more important one, is the qualitative aspect.

123579 The second is the quantitative aspect.

123580 • Qualitative Aspect

123581 To better understand this aspect, let us consider an example. Suppose that you want to  
 123582 organize a number of actions to be performed during the day. Some of these actions are  
 123583 known at the beginning of the day. Some others, which may be more or less important,  
 123584 will be triggered by reading your mail. During the day you will make some phone calls  
 123585 and synchronously receive some more information. Finally you will receive asynchronous  
 123586 phone calls that also will trigger actions. If you, or somebody else, examines your day at  
 123587 work, you, or he, can discover that you have not efficiently organized your work. For  
 123588 instance, relative to the phone calls you made, would it be preferable to make some of  
 123589 these early in the morning? Or to delay some others until the end of the day? Relative to  
 123590 the phone calls you have received, you might find that somebody you called in the  
 123591 morning has called you 10 times while you were performing some important work. To  
 123592 examine, afterwards, your day at work, you record in sequence all the trace events relative  
 123593 to your work. This should give you a chance of organizing your next day at work.

123594 This is the qualitative aspect of the trace interface. The user of a system needs to keep a  
 123595 trace of particular points the application passes through, so that he can eventually make  
 123596 some changes in the application and/or system configuration, to give the application a  
 123597 chance of running more efficiently.

123598 • Quantitative Aspect

123599 This aspect concerns primarily realtime applications, where missed deadlines can be  
 123600 undesirable. Although there are, in POSIX.1-2008, some interfaces useful for such  
 123601 applications (timeouts, execution time monitoring, and so on), there are no APIs to aid in  
 123602 the tuning of a realtime application's behavior (**timespec** in timeouts, length of message  
 123603 queues, duration of driver interrupt service routine, and so on). The tuning of an  
 123604 application needs a means of recording timestamped important trace events during  
 123605 execution in order to analyze offline, and eventually, to tune some realtime features  
 123606 (redesign the system with less functionalities, readjust timeouts, redesign driver interrupts,  
 123607 and so on).

123608 **Detailed Objectives**

123609 Objectives were defined to build the trace interface and are kept for historical interest. Although  
 123610 some objectives are not fully respected in this trace interface, the concept of the POSIX trace  
 123611 interface assumes the following points:

- 123612 1. It must be possible to trace both system and user trace events concurrently.
- 123613 2. It must be possible to trace per-process trace events and also to trace system trace events  
 123614 which are unrelated to any particular process. A per-process trace event is either user-  
 123615 initiated or system-initiated.
- 123616 3. It must be possible to control tracing on a per-process basis from either inside or outside  
 123617 the process.
- 123618 4. It must be possible to control tracing on a per-thread basis from inside the enclosing  
 123619 process.
- 123620 5. Trace points must be controllable by trace event type ID from inside and outside of the  
 123621 process. Multiple trace points can have the same trace event type ID, and will be  
 123622 controlled jointly.

- 123623 6. Recording of trace events is dependent on both trace event type ID and the  
123624 process/thread. Both must be enabled in order to record trace events. System trace events  
123625 may or may not be handled differently.
- 123626 7. The API must not mandate the ability to control tracing for more than one process at the  
123627 same time.
- 123628 8. There is no objective for trace control on anything bigger than a process; for example,  
123629 group or session.
- 123630 9. Trace propagation and control:
  - 123631 a. Trace propagation across *fork()* is optional; the default is to not trace a child  
123632 process.
  - 123633 b. Trace control must span *pthread\_create()* operations; that is, if a process is being  
123634 traced, any thread will be traced as well if this thread allows tracing. The default is  
123635 to allow tracing.
- 123636 10. Trace control must not span *exec* or *posix\_spawn()* operations.
- 123637 11. A triggering API is not required. The triggering API is the ability to command or stop  
123638 tracing based on the occurrence of a specific trace event other than a  
123639 POSIX\_TRACE\_START trace event or a POSIX\_TRACE\_STOP trace event.
- 123640 12. Trace log entries must have timestamps of implementation-defined resolution.  
123641 Implementations are exhorted to support at least microsecond resolution. When a trace  
123642 log entry is retrieved, it must have timestamp, PC address, PID, and TID of the entity that  
123643 generated the trace event.
- 123644 13. Independently developed code should be able to use trace facilities without coordination  
123645 and without conflict.
- 123646 14. Even if the trace points in the trace calls are not unique, the trace log entries (after any  
123647 processing) must be uniquely identified as to trace point.
- 123648 15. There must be a standard API to read the trace stream.
- 123649 16. The format of the trace stream and the trace log is opaque and unspecified.
- 123650 17. It must be possible to read a completed trace, if recorded on some suitable non-volatile  
123651 storage, even subsequent to a power cycle or subsequent cold boot of the system.
- 123652 18. Support of analysis of a trace log while it is being formed is implementation-defined.
- 123653 19. The API must allow the application to write trace stream identification information into  
123654 the trace stream and to be able to retrieve it, without it being overwritten by trace entries,  
123655 even if the trace stream is full.
- 123656 20. It must be possible to specify the destination of trace data produced by trace events.
- 123657 21. It must be possible to have different trace streams, and for the tracing enabled by one  
123658 trace stream to be completely independent of the tracing of another trace stream.
- 123659 22. It must be possible to trace events from threads in different CPUs.
- 123660 23. The API must support one or more trace streams per-system, and one or more trace  
123661 streams per-process, up to an implementation-defined set of per-system and per-process  
123662 maximums.
- 123663 24. It must be possible to determine the order in which the trace events happened, without  
123664 necessarily depending on the clock, up to an implementation-defined time resolution.

- 123665 25. For performance reasons, the trace event point call(s) must be implementable as a macro  
123666 (see the ISO POSIX-1: 1996 standard, 1.3.4, Statement 2).
- 123667 26. POSIX.1-2008 must not define the trace points which a conforming system must  
123668 implement, except for trace points used in the control of tracing.
- 123669 27. The APIs must be thread-safe, and trace points should be lock-free (that is, not require a  
123670 lock to gain exclusive access to some resource).
- 123671 28. The user-provided information associated with a trace event is variable-sized, up to some  
123672 maximum size.
- 123673 29. Bounds on record and trace stream sizes:
- 123674 a. The API must permit the application to declare the upper bounds on the length of  
123675 an application data record. The system must return the limit it used. The limit used  
123676 may be smaller than requested.
- 123677 b. The API must permit the application to declare the upper bounds on the size of  
123678 trace streams. The system must return the limit it used. The limit used may be  
123679 different, either larger or smaller, than requested.
- 123680 30. The API must be able to pass any fundamental data type, and a structured data type  
123681 composed only of fundamental types. The API must be able to pass data by reference,  
123682 given only as an address and a length. Fundamental types are the POSIX.1 types (see the  
123683 `<sys/types.h>` header) plus those defined in the ISO C standard.
- 123684 31. The API must apply the POSIX notions of ownership and permission to recorded trace  
123685 data, corresponding to the sources of that data.

### Comments on Objectives

**Note:** In the following comments, numbers in square brackets refer to the above objectives.

It is necessary to be able to obtain a trace stream for a complete activity. Thus there is a requirement to be able to trace both application and system trace events. A per-process trace event is either user-initiated, like the `write()` function, or system-initiated, like a timer expiration. There is also a need to be able to trace the activity of an entire process even when it has threads in multiple CPUs. To avoid excess trace activity, it is necessary to be able to control tracing on a trace event type basis.

[Objectives 1,2,5,22]

There is a need to be able to control tracing on a per-process basis, both from inside and outside the process; that is, a process can start a trace activity on itself or any other process. There is also the perceived need to allow the definition of a maximum number of trace streams per system.

[Objectives 3,23]

From within a process, it is necessary to be able to control tracing on a per-thread basis. This provides an additional filtering capability to keep the amount of traced data to a minimum. It also allows for less ambiguity as to the origin of trace events. It is recognized that thread-level control is only valid from within the process itself. It is also desirable to know the maximum number of trace streams per process that can be started. The API should not require thread synchronization or mandate priority inversions that would cause the thread to block. However, the API must be thread-safe.

[Objectives 4,23,24,27]

There was no perceived objective to control tracing on anything larger than a process; for example, a group or session. Also, the ability to start or stop a trace activity on multiple processes atomically may be very difficult or cumbersome in some implementations.

[Objectives 6,8]

123711 It is also necessary to be able to control tracing by trace event type identifier, sometimes called a  
123712 trace hook ID. However, there is no mandated set of system trace events, since such trace points  
123713 are implementation-defined. The API must not require from the operating system facilities that  
123714 are not standard.  
123715 [Objectives 6,26]

123716 Trace control must span *fork()* and *pthread\_create()*. If not, there will be no way to ensure that an  
123717 application's activity is entirely traced. The newly forked child would not be able to turn on its  
123718 tracing until after it obtained control after the fork, and trace control externally would be even  
123719 more problematic.  
123720 [Objective 9]

123721 Since *exec* and *posix\_spawn()* represent a complete change in the execution of a task (a new  
123722 program), trace control need not persist over an *exec* or *posix\_spawn()*.  
123723 [Objective 10]

123724 Where trace activities are started on multiple processes, these trace activities should not interfere  
123725 with each other.  
123726 [Objective 21]

123727 There is no need for a triggering objective, primarily for performance reasons; see also [Section](#)  
123728 [B.2.11.8](#) (on page 3645), rationale on triggering.  
123729 [Objective 11]

123730 It must be possible to determine the origin of each traced event. The process and thread  
123731 identifiers for each trace event are needed. Also there was a perceived need for a user-specifiable  
123732 origin, but it was felt that this would create too much overhead.  
123733 [Objectives 12,14]

123734 An allowance must be made for trace points to come embedded in software components from  
123735 several different sources and vendors without requiring coordination.  
123736 [Objective 13]

123737 There is a requirement to be able to uniquely identify trace points that may have the same trace  
123738 stream identifier. This is only necessary when a trace report is produced.  
123739 [Objectives 12,14]

123740 Tracing is a very performance-sensitive activity, and will therefore likely be implemented at a  
123741 low level within the system. Hence the interface must not mandate any particular buffering or  
123742 storage method. Therefore, a standard API is needed to read a trace stream. Also the interface  
123743 must not mandate the format of the trace data, and the interface must not assume a trace storage  
123744 method. Due to the possibility of a monolithic kernel and the possible presence of multiple  
123745 processes capable of running trace activities, the two kinds of trace events may be stored in two  
123746 separate streams for performance reasons. A mandatory dump mechanism, common in some  
123747 existing practice, has been avoided to allow the implementation of this set of functions on small  
123748 realtime profiles for which the concept of a file system is not defined. The trace API calls should  
123749 be implemented as macros.  
123750 [Objectives 15,16,25,30]

123751 Since a trace facility is a valuable service tool, the output (or log) of a completed trace stream  
123752 that is written to permanent storage must be readable on other systems of the type that  
123753 produced the trace log. Note that there is no objective to be able to interpret a trace log that was  
123754 not successfully completed.  
123755 [Objectives 17,18,19]

123756 For trace streams written to permanent storage, a way to specify the destination of the trace  
123757 stream is needed.  
123758 [Objective 20]



There is a requirement to be able to depend on the ordering of trace events up to some implementation-defined time interval. For example, there is a need to know the time period during which, if trace events are closer together, their ordering is unspecified. Events that occur within an interval smaller than this resolution may or may not be read back in the correct order. [Objective 24]

The application should be able to know how much data can be traced. When trace event types can be filtered, the application should be able to specify the approximate maximum amount of data that will be traced in a trace event so resources can be more efficiently allocated. [Objectives 28,29]

Users should not be able to trace data to which they would not normally have access. System trace events corresponding to a process/thread should be associated with the ownership of that process/thread. [Objective 31]

#### 123772 B.2.11.2 Trace Model

##### 123773 Introduction

The model is based on two base entities: the "Trace Stream" and the "Trace Log", and a recorded unit called the "Trace Event". The possibility of using Trace Streams and Trace Logs separately gives two use dimensions and solves both the performance issue and the full-information system issue. In the case of a trace stream without log, specific information, although reduced in quantity, is required to be registered, in a possibly small realtime system, with as little overhead as possible. The Trace Log option has been added for small realtime systems. In the case of a trace stream with log, considerable complex application-specific information needs to be collected.

##### 123782 Trace Model Description

The trace model can be examined for three different subfunctions: Application Instrumentation, Trace Operation Control, and Trace Analysis.

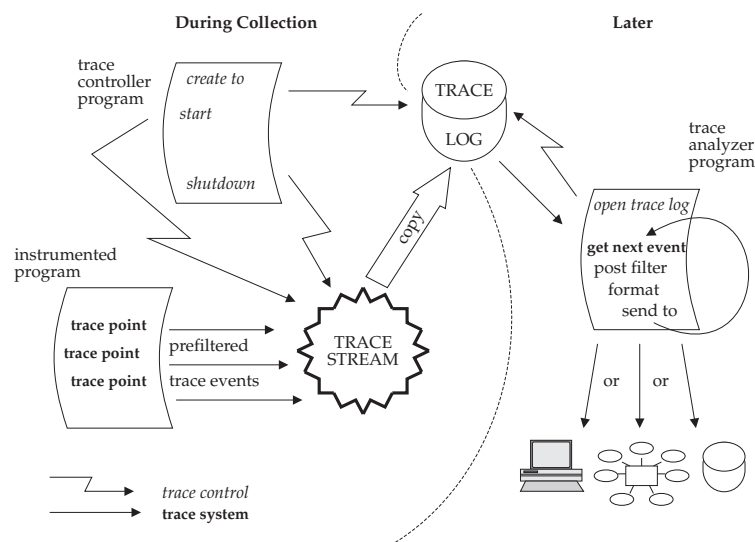


**Figure B-2** Trace System Overview: for Offline Analysis

Each of these subfunctions requires specific characteristics of the trace mechanism API.

- Application Instrumentation

When instrumenting an application, the programmer is not concerned about the future use of the trace events in the trace stream or the trace log, the full policy of the trace stream, or the eventual pre-filtering of trace events. But he is concerned about the correct determination of the specific trace event type identifier, regardless of how many independent libraries are used in the same user application; see [Figure B-2](#) (on page 3627) and [Figure B-3](#) (on page 3629).

This trace API provides the necessary operations to accomplish this subfunction. This is done by providing functions to associate a programmer-defined name with an implementation-defined trace event type identifier (see the *posix\_trace\_eventid\_open()* function), and to send this trace event into a potential trace stream (see the *posix\_trace\_event()* function).

- Trace Operation Control

When controlling the recording of trace events in a trace stream, the programmer is concerned with the correct initialization of the trace mechanism (that is, the sizing of the trace stream), the correct retention of trace events in a permanent storage, the correct dynamic recording of trace events, and so on.

This trace API provides the necessary material to permit this efficiently. This is done by providing functions to initialize a new trace stream, and optionally a trace log:

- Trace Stream Attributes Object Initialization (see *posix\_trace\_attr\_init()*)
- Functions to Retrieve or Set Information About a Trace Stream (see *posix\_trace\_attr\_getgenversion()*)
- Functions to Retrieve or Set the Behavior of a Trace Stream (see *posix\_trace\_attr\_getinherited()*)
- Functions to Retrieve or Set Trace Stream Size Attributes (see *posix\_trace\_attr\_getmaxusereventsize()*)
- Trace Stream Initialization, Flush, and Shutdown from a Process (see *posix\_trace\_create()*)
- Clear Trace Stream and Trace Log (see *posix\_trace\_clear()*)

To select the trace event types that are to be traced:

- Manipulate Trace Event Type Identifier (see *posix\_trace\_trid\_eventid\_open()*)
- Iterate over a Mapping of Trace Event Type (see *posix\_trace\_eventtypelist\_getnext\_id()*)
- Manipulate Trace Event Type Sets (see *posix\_trace\_eventset\_empty()*)
- Set Filter of an Initialized Trace Stream (see *posix\_trace\_set\_filter()*)

To control the execution of an active trace stream:

- Trace Start and Stop (see *posix\_trace\_start()*)
- Functions to Retrieve the Trace Attributes or Trace Statuses (see *posix\_trace\_get\_attr()*)

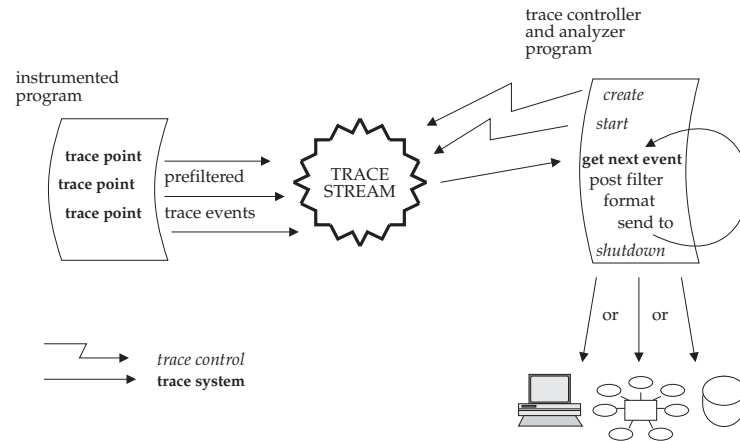


**Figure B-3** Trace System Overview: for Online Analysis

### • Trace Analysis

Once correctly recorded, on permanent storage or not, an ultimate activity consists of the analysis of the recorded information. If the recorded data is on permanent storage, a specific open operation is required to associate a trace stream to a trace log.

The first intent of the group was to request the presence of a system identification structure in the trace stream attribute. This was, for the application, to allow some portable way to process the recorded information. However, there is no requirement that the **utsname** structure, on which this system identification was based, be portable from one machine to another, so the contents of the attribute cannot be interpreted correctly by an application conforming to POSIX.1-2008.

This modification has been incorporated and requests that some unspecified information be recorded in the trace log in order to fail opening it if the analysis process and the controller process were running in different types of machine, but does not request that this information be accessible to the application. This modification has implied a modification in the `posix_trace_open()` function error code returns.

This trace API provides functions to:

- Extract trace stream identification attributes (see `posix_trace_attr_getgenversion()`)
- Extract trace stream behavior attributes (see `posix_trace_attr_getinherited()`)
- Extract trace event, stream, and log size attributes (see `posix_trace_attr_getmaxusersize()`)
- Look up trace event type names (see `posix_trace_eventid_get_name()`)
- Iterate over trace event type identifiers (see `posix_trace_eventtypelist_getnext_id()`)
- Open, rewind, and close a trace log (see `posix_trace_open()`)
- Read trace stream attributes and status (see `posix_trace_get_attr()`)
- Read trace events (see `posix_trace_getnext_event()`)

Due to the following two reasons:

1. The requirement that the trace system must not add unacceptable overhead to the traced process and so that the trace event point execution must be fast
2. The traced application does not care about tracing errors

the trace system cannot return any internal error to the application. Internal error conditions can range from unrecoverable errors that will force the active trace stream to abort, to small errors that can affect the quality of tracing without aborting the trace stream. The group decided to define a system trace event to report to the analysis process such internal errors. It is not the intention of POSIX.1-2008 to require an implementation to report an internal error that corrupts or terminates tracing operation. The implementor is free to decide which internal documented errors, if any, the trace system is able to report.

### States of a Trace Stream

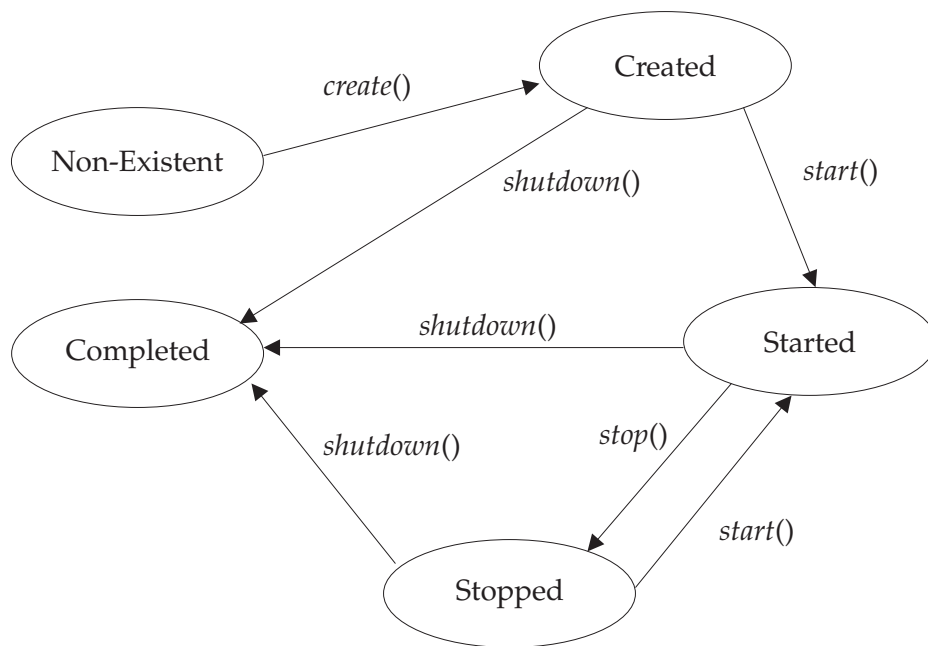


**Figure B-4** Trace System Overview: States of a Trace Stream

Figure B-4 shows the different states an active trace stream passes through. After the `posix_trace_create()` function call, a trace stream becomes CREATED and a trace stream is associated for the future collection of trace events. The status of the trace stream is `POSIX_TRACE_SUSPENDED`. The state becomes STARTED after a call to the `posix_trace_start()` function, and the status becomes `POSIX_TRACE_RUNNING`. In this state, all trace events that are not filtered out will be stored into the trace stream. After a call to `posix_trace_stop()`, the trace stream becomes STOPPED (and the status `POSIX_TRACE_SUSPENDED`). In this state, no new trace events will be recorded in the trace stream, but previously recorded trace events may continue to be read.

After a call to `posix_trace_shutdown()`, the trace stream is in the state COMPLETED. The trace stream no longer exists but, if the Trace Log option is supported, all the information contained in it has been logged. If a log object has not been associated with the trace stream at the creation, it

123876 is the responsibility of the trace controller process to not shut the trace stream down while trace  
123877 events remain to be read in the stream.

## 123878 **Tracing All Processes**

123879 Some implementations have a tracing subsystem with the ability to trace all processes. This is  
123880 useful to debug some types of device drivers such as those for ATM or X25 adapters. These  
123881 types of adapters are used by several independent processes, that are not issued from the same  
123882 process.

123883 The POSIX trace interface does not define any constant or option to create a trace stream tracing  
123884 all processes. POSIX.1 does not prevent this type of implementation and an implementor is free  
123885 to add this capability. Nevertheless, the trace interface allows tracing of all the system trace  
123886 events and all the processes issued from the same process.

123887 If such a tracing system capability has to be implemented, when a trace stream is created, it is  
123888 recommended that a constant named `POSIX_TRACE_ALLPROC` be used instead of the process  
123889 identifier in the argument of the `posix_trace_create()` or `posix_trace_create_withlog()` function. A  
123890 possible value for `POSIX_TRACE_ALLPROC` may be `-1` instead of a real process identifier.

123891 The implementor has to be aware that there is some impact on the tracing behavior as defined in  
123892 the POSIX trace interface. For example:

- 123893 • If the default value for the inheritance attribute is set to  
123894 `POSIX_TRACE_CLOSE_FOR_CHILD`, the implementation has to stop tracing for the child  
123895 process.
- 123896 • The trace controller which is creating this type of trace stream must have the appropriate  
123897 privilege to trace all the processes.

## 123898 **Trace Storage**

123899 The model is based on two types of trace events: system trace events and user-defined trace  
123900 events. The internal representation of trace events is implementation-defined, and so the  
123901 implementor is free to choose the more suitable, practical, and efficient way to design the  
123902 internal management of trace events. For the timestamping operation, the model does not  
123903 impose the `CLOCK_REALTIME` or any other clock. The buffering allocation and operation  
123904 follow the same principle. The implementor is free to use one or more buffers to record trace  
123905 events; the interface assumes only a logical trace stream of sequentially recorded trace events.  
123906 Regarding flushing of trace events, the interface allows the definition of a trace log object which  
123907 typically can be a file. But the group was also aware of defining functions to permit the use of  
123908 this interface in small realtime systems, which may not have general file system capabilities. For  
123909 instance, the three functions `posix_trace_getnext_event()` (blocking),  
123910 `posix_trace_timedgetnext_event()` (blocking with timeout), and `posix_trace_trygetnext_event()` (non-  
123911 blocking) are proposed to read the recorded trace events.

123912 The policy to be used when the trace stream becomes full also relies on common practice:

- 123913 • For an active trace stream, the `POSIX_TRACE_LOOP` trace stream policy permits  
123914 automatic overrun (overwrite of oldest trace events) while waiting for some user-defined  
123915 condition to cause tracing to stop. By contrast, the `POSIX_TRACE_UNTIL_FULL` trace  
123916 stream policy requires the system to stop tracing when the trace stream is full. However, if  
123917 the trace stream that is full is at least partially emptied by a call to the `posix_trace_flush()`  
123918 function or by calls to the `posix_trace_getnext_event()` function, the trace system will  
123919 automatically resume tracing.

123920 If the Trace Log option is supported, the operation of the `POSIX_TRACE_FLUSH` policy is  
123921 an extension of the `POSIX_TRACE_UNTIL_FULL` policy. The automatic free operation (by

123922 flushing to the associated trace log) is added.

- 123923 • If a log is associated with the trace stream and this log is a regular file, these policies also  
123924 apply for the log. One more policy, POSIX\_TRACE\_APPEND, is defined to allow  
123925 indefinite extension of the log. Since the log destination can be any device or pseudo-  
123926 device, the implementation may not be able to manipulate the destination as required by  
123927 POSIX.1-2008. For this reason, the behavior of the log full policy may be unspecified  
123928 depending on the trace log type.

123929 The current trace interface does not define a service to preallocate space for a trace log file,  
123930 because this space can be preallocated by means of a call to the *posix\_fallocate()* function.  
123931 This function could be called after the file has been opened, but before the trace stream is  
123932 created. The *posix\_fallocate()* function ensures that any required storage for regular file data  
123933 is allocated on the file system storage media. If *posix\_fallocate()* returns successfully,  
123934 subsequent writes to the specified file data will not fail due to the lack of free space on the  
123935 file system storage media. Besides trace events, a trace stream also includes trace attributes  
123936 and the mapping from trace event names to trace event type identifiers. The implementor  
123937 is free to choose how to store the trace attributes and the trace event type map, but must  
123938 ensure that this information is not lost when a trace stream overrun occurs.

### 123939 B.2.11.3 Trace Programming Examples

123940 Several programming examples are presented to show the code of the different possible  
123941 subfunctions using a trace subsystem. All these programs need to include the **<trace.h>** header.  
123942 In the examples shown, error checking is omitted for more simplicity.

### 123943 Trace Operation Control

123944 These examples show the creation of a trace stream for another process; one which is already  
123945 trace instrumented. All the default trace stream attributes are used to simplify programming in  
123946 the first example. The second example shows more possibilities.

### 123947 First Example

```
123948 /* Caution. Error checks omitted */
123949 {
123950 trace_attr_t attr;
123951 pid_t pid = traced_process_pid;
123952 int fd;
123953 trace_id_t trid;
123954
123955 - - - - -
123956 /* Initialize trace stream attributes */
123957 posix_trace_attr_init(&attr);
123958 /* Open a trace log */
123959 fd=open("/tmp/mytracelog",...);
123960 /*
123961 * Create a new trace associated with a log
123962 * and with default attributes
123963 */
123964 posix_trace_create_withlog(pid, &attr, fd, &trid);
123965 /* Trace attribute structure can now be destroyed */
123966 posix_trace_attr_destroy(&attr);
123967 /* Start of trace event recording */
```

```

123967 posix_trace_start(trid);
123968 - - - - -
123969 - - - - -
123970 /* Duration of tracing */
123971 - - - - -
123972 - - - - -
123973 /* Stop and shutdown of trace activity */
123974 posix_trace_shutdown(trid);
123975 - - - - -
123976 }

```

## Second Example

Between the initialization of the trace stream attributes and the creation of the trace stream, these trace stream attributes may be modified; see [Trace Stream Attribute Manipulation](#) (on page 3637) for a specific programming example. Between the creation and the start of the trace stream, the event filter may be set; after the trace stream is started, the event filter may be changed. The setting of an event set and the change of a filter is shown in [Create a Trace Event Type Set and Change the Trace Event Type Filter](#) (on page 3637).

```

123984 /* Caution. Error checks omitted */
123985 {
123986 trace_attr_t attr;
123987 pid_t pid = traced_process_pid;
123988 int fd;
123989 trace_id_t trid;
123990 - - - - -
123991 /* Initialize trace stream attributes */
123992 posix_trace_attr_init(&attr);
123993 /* Attr default may be changed at this place; see example */
123994 - - - - -
123995 /* Create and open a trace log with R/W user access */
123996 fd=open("/tmp/mytracelog",O_WRONLY|O_CREAT,S_IRUSR|S_IWUSR);
123997 /* Create a new trace associated with a log */
123998 posix_trace_create_withlog(pid, &attr, fd, &trid);
123999 /*
124000 * If the Trace Filter option is supported
124001 * trace event type filter default may be changed at this place;
124002 * see example about changing the trace event type filter
124003 */
124004 posix_trace_start(trid);
124005 - - - - -
124006
124007 /*
124008 * If you have an uninteresting part of the application
124009 * you can stop temporarily.
124010 *
124011 * posix_trace_stop(trid);
124012 * - - - - -
124013 * - - - - -
124014 * posix_trace_start(trid);
124015 */
124016 - - - - -
124017 }

```

```

124017 * If the Trace Filter option is supported
124018 * the current trace event type filter can be changed
124019 * at any time (see example about how to set
124020 * a trace event type filter)
124021 */
124022 - - - - -
124023 /* Stop the recording of trace events */
124024 posix_trace_stop(trid);
124025 /* Shutdown the trace stream */
124026 posix_trace_shutdown(trid);
124027 /*
124028 * Destroy trace stream attributes; attr structure may have
124029 * been used during tracing to fetch the attributes
124030 */
124031 posix_trace_attr_destroy(&attr);
124032 - - - - -
124033 }

```

### Application Instrumentation

This example shows an instrumented application. The code is included in a block of instructions, perhaps a function from a library. Possibly in an initialization part of the instrumented application, two user trace event names are mapped to two trace event type identifiers (function `posix_trace_eventid_open()`). Then two trace points are programmed.

```

124039 /* Caution. Error checks omitted */
124040 {
124041 trace_event_id_t eventidl, eventid2;
124042 - - - - -
124043 /* Initialization of two trace event type ids */
124044 posix_trace_eventid_open("my_first_event",&eventidl);
124045 posix_trace_eventid_open("my_second_event",&eventid2);
124046 - - - - -
124047 - - - - -
124048 - - - - -
124049 /* Trace point */
124050 posix_trace_event(eventidl,NULL,0);
124051 - - - - -
124052 /* Trace point */
124053 posix_trace_event(eventid2,NULL,0);
124054 - - - - -
124055 }

```

### Trace Analyzer

This example shows the manipulation of a trace log resulting from the dumping of a completed trace stream. All the default attributes are used to simplify programming, and data associated with a trace event is not shown in the first example. The second example shows more possibilities.



**First Example**

```

124061
124062 /* Caution. Error checks omitted */
124063 {
124064 int fd;
124065 trace_id_t trid;
124066 posix_trace_event_info trace_event;
124067 char trace_event_name[TRACE_EVENT_NAME_MAX];
124068 int return_value;
124069 size_t returndatasize;
124070 int lost_event_number;
124071
124072 -- -- -- --
124073
124074 /* Open an existing trace log */
124075 fd=open("/tmp/tracelog", O_RDONLY);
124076 /* Open a trace stream on the open log */
124077 posix_trace_open(fd, &trid);
124078 /* Read a trace event */
124079 posix_trace_getnext_event(trid, &trace_event,
124080 NULL, 0, &returndatasize,&return_value);
124081
124082 /* Read and print all trace event names out in a loop */
124083 while (return_value == NULL)
124084 {
124085 /*
124086 * Get the name of the trace event associated
124087 * with trid trace ID
124088 */
124089 posix_trace_eventid_get_name(trid, trace_event.event_id,
124090 trace_event_name);
124091 /* Print the trace event name out */
124092 printf("%s\n",trace_event_name);
124093 /* Read a trace event */
124094 posix_trace_getnext_event(trid, &trace_event,
124095 NULL, 0, &returndatasize,&return_value);
124096 }
124097
124098 /* Close the trace stream */
124099 posix_trace_close(trid);
124100 /* Close the trace log */
124101 close(fd);
124102 }
124103

```

**Second Example**

The complete example includes the two other examples in [Retrieve Information from a Trace Log](#) (on page 3638) and in [Retrieve the List of Trace Event Types Used in a Trace Log](#) (on page 3639). For example, the *maxdatasize* variable is set in [Retrieve the List of Trace Event Types Used in a Trace Log](#) (on page 3639).

```

124104 /* Caution. Error checks omitted */
124105 {
124106 int fd;
124107 trace_id_t trid;
124108 posix_trace_event_info trace_event;

```

```

124109 char trace_event_name[TRACE_EVENT_NAME_MAX];
124110 char * data;
124111 size_t maxdatasize=1024, returndatasize;
124112 int return_value;
124113 - - - - -
124114 /* Open an existing trace log */
124115 fd=open("/tmp/tracelog", O_RDONLY);
124116 /* Open a trace stream on the open log */
124117 posix_trace_open(fd, &trid);
124118 /*
124119 * Retrieve information about the trace stream which
124120 * was dumped in this trace log (see example)
124121 */
124122 - - - - -
124123 /* Allocate a buffer for trace event data */
124124 data=(char *)malloc(maxdatasize);
124125 /*
124126 * Retrieve the list of trace events used in this
124127 * trace log (see example)
124128 */
124129 - - - - -
124130 /* Read and print all trace event names and data out in a loop */
124131 while (1)
124132 {
124133 posix_trace_getnext_event(trid, &trace_event,
124134 data, maxdatasize, &returndatasize,&return_value);
124135 if (return_value != NULL) break;
124136 /*
124137 * Get the name of the trace event type associated
124138 * with trid trace ID
124139 */
124140 posix_trace_eventid_get_name(trid, trace_event.event_id,
124141 trace_event_name);
124142 {
124143 int i;
124144 /* Print the trace event name out */
124145 printf("%s: ", trace_event_name);
124146 /* Print the trace event data out */
124147 for (i=0; i<returndatasize, i++) printf("%02.2X",
124148 (unsigned char)data[i]);
124149 printf("\n");
124150 }
124151 }
124152 /* Close the trace stream */
124153 posix_trace_close(trid);
124154 /* The buffer data is deallocated */
124155 free(data);
124156 /* Now the file can be closed */
124157 close(fd);
124158 }

```

**Several Programming Manipulations**

The following examples show some typical sets of operations needed in some contexts.

**Trace Stream Attribute Manipulation**

This example shows the manipulation of a trace stream attribute object in order to change the default value provided by a previous *posix\_trace\_attr\_init()* call.

```

/* Caution. Error checks omitted */
{
 trace_attr_t attr;
 size_t logsize=100000;
 - - - - -
 /* Initialize trace stream attributes */
 posix_trace_attr_init(&attr);
 /* Set the trace name in the attributes structure */
 posix_trace_attr_setname(&attr, "my_trace");
 /* Set the trace full policy */
 posix_trace_attr_setstreamfullpolicy(&attr, POSIX_TRACE_LOOP);
 /* Set the trace log size */
 posix_trace_attr_setlogsize(&attr, logsize);
 - - - - -
}

```

**Create a Trace Event Type Set and Change the Trace Event Type Filter**

This example is valid only if the Trace Event Filter option is supported. This example shows the manipulation of a trace event type set in order to change the trace event type filter for an existing active trace stream, which may be just-created, running, or suspended. Some sets of trace event types are well-known, such as the set of trace event types not associated with a process, some trace event types are just-built trace event types for this trace stream; one trace event type is the predefined trace event error type which is deleted from the trace event type set.

```

/* Caution. Error checks omitted */
{
 trace_id_t trid = existing_trace;
 trace_event_set_t set;
 trace_event_id_t trace_event1, trace_event2;
 - - - - -
 /* Initialize to an empty set of trace event types */
 /* (not strictly required because posix_trace_event_set_fill() */
 /* will ignore the prior contents of the event set.) */
 posix_trace_eventset_emptyset(&set);
 /*
 * Fill the set with all system trace events
 * not associated with a process
 */
 posix_trace_eventset_fill(&set, POSIX_TRACE_WOPID_EVENTS);
 /*
 * Get the trace event type identifier of the known trace event name
 * my_first_event for the trid trace stream
 */
 posix_trace_trid_eventid_open(trid, "my_first_event", &trace_event1);
}

```

```

124206 /* Add the set with this trace event type identifier */
124207 posix_trace_eventset_add_event(trace_event1, &set);
124208 /*
124209 * Get the trace event type identifier of the known trace event name
124210 * my_second_event for the trid trace stream
124211 */
124212 posix_trace_trid_eventid_open(trid, "my_second_event", &trace_event2);
124213 /* Add the set with this trace event type identifier */
124214 posix_trace_eventset_add_event(trace_event2, &set);
124215 - - - - -
124216 /* Delete the system trace event POSIX_TRACE_ERROR from the set */
124217 posix_trace_eventset_del_event(POSIX_TRACE_ERROR, &set);
124218 - - - - -
124219 /* Modify the trace stream filter making it equal to the new set */
124220 posix_trace_set_filter(trid, &set, POSIX_TRACE_SET_EVENTSET);
124221 - - - - -
124222 /*
124223 * Now trace_event1, trace_event2, and all system trace event types
124224 * not associated with a process, except for the POSIX_TRACE_ERROR
124225 * system trace event type, are filtered out of (not recorded in) the
124226 * existing trace stream.
124227 */
124228 }

```

### Retrieve Information from a Trace Log

This example shows how to extract information from a trace log, the dump of a trace stream. This code:

- Asks if the trace stream has lost trace events
- Extracts the information about the version of the trace subsystem which generated this trace log
- Retrieves the maximum size of trace event data; this may be used to dynamically allocate an array for extracting trace event data from the trace log without overflow

```

124237 /* Caution. Error checks omitted */
124238 {
124239 struct posix_trace_status_info statusinfo;
124240 trace_attr_t attr;
124241 trace_id_t trid = existing_trace;
124242 size_t maxdatasize;
124243 char genversion[TRACE_NAME_MAX];
124244 - - - - -
124245 /* Get the trace stream status */
124246 posix_trace_get_status(trid, &statusinfo);
124247 /* Detect an overrun condition */
124248 if (statusinfo.posix_stream_overrun_status == POSIX_TRACE_OVERRUN)
124249 printf("trace events have been lost\n");
124250 /* Get attributes from the trid trace stream */
124251 posix_trace_get_attr(trid, &attr);
124252 /* Get the trace generation version from the attributes */

```

```

124253 posix_trace_attr_getgenversion(&attr, genversion);
124254 /* Print the trace generation version out */
124255 printf("Information about Trace Generator:%s\n",genversion);

124256 /* Get the trace event max data size from the attributes */
124257 posix_trace_attr_getmaxdatasize(&attr, &maxdatasize);
124258 /* Print the trace event max data size out */
124259 printf("Maximum size of associated data:%d\n",maxdatasize);
124260 /* Destroy the trace stream attributes */
124261 posix_trace_attr_destroy(&attr);
124262 }

```

### Retrieve the List of Trace Event Types Used in a Trace Log

This example shows the retrieval of a trace stream's trace event type list. This operation may be very useful if you are interested only in tracking the type of trace events in a trace log.

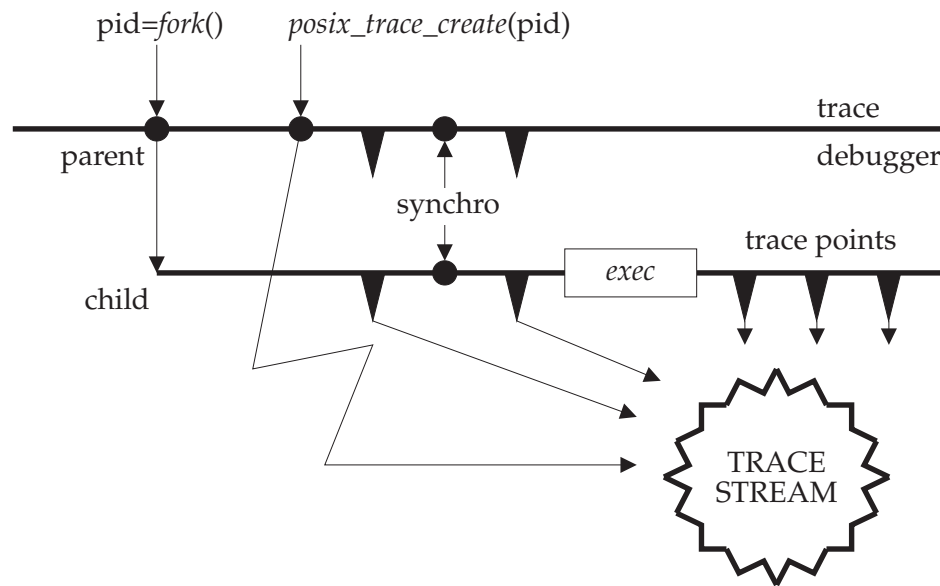
```

124266 /* Caution. Error checks omitted */
124267 {
124268 trace_id_t trid = existing_trace;
124269 trace_event_id_t event_id;
124270 char event_name[TRACE_EVENT_NAME_MAX];
124271 int return_value;
124272 - - - - -

124273 /*
124274 * In a loop print all existing trace event names out
124275 * for the trid trace stream
124276 */
124277 while (1)
124278 {
124279 posix_trace_eventtypelist_getnext_id(trid, &event_id
124280 &return_value);
124281 if (return_value != NULL) break;
124282 /*
124283 * Get the name of the trace event associated
124284 * with trid trace ID
124285 */
124286 posix_trace_eventid_get_name(trid, event_id, event_name);
124287 /* Print the name out */
124288 printf("%s\n", event_name);
124289 }
124290 }

```

## 124291 B.2.11.4 Rationale on Trace for Debugging

124292 **Figure B-5** Trace Another Process

124293 Among the different possibilities offered by the trace interface defined in POSIX.1-2008, the  
 124294 debugging of an application is the most interesting one. Typical operations in the controlling  
 124295 debugger process are to filter trace event types, to get trace events from the trace stream, to stop  
 124296 the trace stream when the debugged process is executing uninteresting code, to start the trace  
 124297 stream when some interesting point is reached, and so on. The interface defined in POSIX.1-2008  
 124298 should define all the necessary base functions to allow this dynamic debug handling.

124299 **Figure B-5** shows an example in which the trace stream is created after the call to the `fork()`  
 124300 function. If the user does not want to lose trace events, some synchronization mechanism  
 124301 (represented in the figure) may be needed before calling the `exec()` function, to give the parent a  
 124302 chance to create the trace stream before the child begins the execution of its trace points.

## 124303 B.2.11.5 Rationale on Trace Event Type Name Space

124304 At first, the working group was in favor of the representation of a trace event type by an integer  
 124305 (`event_name`). It seems that existing practice shows the weakness of such a representation. The  
 124306 collision of trace event types is the main problem that cannot be simply resolved using this sort  
 124307 of representation. Suppose, for example, that a third party designs an instrumented library. The  
 124308 user does not have the source of this library and wants to trace his application which uses in  
 124309 some part the third-party library. There is no means for him to know what are the trace event  
 124310 types used in the instrumented library so he has some chance of duplicating some of them and  
 124311 thus to obtain a contaminated tracing of his application.

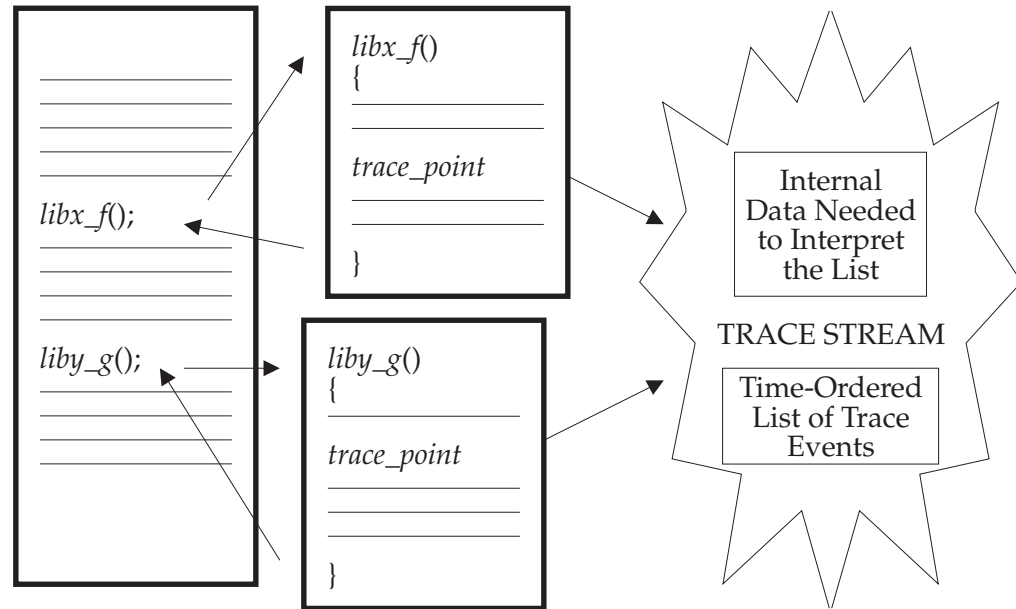


**Figure B-6** Trace Name Space Overview: With Third-Party Library

There are requirements to allow program images containing pieces from various vendors to be traced without also requiring those of any other vendors to coordinate their uses of the trace facility, and especially the naming of their various trace event types and trace point IDs. The chosen solution is to provide a very large name space, large enough so that the individual vendors can give their trace types and tracepoint IDs sufficiently long and descriptive names making the occurrence of collisions quite unlikely. The probability of collision is thus made sufficiently low so that the problem may, as a practical matter, be ignored. By requirement, the consequence of collisions will be a slight ambiguity in the trace streams; tracing will continue in spite of collisions and ambiguities. “The show must go on”. The *posix\_prog\_address* member of the **posix\_trace\_event\_info** structure is used to allow trace streams to be unambiguously interpreted, despite the fact that trace event types and trace event names need not be unique.

The *posix\_trace\_eventid\_open()* function is required to allow the instrumented third-party library to get a valid trace event type identifier for its trace event names. This operation is, somehow, an allocation, and the group was aware of proposing some deallocation mechanism which the instrumented application could use to recover the resources used by a trace event type identifier. This would have given the instrumented application the benefit of being capable of reusing a possible minimum set of trace event type identifiers, but also the inconvenience to have, possibly in the same trace stream, one trace event type identifier identifying two different trace event types. After some discussions the group decided to not define such a function which would make this API thicker for little benefit, the user having always the possibility of adding identification information in the *data* member of the trace event structure.

The set of the trace event type identifiers the controlling process wants to filter out is initialized in the trace mechanism using the function *posix\_trace\_set\_filter()*, setting the arguments according to the definitions explained in *posix\_trace\_set\_filter()*. This operation can be done statically (when the trace is in the STOPPED state) or dynamically (when the trace is in the STARTED state). The preparation of the filter is normally done using the function defined in *posix\_trace\_eventtypelist\_getnext\_id()* and eventually the function *posix\_trace\_eventtypelist\_rewind()* in order to know (before the recording) the list of the potential

set of trace event types that can be recorded. In the case of an active trace stream, this list may not be exhaustive. Actually, the target process may not have yet called the function `posix_trace_eventid_open()`. But it is a common practice, for a controlling process, to prepare the filtering of a future trace stream before its start. Therefore the user must have a way to get the trace event type identifier corresponding to a well-known trace event name before its future association by the pre-cited function. This is done by calling the `posix_trace_trid_eventid_open()` function, given the trace stream identifier and the trace name, and described hereafter. Because this trace event type identifier is associated with a trace stream identifier, where a unique process has initialized two or more traces, the implementation is expected to return the same trace event type identifier for successive calls to `posix_trace_trid_eventid_open()` with different trace stream identifiers. The `posix_trace_eventid_get_name()` function is used by the controller process to identify, by the name, the trace event type returned by a call to the `posix_trace_eventtypelist_getnext_id()` function.

Afterwards, the set of trace event types is constructed using the functions defined in `posix_trace_eventset_empty()`, `posix_trace_eventset_fill()`, `posix_trace_eventset_add()`, and `posix_trace_eventset_del()`.

A set of functions is provided devoted to the manipulation of the trace event type identifier and names for an active trace stream. All these functions require the trace stream identifier argument as the first parameter. The opacity of the trace event type identifier implies that the user cannot associate directly its well-known trace event name with the system-associated trace event type identifier.

The `posix_trace_trid_eventid_open()` function allows the application to get the system trace event type identifier back from the system, given its well-known trace event name. This function is useful only when a controlling process needs to specify specific events to be filtered.

The `posix_trace_eventid_get_name()` function allows the application to obtain a trace event name given its trace event type identifier. One possible use of this function is to identify the type of a trace event retrieved from the trace stream, and print it. The easiest way to implement this requirement, is to use a single trace event type map for all the processes whose maps are required to be identical. A more difficult way is to attempt to keep multiple maps identical at every call to `posix_trace_eventid_open()` and `posix_trace_trid_eventid_open()`.

#### 124371 B.2.11.6 Rationale on Trace Events Type Filtering

The most basic rationale for runtime and pre-registration filtering (selection/rejection) of trace event types is to prevent choking of the trace collection facility, and/or overloading of the computer system. Any worthwhile trace facility can bring even the largest computer to its knees. Otherwise, everything would be recorded and filtered after the fact; it would be much simpler, but impractical.

To achieve debugging, measurement, or whatever the purpose of tracing, the filtering of trace event types is an important part of trace analysis. Due to the fact that the trace events are put into a trace stream and probably logged afterwards into a file, different levels of filtering—that is, rejection of trace event types—are possible.



### Filtering of Trace Event Types Before Tracing

This function, represented by the `posix_trace_set_filter()` function in POSIX.1-2008 (see `posix_trace_set_filter()`), selects, before or during tracing, the set of trace event types to be filtered out. It should be possible also (as OSF suggested in their ETAP trace specifications) to select the kernel trace event types to be traced in a system-wide fashion. These two functionalities are called the pre-filtering of trace event types.

The restriction on the actual type used for the `trace_event_set_t` type is intended to guarantee that these objects can always be assigned, have their address taken, and be passed by value as parameters. It is not intended that this type be a structure including pointers to other data structures, as that could impact the portability of applications performing such operations. A reasonable implementation could be a structure containing an array of integer types.

### Filtering of Trace Event Types at Runtime

It is possible to build this functionality using the `posix_trace_set_filter()` function. A privileged process or a privileged thread can get trace events from the trace stream of another process or thread, and thus specify the type of trace events to record into a file, using implementation-defined methods and interfaces. This functionality, called inline filtering of trace event types, is used for runtime analysis of trace streams.

### Post-Mortem Filtering of Trace Event Types

The word “post-mortem” is used here to indicate that some unanticipated situation occurs during execution that does not permit a pre or inline filtering of trace events and that it is necessary to record all trace event types to have a chance to discover the problem afterwards. When the program stops, all the trace events recorded previously can be analyzed in order to find the solution. This functionality could be named the post-filtering of trace event types.

### Discussions about Trace Event Type Filtering

After long discussions with the parties involved in the process of defining the trace interface, it seems that the sensitivity to the filtering problem is different, but everybody agrees that the level of the overhead introduced during the tracing operation depends on the filtering method elected. If the time that it takes the trace event to be recorded can be neglected, the overhead introduced by the filtering process can be classified as follows:

Pre-filtering      System and process/thread-level overhead

Inline-filtering    Process/thread-level overhead

Post-filtering      No overhead; done offline

The pre-filtering could be named “critical realtime” filtering in the sense that the filtering of trace event type is manageable at the user level so the user can lower to a minimum the filtering overhead at some user selected level of priority for the inline filtering, or delay the filtering to after execution for the post-filtering. The counterpart of this solution is that the size of the trace stream must be sufficient to record all the trace events. The advantage of the pre-filtering is that the utilization of the trace stream is optimized.

Only pre-filtering is defined by POSIX.1-2008. However, great care must be taken in specifying pre-filtering, so that it does not impose unacceptable overhead. Moreover, it is necessary to isolate all the functionality relative to the pre-filtering.

The result of this rationale is to define a new option, the Trace Event Filter option, not necessarily implemented in small realtime systems, where system overhead is minimized to the extent possible.

## 124425 B.2.11.7 Tracing, pthread API

124426 The objective to be able to control tracing for individual threads may be in conflict with the  
 124427 efficiency expected in threads with a *contentionscope* attribute of PTHREAD\_SCOPE\_PROCESS.  
 124428 For these threads, context switches from one thread that has tracing enabled to another thread  
 124429 that has tracing disabled may require a kernel call to inform the kernel whether it has to trace  
 124430 system events executed by that thread or not. For this reason, it was proposed that the ability to  
 124431 enable or disable tracing for PTHREAD\_SCOPE\_PROCESS threads be made optional, through  
 124432 the introduction of a Trace Scope Process option. A trace implementation which did not  
 124433 implement the Trace Scope Process option would not honor the tracing-state attribute of a thread  
 124434 with PTHREAD\_SCOPE\_PROCESS; it would, however, honor the tracing-state attribute of a  
 124435 thread with PTHREAD\_SCOPE\_SYSTEM. This proposal was rejected as:

- 124436 1. Removing desired functionality (per-thread trace control)
- 124437 2. Introducing counter-intuitive behavior for the tracing-state attribute
- 124438 3. Mixing logically orthogonal ideas (thread scheduling and thread tracing)
- 124439 [Objective 4]

124440 Finally, to solve this complex issue, this API does not provide *pthread\_gettracingstate()*,  
 124441 *pthread\_settracingstate()*, *pthread\_attr\_gettracingstate()*, and *pthread\_attr\_settracingstate()*  
 124442 interfaces. These interfaces force the thread implementation to add to the weight of the thread  
 124443 and cause a revision of the threads libraries, just to support tracing. Worse yet,  
 124444 *posix\_trace\_event()* must always test this per-thread variable even in the common case where it is  
 124445 not used at all. Per-thread tracing is easy to implement using existing interfaces where  
 124446 necessary; see the following example.

124447 **Example**

```

124448 /* Caution. Error checks omitted */
124449 static pthread_key_t my_key;
124450 static trace_event_id_t my_event_id;
124451 static pthread_once_t my_once = PTHREAD_ONCE_INIT;

124452 void my_init(void)
124453 {
124454 (void) pthread_key_create(&my_key, NULL);
124455 (void) posix_trace_eventid_open("my", &my_event_id);
124456 }

124457 int get_trace_flag(void)
124458 {
124459 pthread_once(&my_once, my_init);
124460 return (pthread_getspecific(my_key) != NULL);
124461 }

124462 void set_trace_flag(int f)
124463 {
124464 pthread_once(&my_once, my_init);
124465 pthread_setspecific(my_key, f? &my_event_id: NULL);
124466 }

124467 fn()
124468 {
124469 if (get_trace_flag())
124470 posix_trace_event(my_event_id, ...)
124471 }
```

- 124472 The above example does not implement third-party state setting.
- 124473 Lastly, per-thread tracing works poorly for threads with PTHREAD\_SCOPE\_PROCESS  
124474 contention scope. These “library” threads have minimal interaction with the kernel and would  
124475 have to explicitly set the attributes whenever they are context switched to a new kernel thread in  
124476 order to trace system events. Such state was explicitly avoided in POSIX threads to keep  
124477 PTHREAD\_SCOPE\_PROCESS threads lightweight.
- 124478 The reason that keeping PTHREAD\_SCOPE\_PROCESS threads lightweight is important is that  
124479 such threads can be used not just for simple multi-processors but also for co-routine style  
124480 programming (such as discrete event simulation) without inventing a new threads paradigm.  
124481 Adding extra runtime cost to thread context switches will make using POSIX threads less  
124482 attractive in these situations.
- 124483 *B.2.11.8 Rationale on Triggering*
- 124484 The ability to start or stop tracing based on the occurrence of specific trace event types has been  
124485 proposed as a parallel to similar functionality appearing in logic analyzers. Such triggering, in  
124486 order to be very useful, should be based not only on the trace event type, but on trace event-  
124487 specific data, including tests of user-specified fields for matching or threshold values.
- 124488 Such a facility is unnecessary where the buffering of the stream is not a constraint, since such  
124489 checks can be performed offline during post-mortem analysis.
- 124490 For example, a large system could incorporate a daemon utility to collect the trace records from  
124491 memory buffers and spool them to secondary storage for later analysis. In the instances where  
124492 resources are truly limited, such as embedded applications, the application incorporation of  
124493 application code to test the circumstances of a trace event and call the trace point only if needed  
124494 is usually straightforward.
- 124495 For performance reasons, the *posix\_trace\_event()* function should be implemented using a macro,  
124496 so if the trace is inactive, the trace event point calls are latent code and must cost no more than a  
124497 scalar test.
- 124498 The API proposed in POSIX.1-2008 does not include any triggering functionality.
- 124499 *B.2.11.9 Rationale on Timestamp Clock*
- 124500 It has been suggested that the tracing mechanism should include the possibility of specifying the  
124501 clock to be used in timestamping the trace events. When application trace events must be  
124502 correlated to remote trace events, such a facility could provide a global time reference not  
124503 available from a local clock. Further, the application may be driven by timers based on a clock  
124504 different from that used for the timestamp, and the correlation of the trace to those untraced  
124505 timer activities could be an important part of the analysis of the application.
- 124506 However, the tracing mechanism needs to be fast and just the provision of such an option can  
124507 materially affect its performance. Leaving aside the performance costs of reading some clocks,  
124508 this notion is also ill-defined when kernel trace events are to be traced by two applications  
124509 making use of different tracing clocks. This can even happen within a single application where  
124510 different parts of the application are served by different clocks. Another complication can occur  
124511 when a clock is maintained strictly at the user level and is unavailable at the kernel level.
- 124512 It is felt that the benefits of a selectable trace clock do not match its costs. Applications that wish  
124513 to correlate clocks other than the default tracing clock can include trace events with sample  
124514 values of those other clocks, allowing correlation of timestamps from the various independent  
124515 clocks. In any case, such a technique would be required when applications are sensitive to

124516 multiple clocks.

#### 124517 B.2.11.10 Rationale on Different Overrun Conditions

124518 The analysis of the dynamic behavior of the trace mechanism shows that different overrun  
124519 conditions may occur. The API must provide a means to manage such conditions in a portable  
124520 way.

#### 124521 **Overrun in Trace Streams Initialized with POSIX\_TRACE\_LOOP Policy**

124522 In this case, the user of the trace mechanism is interested in using the trace stream with  
124523 POSIX\_TRACE\_LOOP policy to record trace events continuously, but ideally without losing any  
124524 trace events. The online analyzer process must get the trace events at a mean speed equivalent to  
124525 the recording speed. Should the trace stream become full, a trace stream overrun occurs. This  
124526 condition is detected by getting the status of the active trace stream (function  
124527 *posix\_trace\_get\_status()*) and looking at the member *posix\_stream\_overrun\_status* of the read  
124528 **posix\_stream\_status** structure. In addition, two predefined trace event types are defined:

- 124529 1. The beginning of a trace overflow, to locate the beginning of an overflow when reading a  
124530 trace stream
- 124531 2. The end of a trace overflow, to locate the end of an overflow, when reading a trace stream

124532 As a timestamp is associated with these predefined trace events, it is possible to know the  
124533 duration of the overflow.

#### 124534 **Overrun in Dumping Trace Streams into Trace Logs**

124535 The user lets the trace mechanism dump the trace stream initialized with  
124536 POSIX\_TRACE\_FLUSH policy automatically into a trace log. If the dump operation is slower  
124537 than the recording of trace events, the trace stream can overrun. This condition is detected by  
124538 getting the status of the active trace stream (the *posix\_trace\_get\_status()* function) and looking at  
124539 the member *posix\_stream\_overrun\_status* of the read **posix\_stream\_status** structure. This overrun  
124540 indicates that the trace mechanism is not able to operate in this mode at this speed. It is the  
124541 responsibility of the user to modify one of the trace parameters (the stream size or the trace  
124542 event type filter, for instance) to avoid such overrun conditions, if overruns are to be prevented.  
124543 The same already predefined trace event types (see [Overrun in Trace Streams Initialized with  
124544 POSIX\\_TRACE\\_LOOP Policy](#)) are used to detect and to know the duration of an overflow.

#### 124545 **Reading an Active Trace Stream**

124546 Although this trace API allows one to read an active trace stream with log while it is tracing, this  
124547 feature can lead to false overflow origin interpretation: the trace log or the reader of the trace  
124548 stream. Reading from an active trace stream with log is thus non-portable, and has been left  
124549 unspecified.

124550 **B.2.12 Data Types**124551 *B.2.12.1 Defined Types*

124552 The requirement that additional types defined in this section end in “\_t” was prompted by the  
 124553 problem of name space pollution. It is difficult to define a type (where that type is not one  
 124554 defined by POSIX.1-2008) in one header file and use it in another without adding symbols to the  
 124555 name space of the program. To allow implementors to provide their own types, all conforming  
 124556 applications are required to avoid symbols ending in “\_t”, which permits the implementor to  
 124557 provide additional types. Because a major use of types is in the definition of structure members,  
 124558 which can (and in many cases must) be added to the structures defined in POSIX.1-2008, the  
 124559 need for additional types is compelling.

124560 The types, such as **ushort** and **ulong**, which are in common usage, are not defined in  
 124561 POSIX.1-2008 (although **ushort\_t** would be permitted as an extension). They can be added to  
 124562 **<sys/types.h>** using a feature test macro (see [Section B.2.2.1](#), on page 3526). A suggested symbol  
 124563 for these is **\_SYSIII**. Similarly, the types like **u\_short** would probably be best controlled by **\_BSD**.

124564 Some of these symbols may appear in other headers; see [Section B.2.2.2](#) (on page 3527).

124565 **dev\_t** This type may be made large enough to accommodate host-locality considerations  
 124566 of networked systems.

124567 This type must be arithmetic. Earlier proposals allowed this to be non-arithmetic  
 124568 (such as a structure) and provided a *samefile()* function for comparison.

124569 **gid\_t** Some implementations had separated **gid\_t** from **uid\_t** before POSIX.1 was  
 124570 completed. It would be difficult for them to coalesce them when it was  
 124571 unnecessary. Additionally, it is quite possible that user IDs might be different from  
 124572 group IDs because the user ID might wish to span a heterogeneous network,  
 124573 where the group ID might not.

124574 For current implementations, the cost of having a separate **gid\_t** will be only  
 124575 lexical.

124576 **mode\_t** This type was chosen so that implementations could choose the appropriate  
 124577 integer type, and for compatibility with the ISO C standard. 4.3 BSD uses  
 124578 **unsigned short** and the SVID uses **ushort**, which is the same. Historically, only the  
 124579 low-order sixteen bits are significant.

124580 **nlink\_t** This type was introduced in place of **short** for *st\_nlink* (see the **<sys/stat.h>** header)  
 124581 in response to an objection that **short** was too small.

124582 **off\_t** This type is used to represent a file offset or file size. On systems supporting large  
 124583 files, **off\_t** is larger than 32 bits in at least one programming environment. Other  
 124584 programming environments may use different sizes for **off\_t**, for compatibility or  
 124585 other reasons.

124586 **pid\_t** The inclusion of this symbol was controversial because it is tied to the issue of the  
 124587 representation of a process ID as a number. From the point of view of a  
 124588 conforming application, process IDs should be “magic cookies”<sup>8</sup> that are produced  
 124589 by calls such as *fork()*, used by calls such as *waitpid()* or *kill()*, and not otherwise

---

124590 8. An historical term meaning: “An opaque object, or token, of determinate size, whose significance is known only to the entity which  
 124591 created it. An entity receiving such a token from the generating entity may only make such use of the ‘cookie’ as is defined and permitted  
 124592 by the supplying entity.”

|        |                |                                                                                                                                       |
|--------|----------------|---------------------------------------------------------------------------------------------------------------------------------------|
| 124593 |                | analyzed (except that the sign is used as a flag for certain operations).                                                             |
| 124594 |                | The concept of a {PID_MAX} value interacted with this in early proposals. Treating                                                    |
| 124595 |                | process IDs as an opaque type both removes the requirement for {PID_MAX} and                                                          |
| 124596 |                | allows systems to be more flexible in providing process IDs that span a large range                                                   |
| 124597 |                | of values, or a small one.                                                                                                            |
| 124598 |                | Since the values in <b>uid_t</b> , <b>gid_t</b> , and <b>pid_t</b> will be numbers generally, and                                     |
| 124599 |                | potentially both large in magnitude and sparse, applications that are based on                                                        |
| 124600 |                | arrays of objects of this type are unlikely to be fully portable in any case. Solutions                                               |
| 124601 |                | that treat them as magic cookies will be portable.                                                                                    |
| 124602 |                | {CHILD_MAX} precludes the possibility of a “toy implementation”, where there                                                          |
| 124603 |                | would only be one process.                                                                                                            |
| 124604 | <b>ssize_t</b> | This is intended to be a signed analog of <b>size_t</b> . The wording is such that an                                                 |
| 124605 |                | implementation may either choose to use a longer type or simply to use the signed                                                     |
| 124606 |                | version of the type that underlies <b>size_t</b> . All functions that return <b>ssize_t</b> ( <i>read()</i>                           |
| 124607 |                | and <i>write()</i> ) describe as “implementation-defined” the result of an input exceeding                                            |
| 124608 |                | {SSIZE_MAX}. It is recognized that some implementations might have <b>ints</b> that                                                   |
| 124609 |                | are smaller than <b>size_t</b> . A conforming application would be constrained not to                                                 |
| 124610 |                | perform I/O in pieces larger than {SSIZE_MAX}, but a conforming application                                                           |
| 124611 |                | using extensions would be able to use the full range if the implementation                                                            |
| 124612 |                | provided an extended range, while still having a single type-compatible interface.                                                    |
| 124613 |                | The symbols <b>size_t</b> and <b>ssize_t</b> are also required in <b>&lt;unistd.h&gt;</b> to minimize the                             |
| 124614 |                | changes needed for calls to <i>read()</i> and <i>write()</i> . Implementors are reminded that it                                      |
| 124615 |                | must be possible to include both <b>&lt;sys/types.h&gt;</b> and <b>&lt;unistd.h&gt;</b> in the same                                   |
| 124616 |                | program (in either order) without error.                                                                                              |
| 124617 | <b>uid_t</b>   | Before the addition of this type, the data types used to represent these values                                                       |
| 124618 |                | varied throughout early proposals. The <b>&lt;sys/stat.h&gt;</b> header defined these values                                          |
| 124619 |                | as type <b>short</b> , the <b>&lt;passwd.h&gt;</b> file (now <b>&lt;pwd.h&gt;</b> and <b>&lt;grp.h&gt;</b> ) used an <b>int</b> , and |
| 124620 |                | <i>getuid()</i> returned an <b>int</b> . In response to a strong objection to the inconsistent                                        |
| 124621 |                | definitions, all the types were switched to <b>uid_t</b> .                                                                            |
| 124622 |                | In practice, those historical implementations that use varying types of this sort can                                                 |
| 124623 |                | typedef <b>uid_t</b> to <b>short</b> with no serious consequences.                                                                    |
| 124624 |                | The problem associated with this change concerns object compatibility after                                                           |
| 124625 |                | structure size changes. Since most implementations will define <b>uid_t</b> as a short, the                                           |
| 124626 |                | only substantive change will be a reduction in the size of the <b>passwd</b> structure.                                               |
| 124627 |                | Consequently, implementations with an overriding concern for object                                                                   |
| 124628 |                | compatibility can pad the structure back to its current size. For that reason, this                                                   |
| 124629 |                | problem was not considered critical enough to warrant the addition of a separate                                                      |
| 124630 |                | type to POSIX.1.                                                                                                                      |
| 124631 |                | The types <b>uid_t</b> and <b>gid_t</b> are magic cookies. There is no {UID_MAX} defined by                                           |
| 124632 |                | POSIX.1, and no structure imposed on <b>uid_t</b> and <b>gid_t</b> other than that they be                                            |
| 124633 |                | positive arithmetic types. (In fact, they could be <b>unsigned char</b> .) There is no                                                |
| 124634 |                | maximum or minimum specified for the number of distinct user or group IDs.                                                            |

124635 B.2.12.2 *The char Type*

124636 POSIX.1-2008 explicitly requires that a **char** type is exactly one byte (8 bits).

124637 B.2.12.3 *Pointer Types*

124638 POSIX.1-2008 explicitly requires implementations to convert pointers to **void \*** and back with no  
124639 loss of information. This is an extension over the ISO C standard.

## 124640 B.3 System Interfaces

124641 See the RATIONALE sections on the individual reference pages.

### 124642 B.3.1 System Interfaces Removed in this Version

124643 The following section contains a list of the interfaces removed in POSIX.1-2008, together with  
124644 advice for application developers on the alternative interfaces that should be used for maximum  
124645 portability.

124646 B.3.1.1 *bcmp()*

124647 Applications are recommended to use the *memcmp()* function instead of this function.

124648 For maximum portability, it is recommended to replace the function call to *bcmp()* as follows:

124649 `#define bcmp(b1,b2,len) memcmp((b1), (b2), (size_t)(len))`

124650 B.3.1.2 *bcopy()*

124651 Applications are recommended to use the *memmove()* function instead of this function.

124652 The following are approximately equivalent (note the order of the arguments):

124653 `bcopy(s1,s2,n) ≈ memmove(s2,s1,n)`

124654 For maximum portability, it is recommended to replace the function call to *bcopy()* as follows:

124655 `#define bcopy(b1,b2,len) (void)memmove((b2), (b1), (len))`

124656 B.3.1.3 *bsd\_signal()*

124657 Applications are recommended to use the *sigaction()* function instead of this function.

124658 The *bsd\_signal()* function was supplied as a migration path for the BSD *signal()* function for  
124659 simple applications that installed a single-argument signal handler function.

124660 Historically, the *bsd\_signal()* function differs from *signal()* in that the SA\_RESTART flag is set  
124661 and the SA\_RESETHAND flag is clear when *bsd\_signal()* is used. The state of these flags is not  
124662 specified for *signal()*.

- 124663 B.3.1.4 *bzero()*
- 124664 Applications are recommended to use the *memset()* function instead of this function.
- 124665 For maximum portability, it is recommended to replace the function call to *bzero()* as follows:
- 124666 `#define bzero(b, len) (void) (memset ((b), '\0', (len)))`
- 
- 124667 B.3.1.5 *ecvt()*, *fcvt()*, *gcvt()*
- 124668 Applications are recommended to use the *sprintf()* function instead of these functions.
- 124669 The *sprintf()* function is required by ISO C and is thus more portable.
- 
- 124670 B.3.1.6 *ftime()*
- 124671 Applications are recommended to use the *time()* function to determine the current time.
- 124672 Realtime applications should use *clock\_gettime()* to determine the current time.
- 
- 124673 B.3.1.7 *getcontext()*, *makecontext()*, *swapcontext()*
- 124674 Due to portability issues with these functions, especially with the manipulation of contexts,
- 124675 applications are recommended to be rewritten to use POSIX threads.
- 
- 124676 B.3.1.8 *gethostbyaddr()*, *gethostbyname()*
- 124677 Applications are recommended to use the *getaddrinfo()* and *getnameinfo()* functions instead of
- 124678 these functions.
- 124679 The *gethostbyaddr()* and *gethostbyname()* functions may return pointers to static data, which may
- 124680 be overwritten by subsequent calls to any of these functions. The suggested replacements do not
- 124681 have this problem and are also IPv6-capable.
- 
- 124682 B.3.1.9 *getwd()*
- 124683 Applications are recommended to use the *getcwd()* function to determine the current working
- 124684 directory.
- 
- 124685 B.3.1.10 *h\_errno*
- 124686 Applications are recommended not to use this error return code. Previously it was set by the
- 124687 *gethostbyname()* and *gethostbyaddr()* functions.
- 
- 124688 B.3.1.11 *index()*
- 124689 Applications are recommended to use the *strchr()* function instead of this function.
- 124690 For maximum portability, it is recommended to replace the function call to *index()* as follows:
- 124691 `#define index(a, b) strchr ((a), (b))`



124692 B.3.1.12 *makecontext()*

124693 Applications using the *getcontext()*, *makecontext()*, and *swapcontext()* functions should be  
124694 rewritten to use POSIX threads.

124695 B.3.1.13 *mktemp()*

124696 Applications are recommended to use the *mkstemp()* function instead of this function.

124697 The *mktemp()* function makes an application vulnerable to possible security problems since  
124698 between the time a pathname is created and the file opened, it is possible for some other process  
124699 to create a file with the same name. The *mkstemp()* function does not have this vulnerability.

124700 B.3.1.14 *pthread\_attr\_getstackaddr()*, *pthread\_attr\_setstackaddr()*

124701 Applications are recommended to use the *pthread\_attr\_setstack()* and *pthread\_attr\_getstack()*  
124702 functions instead of these functions.

124703 There are a number of ambiguities in the specification of the *stackaddr* attribute that makes  
124704 portable use of these interfaces impossible.

124705 B.3.1.15 *rindex()*

124706 Applications are recommended to use the *strrchr()* function instead of this function.

124707 For maximum portability, it is recommended to replace the function call to *rindex()* as follows:

124708 `#define rindex(a,b) strrchr((a),(b))`

124709 B.3.1.16 *scalb()*

124710 Applications are recommended to use either *scalbln()*, *scalblnf()*, or *scalblnl()* instead of these  
124711 functions.

124712 The behavior for the *scalb()* function was only defined when the *n* argument is an integer, a  
124713 NaN, or Inf. The behavior of other values for the *n* argument was unspecified.

124714 B.3.1.17 *ualarm()*

124715 Applications are recommended to use *timer\_create()*, *timer\_delete()*, *timer\_getoverrun()*,  
124716 *timer\_gettime()*, or *timer\_settime()* instead of this function.

124717 B.3.1.18 *usleep()*

124718 Applications are recommended to use the *nanosleep()* function instead of this function.

124719 B.3.1.19 *vfork()*

124720 Applications are recommended to use the *fork()* function instead of this function.

124721 The *vfork()* function was previously under-specified.

124722 B.3.1.20 *wcswcs()*

124723 Applications are recommended to use the *wcsstr()* function instead of this function.

124724 The *wcsstr()* function is technically equivalent and is portable across all ISO C implementations.

### 124725 B.3.2 System Interfaces Removed in the Previous Version

124726 The following system interfaces, headers, and external variables were removed in the previous  
124727 version of this standard:

|        |                  |                        |                   |                          |               |
|--------|------------------|------------------------|-------------------|--------------------------|---------------|
| 124728 | <i>advance()</i> | <i>getdtablesize()</i> | <i>re_exec()</i>  | <i>ttyslot()</i>         | <i>loc1</i>   |
| 124729 | <i>brk()</i>     | <i>getpagesize()</i>   | <i>regcmp()</i>   | <i>valloc()</i>          | <i>__loc1</i> |
| 124730 | <i>chroot()</i>  | <i>getpass()</i>       | <i>regex()</i>    | <i>wait3()</i>           | <i>loc2</i>   |
| 124731 | <i>compile()</i> | <i>getw()</i>          | <i>sbrk()</i>     | <b>&lt;re_comp.h&gt;</b> | <i>locs</i>   |
| 124732 | <i>cuserid()</i> | <i>putw()</i>          | <i>sigstack()</i> | <b>&lt;regexp.h&gt;</b>  |               |
| 124733 | <i>gamma()</i>   | <i>re_comp()</i>       | <i>step()</i>     | <b>&lt;varargs.h&gt;</b> |               |

### 124734 B.3.3 Examples for Spawn

124735 The following long examples are provided in the Rationale (Informative) volume of  
124736 POSIX.1-2008 as a supplement to the reference page for *posix\_spawn()*.

#### 124737 Example Library Implementation of Spawn

124738 The *posix\_spawn()* or *posix\_spawnnp()* functions provide the following:

- 124739 • Simply start a process executing a process image. This is the simplest application for  
124740 process creation, and it may cover most executions of *fork()*.
- 124741 • Support I/O redirection, including pipes.
- 124742 • Run the child under a user and group ID in the domain of the parent.
- 124743 • Run the child at any priority in the domain of the parent.

124744 The *posix\_spawn()* or *posix\_spawnnp()* functions do not cover every possible use of the *fork()*  
124745 function, but they do span the common applications: typical use by a shell and a login utility.

124746 The price for an application is that before it calls *posix\_spawn()* or *posix\_spawnnp()*, the parent  
124747 must adjust to a state that *posix\_spawn()* or *posix\_spawnnp()* can map to the desired state for the  
124748 child. Environment changes require the parent to save some of its state and restore it afterwards.  
124749 The effective behavior of a successful invocation of *posix\_spawn()* is as if the operation were  
124750 implemented with POSIX operations as follows:

```

124751 #include <sys/types.h>
124752 #include <stdlib.h>
124753 #include <stdio.h>
124754 #include <unistd.h>
124755 #include <sched.h>
124756 #include <fcntl.h>
124757 #include <signal.h>
124758 #include <errno.h>
124759 #include <string.h>
124760 #include <signal.h>
124761 /* #include <spawn.h> */

```

```

124762 /*****
124763 /* Things that could be defined in spawn.h */
124764 *****/
124765 typedef struct
124766 {
124767 short posix_attr_flags;
124768 #define POSIX_SPAWN_SETPGROUP 0x1
124769 #define POSIX_SPAWN_SETSIGMASK 0x2
124770 #define POSIX_SPAWN_SETSIGDEF 0x4
124771 #define POSIX_SPAWN_SETSCHEDULER 0x8
124772 #define POSIX_SPAWN_SETSCHEDPARAM 0x10
124773 #define POSIX_SPAWN_RESETHIDS 0x20
124774 pid_t posix_attr_pgroup;
124775 sigset_t posix_attr_sigmask;
124776 sigset_t posix_attr_sigdefault;
124777 int posix_attr_schedpolicy;
124778 struct sched_param posix_attr_schedparam;
124779 } posix_spawnattr_t;
124780
124781 typedef char *posix_spawn_file_actions_t;
124782
124783 int posix_spawn_file_actions_init(
124784 posix_spawn_file_actions_t *file_actions);
124785 int posix_spawn_file_actions_destroy(
124786 posix_spawn_file_actions_t *file_actions);
124787 int posix_spawn_file_actions_addclose(
124788 posix_spawn_file_actions_t *file_actions, int fildes);
124789 int posix_spawn_file_actions_adddup2(
124790 posix_spawn_file_actions_t *file_actions, int fildes,
124791 int newfildes);
124792 int posix_spawn_file_actions_addopen(
124793 posix_spawn_file_actions_t *file_actions, int fildes,
124794 const char *path, int oflag, mode_t mode);
124795 int posix_spawnattr_init(posix_spawnattr_t *attr);
124796 int posix_spawnattr_destroy(posix_spawnattr_t *attr);
124797 int posix_spawnattr_getflags(const posix_spawnattr_t *attr,
124798 short *lags);
124799 int posix_spawnattr_setflags(posix_spawnattr_t *attr, short flags);
124800 int posix_spawnattr_getpgroup(const posix_spawnattr_t *attr,
124801 pid_t *pgroup);
124802 int posix_spawnattr_setpgroup(posix_spawnattr_t *attr, pid_t pgroup);
124803 int posix_spawnattr_getschedpolicy(const posix_spawnattr_t *attr,
124804 int *schedpolicy);
124805 int posix_spawnattr_setschedpolicy(posix_spawnattr_t *attr,
124806 int schedpolicy);
124807 int posix_spawnattr_getschedparam(const posix_spawnattr_t *attr,
124808 struct sched_param *schedparam);
124809 int posix_spawnattr_setschedparam(posix_spawnattr_t *attr,
124810 const struct sched_param *schedparam);
124811 int posix_spawnattr_getsigmask(const posix_spawnattr_t *attr,
124812 sigset_t *sigmask);
124813 int posix_spawnattr_setsigmask(posix_spawnattr_t *attr,
124814 const sigset_t *sigmask);
124815 int posix_spawnattr_getdefault(const posix_spawnattr_t *attr,

```

```

124814 sigset_t *sigdefault);
124815 int posix_spawnattr_setsigdefault(posix_spawnattr_t *attr,
124816 const sigset_t *sigdefault);
124817 int posix_spawn(pid_t *pid, const char *path,
124818 const posix_spawn_file_actions_t *file_actions,
124819 const posix_spawnattr_t *attrp, char *const argv[],
124820 char *const envp[]);
124821 int posix_spawnp(pid_t *pid, const char *file,
124822 const posix_spawn_file_actions_t *file_actions,
124823 const posix_spawnattr_t *attrp, char *const argv[],
124824 char *const envp[]);

124825 /*****
124826 /* Example posix_spawn() library routine */
124827 /*****
124828 int posix_spawn(pid_t *pid,
124829 const char *path,
124830 const posix_spawn_file_actions_t *file_actions,
124831 const posix_spawnattr_t *attrp,
124832 char *const argv[],
124833 char *const envp[])
124834 {
124835 /* Create process */
124836 if ((*pid = fork()) == (pid_t) 0)
124837 {
124838 /* This is the child process */
124839 /* Worry about process group */
124840 if (attrp->posix_attr_flags & POSIX_SPAWN_SETPGROUP)
124841 {
124842 /* Override inherited process group */
124843 if (setpgid(0, attrp->posix_attr_pgroup) != 0)
124844 {
124845 /* Failed */
124846 exit(127);
124847 }
124848 }

124849 /* Worry about thread signal mask */
124850 if (attrp->posix_attr_flags & POSIX_SPAWN_SETSIGMASK)
124851 {
124852 /* Set the signal mask (can't fail) */
124853 sigprocmask(SIG_SETMASK, &attrp->posix_attr_sigmask, NULL);
124854 }

124855 /* Worry about resetting effective user and group IDs */
124856 if (attrp->posix_attr_flags & POSIX_SPAWN_RESETPIDS)
124857 {
124858 /* None of these can fail for this case. */
124859 setuid(getuid());
124860 setgid(getgid());
124861 }

124862 /* Worry about defaulted signals */
124863 if (attrp->posix_attr_flags & POSIX_SPAWN_SETSIGDEF)
124864 {

```

```

124865 struct sigaction deflt;
124866 sigset_t all_signals;

124867 int s;

124868 /* Construct default signal action */
124869 deflt.sa_handler = SIG_DFL;
124870 deflt.sa_flags = 0;

124871 /* Construct the set of all signals */
124872 sigfillset(&all_signals);

124873 /* Loop for all signals */
124874 for (s = 0; sigismember(&all_signals, s); s++)
124875 {
124876 /* Signal to be defaulted? */
124877 if (sigismember(&attrp->posix_attr_sigdefault, s))
124878 {
124879 /* Yes; default this signal */
124880 if (sigaction(s, &deflt, NULL) == -1)
124881 {
124882 /* Failed */
124883 exit(127);
124884 }
124885 }
124886 }
124887 }

124888 /* Worry about the fds if they are to be mapped */
124889 if (file_actions != NULL)
124890 {
124891 /* Loop for all actions in object file_actions */
124892 /* (implementation dives beneath abstraction) */
124893 char *p = *file_actions;

124894 while (*p != '\0')
124895 {
124896 if (strncmp(p, "close(", 6) == 0)
124897 {
124898 int fd;

124899 if (sscanf(p + 6, "%d", &fd) != 1)
124900 {
124901 exit(127);
124902 }
124903 if (close(fd) == -1)
124904 exit(127);
124905 }
124906 else if (strncmp(p, "dup2(", 5) == 0)
124907 {
124908 int fd, newfd;

124909 if (sscanf(p + 5, "%d,%d", &fd, &newfd) != 2)
124910 {
124911 exit(127);
124912 }
124913 if (dup2(fd, newfd) == -1)

```

```

124914 exit(127);
124915 }
124916 else if (strncmp(p, "open(", 5) == 0)
124917 {
124918 int fd, oflag;
124919 mode_t mode;
124920 int tempfd;
124921 char path[1000]; /* Should be dynamic */
124922 char *q;
124923
124924 if (sscanf(p + 5, "%d,", &fd) != 1)
124925 {
124926 exit(127);
124927 }
124928 p = strchr(p, ',') + 1;
124929 q = strchr(p, '*');
124930 if (q == NULL)
124931 exit(127);
124932 strncpy(path, p, q - p);
124933 path[q - p] = '\0';
124934 if (sscanf(q + 1, "%o,%o", &oflag, &mode) != 2)
124935 {
124936 exit(127);
124937 }
124938 if (close(fd) == -1)
124939 {
124940 if (errno != EBADF)
124941 exit(127);
124942 }
124943 tempfd = open(path, oflag, mode);
124944 if (tempfd == -1)
124945 exit(127);
124946 if (tempfd != fd)
124947 {
124948 if (dup2(tempfd, fd) == -1)
124949 {
124950 exit(127);
124951 }
124952 if (close(tempfd) == -1)
124953 {
124954 exit(127);
124955 }
124956 }
124957 }
124958 else
124959 {
124960 exit(127);
124961 }
124962 p = strchr(p, ')') + 1;
124963 }
124964
124965 /* Worry about setting new scheduling policy and parameters */
124966 if (attrp->posix_attr_flags & POSIX_SPAWN_SETSCHEDULER)

```

```

124966 {
124967 if (sched_setscheduler(0, attrp->posix_attr_schedpolicy,
124968 &attrp->posix_attr_schedparam) == -1)
124969 {
124970 exit(127);
124971 }
124972 }

124973 /* Worry about setting only new scheduling parameters */
124974 if (attrp->posix_attr_flags & POSIX_SPAWN_SETSCHEDPARAM)
124975 {
124976 if (sched_setparam(0, &attrp->posix_attr_schedparam) == -1)
124977 {
124978 exit(127);
124979 }
124980 }

124981 /* Now execute the program at path */
124982 /* Any fd that still has FD_CLOEXEC set will be closed */
124983 execve(path, argv, envp);
124984 exit(127); /* exec failed */
124985 }
124986 else
124987 {
124988 /* This is the parent (calling) process */
124989 if (*pid == (pid_t) - 1)
124990 return errno;
124991 return 0;
124992 }
124993 }

124994 /*****
124995 /* Here is a crude but effective implementation of the */
124996 /* file action object operators which store actions as */
124997 /* concatenated token-separated strings. */
124998 *****/
124999 /* Create object with no actions. */
125000 int posix_spawn_file_actions_init(
125001 posix_spawn_file_actions_t *file_actions)
125002 {
125003 *file_actions = malloc(sizeof(char));
125004 if (*file_actions == NULL)
125005 return ENOMEM;
125006 strcpy(*file_actions, "");
125007 return 0;
125008 }

125009 /* Free object storage and make invalid. */
125010 int posix_spawn_file_actions_destroy(
125011 posix_spawn_file_actions_t *file_actions)
125012 {
125013 free(*file_actions);
125014 *file_actions = NULL;
125015 return 0;
125016 }

```

```

125017 /* Add a new action string to object. */
125018 static int add_to_file_actions(
125019 posix_spawn_file_actions_t *file_actions, char *new_action)
125020 {
125021 *file_actions = realloc
125022 (*file_actions, strlen(*file_actions) + strlen(new_action) + 1);
125023 if (*file_actions == NULL)
125024 return ENOMEM;
125025 strcat(*file_actions, new_action);
125026 return 0;
125027 }

125028 /* Add a close action to object. */
125029 int posix_spawn_file_actions_addclose(
125030 posix_spawn_file_actions_t *file_actions, int fildes)
125031 {
125032 char temp[100];
125033 sprintf(temp, "close(%d)", fildes);
125034 return add_to_file_actions(file_actions, temp);
125035 }

125036 /* Add a dup2 action to object. */
125037 int posix_spawn_file_actions_adddup2(
125038 posix_spawn_file_actions_t *file_actions, int fildes,
125039 int newfildes)
125040 {
125041 char temp[100];
125042 sprintf(temp, "dup2(%d,%d)", fildes, newfildes);
125043 return add_to_file_actions(file_actions, temp);
125044 }

125045 /* Add an open action to object. */
125046 int posix_spawn_file_actions_addopen(
125047 posix_spawn_file_actions_t *file_actions, int fildes,
125048 const char *path, int oflag, mode_t mode)
125049 {
125050 char temp[100];
125051 sprintf(temp, "open(%d,%s*%o,%o)", fildes, path, oflag, mode);
125052 return add_to_file_actions(file_actions, temp);
125053 }

125054 /*****
125055 /* Here is a crude but effective implementation of the */
125056 /* spawn attributes object functions which manipulate */
125057 /* the individual attributes. */
125058 *****/
125059 /* Initialize object with default values. */
125060 int posix_spawnattr_init(posix_spawnattr_t *attr)
125061 {
125062 attr->posix_attr_flags = 0;
125063 attr->posix_attr_pgroup = 0;
125064 /* Default value of signal mask is the parent's signal mask; */
125065 /* other values are also allowed */

```



```

125066 sigprocmask(0, NULL, &attr->posix_attr_sigmask);
125067 sigemptyset(&attr->posix_attr_sigdefault);
125068 /* Default values of scheduling attr inherited from the parent; */
125069 /* other values are also allowed */
125070 attr->posix_attr_schedpolicy = sched_getscheduler(0);
125071 sched_getparam(0, &attr->posix_attr_schedparam);
125072 return 0;
125073 }

125074 int posix_spawnattr_destroy(posix_spawnattr_t *attr)
125075 {
125076 /* No action needed */
125077 return 0;
125078 }

125079 int posix_spawnattr_getflags(const posix_spawnattr_t *attr,
125080 short *flags)
125081 {
125082 *flags = attr->posix_attr_flags;
125083 return 0;
125084 }

125085 int posix_spawnattr_setflags(posix_spawnattr_t *attr, short flags)
125086 {
125087 attr->posix_attr_flags = flags;
125088 return 0;
125089 }

125090 int posix_spawnattr_getpgroup(const posix_spawnattr_t *attr,
125091 pid_t *pgroup)
125092 {
125093 *pgroup = attr->posix_attr_pgroup;
125094 return 0;
125095 }

125096 int posix_spawnattr_setpgroup(posix_spawnattr_t *attr, pid_t pgroup)
125097 {
125098 attr->posix_attr_pgroup = pgroup;
125099 return 0;
125100 }

125101 int posix_spawnattr_getschedpolicy(const posix_spawnattr_t *attr,
125102 int *schedpolicy)
125103 {
125104 *schedpolicy = attr->posix_attr_schedpolicy;
125105 return 0;
125106 }

125107 int posix_spawnattr_setschedpolicy(posix_spawnattr_t *attr,
125108 int schedpolicy)
125109 {
125110 attr->posix_attr_schedpolicy = schedpolicy;
125111 return 0;
125112 }

125113 int posix_spawnattr_getschedparam(const posix_spawnattr_t *attr,
125114 struct sched_param *schedparam)

```

```

125115 {
125116 *schedparam = attr->posix_attr_schedparam;
125117 return 0;
125118 }

125119 int posix_spawnattr_setschedparam(posix_spawnattr_t *attr,
125120 const struct sched_param *schedparam)
125121 {
125122 attr->posix_attr_schedparam = *schedparam;
125123 return 0;
125124 }

125125 int posix_spawnattr_getsigmask(const posix_spawnattr_t *attr,
125126 sigset_t *sigmask)
125127 {
125128 *sigmask = attr->posix_attr_sigmask;
125129 return 0;
125130 }

125131 int posix_spawnattr_setsigmask(posix_spawnattr_t *attr,
125132 const sigset_t *sigmask)
125133 {
125134 attr->posix_attr_sigmask = *sigmask;
125135 return 0;
125136 }

125137 int posix_spawnattr_getsigdefault(const posix_spawnattr_t *attr,
125138 sigset_t *sigdefault)
125139 {
125140 *sigdefault = attr->posix_attr_sigdefault;
125141 return 0;
125142 }

125143 int posix_spawnattr_setsigdefault(posix_spawnattr_t *attr,
125144 const sigset_t *sigdefault)
125145 {
125146 attr->posix_attr_sigdefault = *sigdefault;
125147 return 0;
125148 }

```

## 125149 **I/O Redirection with Spawn**

125150 I/O redirection with *posix\_spawn()* or *posix\_spawnnp()* is accomplished by crafting a *file\_actions*  
125151 argument to effect the desired redirection. Such a redirection follows the general outline of the  
125152 following example:

```

125153 /* To redirect new standard output (fd 1) to a file, */
125154 /* and redirect new standard input (fd 0) from my fd socket_pair[1], */
125155 /* and close my fd socket_pair[0] in the new process. */
125156 posix_spawn_file_actions_t file_actions;
125157 posix_spawn_file_actions_init(&file_actions);
125158 posix_spawn_file_actions_addopen(&file_actions, 1, "newout", ...);
125159 posix_spawn_file_actions_dup2(&file_actions, socket_pair[1], 0);
125160 posix_spawn_file_actions_close(&file_actions, socket_pair[0]);
125161 posix_spawn_file_actions_close(&file_actions, socket_pair[1]);
125162 posix_spawn(..., &file_actions, ...);

```

125163        `posix_spawn_file_actions_destroy(&file_actions);`

125164        **Spawning a Process Under a New User ID**

125165        Spawning a process under a new user ID follows the outline shown in the following example:

125166        `Save = getuid();`  
125167        `setuid(newid);`  
125168        `posix_spawn(...);`  
125169        `setuid(Save);`



125170

# *Rationale (Informative)*

125171

## **Part C:**

125172

## **Shell and Utilities**

125173

*The Open Group*

125174

*The Institute of Electrical and Electronics Engineers, Inc.*



125175

125176

*Rationale for Shell and Utilities*125177 **C.1 Introduction**125178 **C.1.1 Change History**

125179 The change history is provided as an informative section, to track changes from earlier versions  
125180 of this standard.

125181 The following sections describe changes made to the Shell and Utilities volume of POSIX.1-2008  
125182 since Issue 6 of the base document. The CHANGE HISTORY section for each utility describes  
125183 technical changes made to that utility from Issue 5. Changes between earlier versions of the base  
125184 document and Issue 5 are not included.

125185 **Changes from Issue 6 to Issue 7 (POSIX.1-2008)**

125186 The following list summarizes the major changes that were made in the Shell and Utilities  
125187 volume of POSIX.1-2008 from Issue 6 to Issue 7:

- 125188 • Austin Group defect reports, IEEE Interpretations against IEEE Std 1003.1, and responses  
125189 to ISO/IEC defect reports against ISO/IEC 9945 are applied.
- 125190 • The Open Group corrigenda and resolutions are applied.
- 125191 • Features, marked legacy or obsolescent in the base document, have been considered for  
125192 removal in this version.
- 125193 • A review of the use of fixed pathnames within the standard has been undertaken; for  
125194 example, the *at*, *batch*, and *crontab* utilities previously had a requirement for use of the  
125195 directory **/usr/lib/cron**.
- 125196 • The options within the standard have been revised.
  - 125197 — The Batch Environment Services and Utilities option is marked obsolescent.
  - 125198 — The UUCP utilities option is added.
  - 125199 — The User Portability Utilities option is revised so that only the *bg*, *ex*, *fc*, *fg*, *jobs*, *more*,  
125200 *talk*, and *vi* utilities are included, the rest being moved to the Base.

125201 **New Features in Issue 7**

125202 There are no new utilities in Issue 7.

125203 **C.1.2 Relationship to Other Documents**

125204 *C.1.2.1 System Interfaces*

125205 It has been pointed out that the Shell and Utilities volume of POSIX.1-2008 assumes that a great  
125206 deal of functionality from the System Interfaces volume of POSIX.1-2008 is present, but never  
125207 states exactly how much (and strictly does not need to since both are mandated on a conforming  
125208 system). This section is an attempt to clarify the assumptions.

125209 **File Read, Write, and Creation**

125210 IEEE Std 1003.1-2001/Cor 2-2004, item XCU/TC2/D6/2 is applied, updating Table 1-1.

125211 **File Removal**

125212 This is intended to be a summary of the *unlink()* and *rmdir()* requirements. Note that it is  
125213 possible using the *unlink()* function for item 4. to occur.

125214 *C.1.2.2 Concepts Derived from the ISO C Standard*

125215 This section was introduced to address the issue that there was insufficient detail presented by  
125216 such utilities as *awk* or *sh* about their procedural control statements and their methods of  
125217 performing arithmetic functions.

125218 The ISO C standard was selected as a model because most historical implementations of the  
125219 standard utilities were written in C. Thus, it was more likely that they would act in the desired  
125220 manner without modification.

125221 Using the ISO C standard is primarily a notational convenience so that the many procedural  
125222 languages in the Shell and Utilities volume of POSIX.1-2008 would not have to be rigorously  
125223 described in every aspect. Its selection does not require that the standard utilities be written in  
125224 Standard C; they could be written in Common Usage C, Ada, Pascal, assembler language, or  
125225 anything else.

125226 The sizes of the various numeric values refer to C-language data types that are allowed to be  
125227 different sizes by the ISO C standard. Thus, like a C-language application, a shell application  
125228 cannot rely on their exact size. However, it can rely on their minimum sizes expressed in the  
125229 ISO C standard, such as {LONG\_MAX} for a **long** type.

125230 The behavior on overflow is undefined for ISO C standard arithmetic. Therefore, the standard  
125231 utilities can use “bignum” representation for integers so that there is no fixed maximum unless  
125232 otherwise stated in the utility description. Similarly, standard utilities can use infinite-precision  
125233 representations for floating-point arithmetic, as long as these representations exceed the ISO C  
125234 standard requirements.

125235 This section addresses only the issue of semantics; it is not intended to specify syntax. For  
125236 example, the ISO C standard requires that 0L be recognized as an integer constant equal to zero,  
125237 but utilities such as *awk* and *sh* are not required to recognize 0L (though they are allowed to, as  
125238 an extension).



The ISO C standard requires that a C compiler must issue a diagnostic for constants that are too large to represent. Most standard utilities are not required to issue these diagnostics; for example, the command:

```
diff -C 2147483648 file1 file2
```

has undefined behavior, and the *diff* utility is not required to issue a diagnostic even if the number 2 147 483 648 cannot be represented.

### C.1.3 Utility Limits

This section grew out of an idea that originated with the original POSIX.1, in the tables of system limits for the *sysconf()* and *pathconf()* functions. The idea being that a conforming application can be written to use the most restrictive values that a minimal system can provide, but it should not have to. The values provided represent compromises so that some vendors can use historically limited versions of UNIX system utilities. They are the highest values that a strictly conforming application can assume, given no other information.

However, by using the *getconf* utility or the *sysconf()* function, the elegant application can be tailored to more liberal values on some of the specific instances of specific implementations.

There is no explicitly stated requirement that an implementation provide finite limits for any of these numeric values; the implementation is free to provide essentially unbounded capabilities (where it makes sense), stopping only at reasonable points such as {ULONG\_MAX} (from the ISO C standard). Therefore, applications desiring to tailor themselves to the values on a particular implementation need to be ready for possibly huge values; it may not be a good idea to allocate blindly a buffer for an input line based on the value of {LINE\_MAX}, for instance. However, unlike the System Interfaces volume of POSIX.1-2008, there is no set of limits that return a special indication meaning “unbounded”. The implementation should always return an actual number, even if the number is very large.

The statement:

“It is not guaranteed that the application ...”

is an indication that many of these limits are designed to ensure that implementors design their utilities without arbitrary constraints related to unimaginative programming. There are certainly conditions under which combinations of options can cause failures that would not render an implementation non-conforming. For example, {EXPR\_NEST\_MAX} and {ARG\_MAX} could collide when expressions are large; combinations of {BC\_SCALE\_MAX} and {BC\_DIM\_MAX} could exceed virtual memory.

In the Shell and Utilities volume of POSIX.1-2008, the notion of a limit being guaranteed for the process lifetime, as it is in the System Interfaces volume of POSIX.1-2008, is not as useful to a shell script. The *getconf* utility is probably a process itself, so the guarantee would be without value. Therefore, the Shell and Utilities volume of POSIX.1-2008 requires the guarantee to be for the session lifetime. This will mean that many vendors will either return very conservative values or possibly implement *getconf* as a built-in.

It may seem confusing to have limits that apply only to a single utility grouped into one global section. However, the alternative, which would be to disperse them out into their utility description sections, would cause great difficulty when *sysconf()* and *getconf* were described. Therefore, the standard developers chose the global approach.

Each language binding could provide symbol names that are slightly different from those shown here. For example, the C-Language Binding option adds a leading <underscore> to the symbols as a prefix.

125284 The following comments describe selection criteria for the symbols and their values:

125285 {ARG\_MAX}

125286 This is defined by the System Interfaces volume of POSIX.1-2008. Unfortunately, it is very

125287 difficult for a conforming application to deal with this value, as it does not know how much

125288 of its argument space is being consumed by the environment variables of the user.

125289 {BC\_BASE\_MAX}

125290 {BC\_DIM\_MAX}

125291 {BC\_SCALE\_MAX}

125292 These were originally one value, {BC\_SCALE\_MAX}, but it was unreasonable to link all

125293 three concepts into one limit.

125294 {CHILD\_MAX}

125295 This is defined by the System Interfaces volume of POSIX.1-2008.

125296 {COLL\_WEIGHTS\_MAX}

125297 The weights assigned to **order** can be considered as “passes” through the collation

125298 algorithm.

125299 {EXPR\_NEST\_MAX}

125300 The value for expression nesting was borrowed from the ISO C standard.

125301 {LINE\_MAX}

125302 This is a global limit that affects all utilities, unless otherwise noted. The {MAX\_CANON}

125303 value from the System Interfaces volume of POSIX.1-2008 may further limit input lines from

125304 terminals. The {LINE\_MAX} value was the subject of much debate and is a compromise

125305 between those who wished to have unlimited lines and those who understood that many

125306 historical utilities were written with fixed buffers. Frequently, utility writers selected the

125307 UNIX system constant BUFSIZ to allocate these buffers; therefore, some utilities were

125308 limited to 512 bytes for I/O lines, while others achieved 4 096 bytes or greater.

125309 It should be noted that {LINE\_MAX} applies only to input line length; there is no

125310 requirement in POSIX.1-2008 that limits the length of output lines. Utilities such as *awk*, *sed*,

125311 and *paste* could theoretically construct lines longer than any of the input lines they received,

125312 depending on the options used or the instructions from the application. They are not

125313 required to truncate their output to {LINE\_MAX}. It is the responsibility of the application

125314 to deal with this. If the output of one of those utilities is to be piped into another of the

125315 standard utilities, line length restrictions will have to be considered; the *fold* utility, among

125316 others, could be used to ensure that only reasonable line lengths reach utilities or

125317 applications.

125318 {LINK\_MAX}

125319 This is defined by the System Interfaces volume of POSIX.1-2008.

125320 {MAX\_CANON}

125321 {MAX\_INPUT}

125322 {NAME\_MAX}

125323 {NGROUPS\_MAX}

125324 {OPEN\_MAX}

125325 {PATH\_MAX}

125326 {PIPE\_BUF}

125327 These limits are defined by the System Interfaces volume of POSIX.1-2008. Note that the

125328 byte lengths described by some of these values continue to represent bytes, even if the

125329 applicable character set uses a multi-byte encoding.

125330 {RE\_DUP\_MAX}  
 125331 The value selected is consistent with historical practice. Although the name implies that it  
 125332 applies to all REs, only BREs use the interval notation  $\{m,n\}$  addressed by this limit.

125333 {POSIX2\_SYMLINKS}  
 125334 The {POSIX2\_SYMLINKS} variable indicates that the underlying operating system supports  
 125335 the creation of symbolic links in specific directories. Many of the utilities defined in  
 125336 POSIX.1-2008 that deal with symbolic links do not depend on this value. For example, a  
 125337 utility that follows symbolic links (or does not, as the case may be) will only be affected by a  
 125338 symbolic link if it encounters one. Presumably, a file system that does not support symbolic  
 125339 links will not contain any. This variable does affect such utilities as *ln -s* and *pax* that  
 125340 attempt to create symbolic links.

125341 There are different limits associated with command lines and input to utilities, depending on the  
 125342 method of invocation. In the case of a C program *exec*-ing a utility, {ARG\_MAX} is the  
 125343 underlying limit. In the case of the shell reading a script and *exec*-ing a utility, {LINE\_MAX}  
 125344 limits the length of lines the shell is required to process, and {ARG\_MAX} will still be a limit. If a  
 125345 user is entering a command on a terminal to the shell, requesting that it invoke the utility,  
 125346 {MAX\_INPUT} may restrict the length of the line that can be given to the shell to a value below  
 125347 {LINE\_MAX}.

125348 When an option is supported, *getconf* returns a value of 1. For example, when C development is  
 125349 supported:

```
125350 if ["$(getconf POSIX2_C_DEV)" -eq 1]; then
125351 echo C supported
125352 fi
```

125353 The *sysconf()* function in the C-Language Binding option would return 1.

125354 The following comments describe selection criteria for the symbols and their values:

125355 POSIX2\_C\_BIND  
 125356 POSIX2\_C\_DEV  
 125357 POSIX2\_FORT\_DEV  
 125358 POSIX2\_FORT\_RUN  
 125359 POSIX2\_SW\_DEV  
 125360 POSIX2\_UPE

125361 It is possible for some (usually privileged) operations to remove utilities that support these  
 125362 options or otherwise to render these options unsupported. The header files, the *sysconf()*  
 125363 function, or the *getconf* utility will not necessarily detect such actions, in which case they  
 125364 should not be considered as rendering the implementation non-conforming. A test suite  
 125365 should not attempt tests such as:

```
125366 rm /usr/bin/c99
125367 getconf POSIX2_C_DEV
```

125368 POSIX2\_LOCALEDEF  
 125369 This symbol was introduced to allow implementations to restrict supported locales to only  
 125370 those supplied by the implementation.

125371 IEEE Std 1003.1-2001/Cor 1-2002, item XCU/TC1/D6/2 is applied, deleting the entry for  
 125372 {POSIX2\_VERSION} since it is not a utility limit minimum value.

125373 IEEE Std 1003.1-2001/Cor 1-2002, item XCU/TC1/D6/3 is applied, changing the text in Utility  
 125374 Limits from: “utility (see *getconf*) through the *sysconf()* function defined in the System Interfaces  
 125375 volume of POSIX.1-2008. The literal names shown in Table 1-3 apply only to the *getconf* utility;  
 125376 the high-level language binding describes the exact form of each name to be used by the

125377 interfaces in that binding.” to: “utility (see *getconf*).”.

#### 125378 **C.1.4 Grammar Conventions**

125379 There is no additional rationale provided for this section.

#### 125380 **C.1.5 Utility Description Defaults**

125381 This section is arranged with headings in the same order as all the utility descriptions. It is a  
125382 collection of related and unrelated information concerning:

- 125383 1. The default actions of utilities
- 125384 2. The meanings of notations used in POSIX.1-2008 that are specific to individual utility  
125385 sections

125386 Although this material may seem out of place here, it is important that this information appear  
125387 before any of the utilities to be described later.

#### 125388 **NAME**

125389 There is no additional rationale provided for this section.

#### 125390 **SYNOPSIS**

125391 There is no additional rationale provided for this section.

#### 125392 **DESCRIPTION**

125393 There is no additional rationale provided for this section.

#### 125394 **OPTIONS**

125395 Although it has not always been possible, the standard developers tried to avoid repeating  
125396 information to reduce the risk that duplicate explanations could each be modified differently.

125397 The need to recognize `--` is required because conforming applications need to shield their  
125398 operands from any arbitrary options that the implementation may provide as an extension. For  
125399 example, if the standard utility *foo* is listed as taking no options, and the application needed to  
125400 give it a pathname with a leading `<hyphen>`, it could safely do it as:

125401 `foo -- -myfile`

125402 and avoid any problems with `-m` used as an extension.

#### 125403 **OPERANDS**

125404 The usage of `-` is never shown in the SYNOPSIS. Similarly, the usage of `--` is never shown.

125405 The requirement for processing operands in command-line order is to avoid a “WeirdNIX”  
125406 utility that might choose to sort the input files alphabetically, by size, or by directory order.  
125407 Although this might be acceptable for some utilities, in general the programmer has a right to  
125408 know exactly what order will be chosen.

125409 Some of the standard utilities take multiple *file* operands and act as if they were processing the  
125410 concatenation of those files. For example:

125411 `asa file1 file2`

125412 and:

125413 `cat file1 file2 | asa`

125414 have similar results when questions of file access, errors, and performance are ignored. Other  
 125415 utilities such as *grep* or *wc* have completely different results in these two cases. This latter type of  
 125416 utility is always identified in its DESCRIPTION or OPERANDS sections, whereas the former is  
 125417 not. Although it might be possible to create a general assertion about the former case, the  
 125418 following points must be addressed:

- 125419 • Access times for the files might be different in the operand case *versus* the *cat* case.
- 125420 • The utility may have error messages that are cognizant of the input filename, and this  
 125421 added value should not be suppressed. (As an example, *awk* sets a variable with the  
 125422 filename at each file boundary.)

## 125423 STDIN

125424 There is no additional rationale provided for this section.

## 125425 INPUT FILES

125426 A conforming application cannot assume the following three commands are equivalent:

```
125427 tail -n +2 file
125428 (sed -n 1q; cat) < file
125429 cat file | (sed -n 1q; cat)
```

125430 The second command is equivalent to the first only when the file is seekable. In the third  
 125431 command, if the file offset in the open file description were not unspecified, *sed* would have to  
 125432 be implemented so that it read from the pipe 1 byte at a time or it would have to employ some  
 125433 method to seek backwards on the pipe. Such functionality is not defined currently in POSIX.1  
 125434 and does not exist on all historical systems. Other utilities, such as *head*, *read*, and *sh*, have similar  
 125435 properties, so the restriction is described globally in this section.

125436 The definition of “text file” is strictly enforced for input to the standard utilities; very few of  
 125437 them list exceptions to the undefined results called for here. (Of course, “undefined” here does  
 125438 not mean that historical implementations necessarily have to change to start indicating error  
 125439 conditions. Conforming applications cannot rely on implementations succeeding or failing when  
 125440 non-text files are used.)

125441 The utilities that allow line continuation are generally those that accept input languages, rather  
 125442 than pure data. It would be unusual for an input line of this type to exceed {LINE\_MAX} bytes  
 125443 and unreasonable to require that the implementation allow unlimited accumulation of multiple  
 125444 lines, each of which could reach {LINE\_MAX}. Thus, for a conforming application the total of all  
 125445 the continued lines in a set cannot exceed {LINE\_MAX}.

125446 The format description is intended to be sufficiently rigorous to allow other applications to  
 125447 generate these input files. However, since <blank> characters can legitimately be included in  
 125448 some of the fields described by the standard utilities, particularly in locales other than the POSIX  
 125449 locale, this intent is not always realized.

**ENVIRONMENT VARIABLES**

There is no additional rationale provided for this section.

**ASYNCHRONOUS EVENTS**

Because there is no language prohibiting it, a utility is permitted to catch a signal, perform some additional processing (such as deleting temporary files), restore the default signal action (or action inherited from the parent process), and resignal itself.

**STDOUT**

The format description is intended to be sufficiently rigorous to allow post-processing of output by other programs, particularly by an *awk* or *lex* parser.

**STDERR**

This section does not describe error messages that refer to incorrect operation of the utility. Consider a utility that processes program source code as its input. This section is used to describe messages produced by a correctly operating utility that encounters an error in the program source code on which it is processing. However, a message indicating that the utility had insufficient memory in which to operate would not be described.

Some utilities have traditionally produced warning messages without returning a non-zero exit status; these are specifically noted in their sections. Other utilities shall not write to standard error if they complete successfully, unless the implementation provides some sort of extension to increase the verbosity or debugging level.

The format descriptions are intended to be sufficiently rigorous to allow post-processing of output by other programs.

**OUTPUT FILES**

The format description is intended to be sufficiently rigorous to allow post-processing of output by other programs, particularly by an *awk* or *lex* parser.

Receipt of the SIGQUIT signal should generally cause termination (unless in some debugging mode) that would bypass any attempted recovery actions.

**EXTENDED DESCRIPTION**

There is no additional rationale provided for this section.

**EXIT STATUS**

Note the additional discussion of exit values in *Exit Status for Commands* in the *sh* utility. It describes requirements for returning exit values greater than 125.

A utility may list zero as a successful return, 1 as a failure for a specific reason, and greater than 1 as “an error occurred”. In this case, unspecified conditions may cause a 2 or 3, or other value, to be returned. A strictly conforming application should be written so that it tests for successful exit status values (zero in this case), rather than relying upon the single specific error value listed in POSIX.1-2008. In that way, it will have maximum portability, even on implementations with extensions.

The standard developers are aware that the general non-enumeration of errors makes it difficult to write test suites that test the *incorrect* operation of utilities. There are some historical implementations that have expended effort to provide detailed status messages and a helpful environment to bypass or explain errors, such as prompting, retrying, or ignoring unimportant

125491 syntax errors; other implementations have not. Since there is no realistic way to mandate system  
 125492 behavior in cases of undefined application actions or system problems—in a manner acceptable  
 125493 to all cultures and environments—attention has been limited to the correct operation of utilities  
 125494 by the conforming application. Furthermore, the conforming application does not need detailed  
 125495 information concerning errors that it caused through incorrect usage or that it cannot correct.

125496 There is no description of defaults for this section because all of the standard utilities specify  
 125497 something (or explicitly state “Unspecified”) for exit status.

## 125498 **CONSEQUENCES OF ERRORS**

125499 Several actions are possible when a utility encounters an error condition, depending on the  
 125500 severity of the error and the state of the utility. Included in the possible actions of various  
 125501 utilities are: deletion of temporary or intermediate work files; deletion of incomplete files; and  
 125502 validity checking of the file system or directory.

125503 The text about recursive traversing is meant to ensure that utilities such as *find* process as many  
 125504 files in the hierarchy as they can. They should not abandon all of the hierarchy at the first error  
 125505 and resume with the next command-line operand, but should attempt to keep going.

125506 POSIX.1-2008, Technical Corrigendum 1, XCU/TC1-2008/0001 [150] is applied.

+

## 125507 **APPLICATION USAGE**

125508 This section provides additional caveats, issues, and recommendations to the developer.

## 125509 **EXAMPLES**

125510 This section provides sample usage.

## 125511 **RATIONALE**

125512 There is no additional rationale provided for this section.

## 125513 **FUTURE DIRECTIONS**

125514 FUTURE DIRECTIONS sections act as pointers to related work that may impact the interface in  
 125515 the future, and often cautions the developer to architect the code to account for a change in this  
 125516 area. Note that a future directions statement should not be taken as a commitment to adopt a  
 125517 feature or interface in the future.

## 125518 **SEE ALSO**

125519 There is no additional rationale provided for this section.

## 125520 **CHANGE HISTORY**

125521 There is no additional rationale provided for this section.

## C.1.6 Considerations for Utilities in Support of Files of Arbitrary Size

This section is intended to clarify the requirements for utilities in support of large files.

The utilities listed in this section are utilities which are used to perform administrative tasks such as to create, move, copy, remove, change the permissions, or measure the resources of a file. They are useful both as end-user tools and as utilities invoked by applications during software installation and operation.

The *chgrp*, *chmod*, *chown*, *ln*, and *rm* utilities probably require use of large file-capable versions of *stat()*, *lstat()*, *ftw()*, and the **stat** structure.

The *cat*, *cksum*, *cmp*, *cp*, *dd*, *mv*, *sum*, and *touch* utilities probably require use of large file-capable versions of *creat()*, *open()*, and *fopen()*.

The *cat*, *cksum*, *cmp*, *dd*, *df*, *du*, *ls*, and *sum* utilities may require writing large integer values. For example:

- The *cat* utility might have a **-n** option which counts <newline> characters.
- The *cksum* and *ls* utilities report file sizes.
- The *cmp* utility reports the line number at which the first difference occurs, and also has a **-l** option which reports file offsets.
- The *dd*, *df*, *du*, *ls*, and *sum* utilities report block counts.

The *dd*, *find*, and *test* utilities may need to interpret command arguments that contain 64-bit values. For *dd*, the arguments include *skip=n*, *seek=n*, and *count=n*. For *find*, the arguments include **-sizen**. For *test*, the arguments are those associated with algebraic comparisons.

The *df* utility might need to access large file systems with *statvfs()*.

The *ulimit* utility will need to use large file-capable versions of *getrlimit()* and *setrlimit()* and be able to read and write large integer values.

## C.1.7 Built-In Utilities

All of these utilities can be *exec*-ed. There is no requirement that these utilities are actually built into the shell itself, but many shells need the capability to do so because XCU [Section 2.9.1.1](#) (on page 2339) requires that they be found prior to the *PATH* search. The shell could satisfy its requirements by keeping a list of the names and directly accessing the file-system versions regardless of *PATH*. Providing all of the required functionality for those such as *cd* or *read* would be more difficult.

There were originally three justifications for allowing the omission of *exec*-able versions:

1. It would require wasting space in the file system, at the expense of very small systems. However, it has been pointed out that all 16 utilities in the table can be provided with 16 links to a single-line shell script:
2. It is not logical to require invocation of utilities such as *cd* because they have no value outside the shell environment or cannot be useful in a child process. However, counter-examples always seemed to be available for even the most unusual cases:

```
find . -type d -exec cd {} \; -exec foo {} \;
 (which invokes "foo" on accessible directories)
```

```
ps ... | sed ... | xargs kill
```



125563 `find . -exec true \; -a ...`

125564 (where “true” is used for temporary debugging)

- 125565 3. It is confusing to have a utility such as *kill* that can easily be in the file system in the base  
 125566 standard, but that requires built-in status for the User Portability Utilities option (for the  
 125567 % job control job ID notation). It was decided that it was more appropriate to describe the  
 125568 required functionality (rather than the implementation) to the system implementors and  
 125569 let them decide how to satisfy it.

125570 On the other hand, it was realized that any distinction like this between utilities was not useful  
 125571 to applications, and that the cost to correct it was small. These arguments were ultimately the  
 125572 most effective.

125573 There were varying reasons for including utilities in the table of built-ins:

125574 *alias, fc, unalias*

125575 The functionality of these utilities is performed more simply within the shell itself and that  
 125576 is the model most historical implementations have used.

125577 *bg, fg, jobs*

125578 All of the job control-related utilities are eligible for built-in status because that is the model  
 125579 most historical implementations have used.

125580 *cd, getopts, newgrp, read, umask, wait*

125581 The functionality of these utilities is performed more simply within the context of the  
 125582 current process. An example can be taken from the usage of the *cd* utility. The purpose of  
 125583 the *cd* utility is to change the working directory for subsequent operations. The actions of *cd*  
 125584 affect the process in which *cd* is executed and all subsequent child processes of that process.  
 125585 Based on the POSIX standard process model, changes in the process environment of a child  
 125586 process have no effect on the parent process. If the *cd* utility were executed from a child  
 125587 process, the working directory change would be effective only in the child process. Child  
 125588 processes initiated subsequent to the child process that executed the *cd* utility would not  
 125589 have a changed working directory relative to the parent process.

125590 *command*

125591 This utility was placed in the table primarily to protect scripts that are concerned about  
 125592 their *PATH* being manipulated. The “secure” shell script example in the *command* utility in  
 125593 the Shell and Utilities volume of POSIX.1-2008 would not be possible if a *PATH* change  
 125594 retrieved an alien version of *command*. (An alternative would have been to implement  
 125595 *getconf* as a built-in, but the standard developers considered that it carried too many  
 125596 changing configuration strings to require in the shell.)

125597 *kill* Since *kill* provides optional job control functionality using shell notation (%1, %2, and so on),  
 125598 some implementations would find it extremely difficult to provide this outside the shell.

125599 *true, false*

125600 These are in the table as a courtesy to programmers who wish to use the “while true”  
 125601 shell construct without protecting *true* from *PATH* searches. (It is acknowledged that  
 125602 “while :” also works, but the idiom with *true* is historically pervasive.)

125603 All utilities, including those in the table, are accessible via the *system()* and *popen()* functions in  
 125604 the System Interfaces volume of POSIX.1-2008. There are situations where the return  
 125605 functionality of *system()* and *popen()* is not desirable. Applications that require the exit status of  
 125606 the invoked utility will not be able to use *system()* or *popen()*, since the exit status returned is  
 125607 that of the command language interpreter rather than that of the invoked utility. The alternative  
 125608 for such applications is the use of the *exec* family.

## 125609 C.2 Shell Command Language

### 125610 C.2.1 Shell Introduction

125611 The System V shell was selected as the starting point for the Shell and Utilities volume of  
125612 POSIX.1-2008. The BSD C shell was excluded from consideration for the following reasons:

- 125613 • Most historically portable shell scripts assume the Version 7 Bourne shell, from which the  
125614 System V shell is derived.
- 125615 • The majority of tutorial materials on shell programming assume the System V shell.

125616 The construct "#!" is reserved for implementations wishing to provide that extension. If it were  
125617 not reserved, the Shell and Utilities volume of POSIX.1-2008 would disallow it by forcing it to be  
125618 a comment. As it stands, a strictly conforming application must not use "#!" as the first two  
125619 characters of the file.

### 125620 C.2.2 Quoting

125621 There is no additional rationale provided for this section.

#### 125622 C.2.2.1 Escape Character (Backslash)

125623 There is no additional rationale provided for this section.

#### 125624 C.2.2.2 Single-Quotes

125625 A <backslash> cannot be used to escape a single-quote in a single-quoted string. An embedded  
125626 quote can be created by writing, for example: "'a'\''b'", which yields "a'b". (See XCU  
125627 [Section 2.6.5](#) (on page 2333) for a better understanding of how portions of words are either split  
125628 into fields or remain concatenated.) A single token can be made up of concatenated partial  
125629 strings containing all three kinds of quoting or escaping, thus permitting any combination of  
125630 characters.

#### 125631 C.2.2.3 Double-Quotes

125632 The escaped <newline> used for line continuation is removed entirely from the input and is not  
125633 replaced by any white space. Therefore, it cannot serve as a token separator.

125634 In double-quoting, if a <backslash> is immediately followed by a character that would be  
125635 interpreted as having a special meaning, the <backslash> is deleted and the subsequent  
125636 character is taken literally. If a <backslash> does not precede a character that would have a  
125637 special meaning, it is left in place unmodified and the character immediately following it is also  
125638 left unmodified. Thus, for example:

125639 "\$" -> \$

125640 "\a" -> \a

125641 It would be desirable to include the statement “The characters from an enclosed "\${" to the  
125642 matching '}' shall not be affected by the double-quotes”, similar to the one for "\${)".  
125643 However, historical practice in the System V shell prevents this.

The requirement that double-quotes be matched inside "\${...}" within double-quotes and the rule for finding the matching '}' in XCU [Section 2.6.2](#) (on page 2328) eliminate several subtle inconsistencies in expansion for historical shells in rare cases; for example:

```
"${foo-bar}"
```

yields **bar** when **foo** is not defined, and is an invalid substitution when **foo** is defined, in many historical shells. The differences in processing the "\${...}" form have led to inconsistencies between historical systems. A consequence of this rule is that single-quotes cannot be used to quote the '}' within "\${...}"; for example:

```
unset bar
foo="${bar-' }' "
```

is invalid because the "\${...}" substitution contains an unpaired unescaped single-quote. The `<backslash>` can be used to escape the '}' in this example to achieve the desired result:

```
unset bar
foo="${bar-\} } "
```

The differences in processing the "\${...}" form have led to inconsistencies between the historical System V shell, BSD, and KornShells, and the text in the Shell and Utilities volume of POSIX.1-2008 is an attempt to converge them without breaking too many applications. The only alternative to this compromise between shells would be to make the behavior unspecified whenever the literal characters single-quote, '{', '}', and '"' appear within "\${...}". To write a portable script that uses these values, a user would have to assign variables; for example:

```
squote='\ ' dquote=\" lbrace='{ ' rbrace='}'
${foo-$squote$rbrace$squote}
```

rather than:

```
${foo-"' }' " }
```

Some implementations have allowed the end of the word to terminate the backquoted command substitution, such as in:

```
" `echo hello"
```

This usage is undefined; the matching backquote is required by the Shell and Utilities volume of POSIX.1-2008. The other undefined usage can be illustrated by the example:

```
sh -c '` echo "foo`'
```

The description of the recursive actions involving command substitution can be illustrated with an example. Upon recognizing the introduction of command substitution, the shell parses input (in a new context), gathering the source for the command substitution until an unbalanced ')' or '`' is located. For example, in the following:

```
echo "$(date; echo "
 one")"
```

the double-quote following the *echo* does not terminate the first double-quote; it is part of the command substitution script. Similarly, in:

```
echo "$(echo *)"
```

the `<asterisk>` is not quoted since it is inside command substitution; however:

```
echo "$(echo "*)" "
```

is quoted (and represents the `<asterisk>` character itself).

**C.2.3 Token Recognition**

The "(" and ")" symbols are control operators in the KornShell, used for an alternative syntax of an arithmetic expression command. A conforming application cannot use "(" as a single token (with the exception of the "\$(" form for shell arithmetic).

On some implementations, the symbol "(" is a control operator; its use produces unspecified results. Applications that wish to have nested subshells, such as:

```
((echo Hello); (echo World))
```

must separate the "(" characters into two tokens by including white space between them. Some systems may treat these as invalid arithmetic expressions instead of subshells.

Certain combinations of characters are invalid in portable scripts, as shown in the grammar. Implementations may use these combinations (such as "|&") as valid control operators. Portable scripts cannot rely on receiving errors in all cases where this volume of POSIX.1-2008 indicates that a syntax is invalid.

The (3) rule about combining characters to form operators is not meant to preclude systems from extending the shell language when characters are combined in otherwise invalid ways. Conforming applications cannot use invalid combinations, and test suites should not penalize systems that take advantage of this fact. For example, the unquoted combination "|&" is not valid in a POSIX script, but has a specific KornShell meaning.

The (10) rule about '#' as the current character is the first in the sequence in which a new token is being assembled. The '#' starts a comment only when it is at the beginning of a token. This rule is also written to indicate that the search for the end-of-comment does not consider escaped <newline> specially, so that a comment cannot be continued to the next line.

**C.2.3.1 Alias Substitution**

The alias capability was added because it is widely used in historical implementations by interactive users.

The definition of "alias name" precludes an alias name containing a <slash> character. Since the text applies to the command words of simple commands, reserved words (in their proper places) cannot be confused with aliases.

The placement of alias substitution in token recognition makes it clear that it precedes all of the word expansion steps.

An example concerning trailing <blank> characters and reserved words follows. If the user types:

```
$ alias foo="/bin/ls "
$ alias while="/"
```

The effect of executing:

```
$ while true
> do
> echo "Hello, World"
> done
```

is a never-ending sequence of "Hello, World" strings to the screen. However, if the user types:

```
$ foo while
```

125728 the result is an *ls* listing of */*. Since the alias substitution for **foo** ends in a <space>, the next word  
 125729 is checked for alias substitution. The next word, **while**, has also been aliased, so it is substituted  
 125730 as well. Since it is not in the proper position as a command word, it is not recognized as a  
 125731 reserved word.

125732 If the user types:

125733 **\$** foo; while

125734 **while** retains its normal reserved-word properties.

## 125735 C.2.4 Reserved Words

125736 All reserved words are recognized syntactically as such in the contexts described. However, note  
 125737 that **in** is the only meaningful reserved word after a **case** or **for**; similarly, **in** is not meaningful as  
 125738 the first word of a simple command.

125739 Reserved words are recognized only when they are delimited (that is, meet the definition of XBD  
 125740 Section 3.440, on page 104), whereas operators are themselves delimiters. For instance, ' ( ' and  
 125741 ' ) ' are control operators, so that no <space> is needed in (*list*). However, ' { ' and ' } ' are  
 125742 reserved words in { *list*; }, so that in this case the leading <space> and <semicolon> are required.

125743 The list of unspecified reserved words is from the KornShell, so conforming applications cannot  
 125744 use them in places a reserved word would be recognized. This list contained **time** in early  
 125745 proposals, but it was removed when the *time* utility was selected for the Shell and Utilities  
 125746 volume of POSIX.1-2008.

125747 There was a strong argument for promoting braces to operators (instead of reserved words), so  
 125748 they would be syntactically equivalent to subshell operators. Concerns about compatibility  
 125749 outweighed the advantages of this approach. Nevertheless, conforming applications should  
 125750 consider quoting ' { ' and ' } ' when they represent themselves.

125751 The restriction on ending a name with a <colon> is to allow future implementations that support  
 125752 named labels for flow control; see the RATIONALE for the *break* built-in utility.

125753 It is possible that a future version of the Shell and Utilities volume of POSIX.1-2008 may require  
 125754 that ' { ' and ' } ' be treated individually as control operators, although the token "{ }" will  
 125755 probably be a special-case exemption from this because of the often-used *find*{ } construct.

## 125756 C.2.5 Parameters and Variables

### 125757 C.2.5.1 Positional Parameters

125758 There is no additional rationale provided for this section.

### 125759 C.2.5.2 Special Parameters

125760 Most historical implementations implement subshells by forking; thus, the special parameter  
 125761 ' \$ ' does not necessarily represent the process ID of the shell process executing the commands  
 125762 since the subshell execution environment preserves the value of ' \$ '.

125763 If a subshell were to execute a background command, the value of " \$ ! " for the parent would  
 125764 not change. For example:

```

125765 (
125766 date &
125767 echo $!
125768)
125769 echo $!

```

125770 would echo two different values for "\$!".

125771 The "\$-" special parameter can be used to save and restore *set* options:

```

125772 Save=$(echo $- | sed 's/[ics]//g')
125773 ...
125774 set +aCefnuvx
125775 if [-n "$Save"]; then
125776 set -$Save
125777 fi

```

125778 The three options are removed using *sed* in the example because they may appear in the value of  
 125779 "\$-" (from the *sh* command line), but are not valid options to *set*.

125780 The descriptions of parameters '\*' and '@' assume the reader is familiar with the field  
 125781 splitting discussion in XCU [Section 2.6.5](#) (on page 2333) and understands that portions of the  
 125782 word remain concatenated unless there is some reason to split them into separate fields.

125783 Some examples of the '\*' and '@' properties, including the concatenation aspects:

```

125784 set "abc" "def ghi" "jkl"

125785 echo $* => "abc" "def" "ghi" "jkl"
125786 echo "$*" => "abc def ghi jkl"
125787 echo $@ => "abc" "def" "ghi" "jkl"

```

125788 but:

```

125789 echo "$@" => "abc" "def ghi" "jkl"
125790 echo "xx$@yy" => "xxabc" "def ghi" "jkl"yy"
125791 echo "$@$@" => "abc" "def ghi" "jklabc" "def ghi" "jkl"

```

125792 In the preceding examples, the double-quote characters that appear after the "=>" do not  
 125793 appear in the output and are used only to illustrate word boundaries.

125794 The following example illustrates the effect of setting *IFS* to a null string:

```

125795 $ IFS=''
125796 $ set foo bar bam
125797 $ echo "$@"
125798 foo bar bam
125799 $ echo "$*"
125800 foobarbam
125801 $ unset IFS
125802 $ echo "$*"
125803 foo bar bam

```

## 125804 C.2.5.3 Shell Variables

125805 See the discussion of *IFS* in [Section C.2.6.5](#) (on page 3688) and the RATIONALE for the *sh* utility.

125806 The prohibition on *LC\_CTYPE* changes affecting lexical processing protects the shell  
 125807 implementor (and the shell programmer) from the ill effects of changing the definition of  
 125808 <blank> or the set of alphabetic characters in the current environment. It would probably not be  
 125809 feasible to write a compiled version of a shell script without this rule. The rule applies only to  
 125810 the current invocation of the shell and its subshells—invoking a shell script or performing *exec*  
 125811 *sh* would subject the new shell to the changes in *LC\_CTYPE*.

125812 Other common environment variables used by historical shells are not specified by the Shell and  
 125813 Utilities volume of POSIX.1-2008, but they should be reserved for the historical uses.

125814 Tilde expansion for components of *PATH* in an assignment such as:

125815 `PATH=~hlj/bin:~dwc/bin:$PATH`

125816 is a feature of some historical shells and is allowed by the wording of XCU [Section 2.6.1](#) (on page  
 125817 2327). Note that the <tilde> characters are expanded during the assignment to *PATH*, not when  
 125818 *PATH* is accessed during command search.

125819 The following entries represent additional information about variables included in the Shell and  
 125820 Utilities volume of POSIX.1-2008, or rationale for common variables in use by shells that have  
 125821 been excluded:

125822 — (Underscore.) While <underscore> is historical practice, its overloaded usage  
 125823 in the KornShell is confusing, and it has been omitted from the Shell and  
 125824 Utilities volume of POSIX.1-2008.

125825 *ENV* This variable can be used to set aliases and other items local to the invocation  
 125826 of a shell. The file referred to by *ENV* differs from *\$HOME/.profile* in that  
 125827 *.profile* is typically executed at session start-up, whereas the *ENV* file is  
 125828 executed at the beginning of each shell invocation. The *ENV* value is  
 125829 interpreted in a manner similar to a dot script, in that the commands are  
 125830 executed in the current environment and the file needs to be readable, but not  
 125831 executable. However, unlike dot scripts, no *PATH* searching is performed. This  
 125832 is used as a guard against Trojan Horse security breaches.

125833 *ERRNO* This variable was omitted from the Shell and Utilities volume of POSIX.1-2008  
 125834 because the values of error numbers are not defined in POSIX.1-2008 in a  
 125835 portable manner.

125836 *FCEDIT* Since this variable affects only the *fc* utility, it has been omitted from this more  
 125837 global place. The value of *FCEDIT* does not affect the command-line editing  
 125838 mode in the shell; see the description of *set -o vi* in the *set* built-in utility.

125839 *PS1* This variable is used for interactive prompts. Historically, the “superuser”  
 125840 has had a prompt of ' # '. Since privileges are not required to be monolithic, it  
 125841 is difficult to define which privileges should cause the alternate prompt.  
 125842 However, a sufficiently powerful user should be reminded of that power by  
 125843 having an alternate prompt.

125844 *PS3* This variable is used by the KornShell for the *select* command. Since the POSIX  
 125845 shell does not include *select*, *PS3* was omitted.

125846 *PS4* This variable is used for shell debugging. For example, the following script:

125847 `PS4=' [ ${LINENO} ] + '`  
 125848 `set -x`  
 125849 `echo Hello`

125850 writes the following to standard error:

125851 [3]+ echo Hello

125852 *RANDOM* This pseudo-random number generator was not seen as being useful to

125853 interactive users.

125854 *SECONDS* Although this variable is sometimes used with *PS1* to allow the display of the

125855 current time in the prompt of the user, it is not one that would be manipulated

125856 frequently enough by an interactive user to include in the Shell and Utilities

125857 volume of POSIX.1-2008.

125858 POSIX.1-2008, Technical Corrigendum 1, XCU/TC1-2008/0002 [152] is applied. +

## 125859 C.2.6 Word Expansions

125860 Step (2) refers to the “portions of fields generated by step (1)”. For example, if the word being

125861 expanded were "\$x+\$y" and *IFS*=+, the word would be split only if "\$x" or "\$y" contained

125862 ' + ' ; the ' + ' in the original word was not generated by step (1).

125863 *IFS* is used for performing field splitting on the results of parameter and command substitution;

125864 it is not used for splitting all fields. Earlier versions of the shell used it for splitting all fields

125865 during field splitting, but this has severe problems because the shell can no longer parse its own

125866 script. There are also important security implications caused by this behavior. All useful

125867 applications of *IFS* use it for parsing input of the *read* utility and for splitting the results of

125868 parameter and command substitution.

125869 The rule concerning expansion to a single field requires that if **foo=abc** and **bar=def**, that:

125870 "\$foo"\$bar"

125871 expands to the single field:

125872 abcdef

125873 The rule concerning empty fields can be illustrated by:

```
125874 $ unset foo
125875 $ set $foo bar ' ' xyz "$foo" abc
125876 $ for i
125877 > do
125878 > echo "-$i-"
125879 > done
125880 -bar-
125881 --
125882 -xyz-
125883 --
125884 -abc-
```

125885 Step (1) indicates that parameter expansion, command substitution, and arithmetic expansion

125886 are all processed simultaneously as they are scanned. For example, the following is valid

125887 arithmetic:

```
125888 x=1
125889 echo $(($(echo 3)+$x))
```

125890 An early proposal stated that tilde expansion preceded the other steps, but this is not the case in

125891 known historical implementations; if it were, and if a referenced home directory contained a ' \$ '

125892 character, expansions would result within the directory name.



|        |                                                                                                                 |   |
|--------|-----------------------------------------------------------------------------------------------------------------|---|
| 125893 | POSIX.1-2008, Technical Corrigendum 1, XCU/TC1-2008/0003 [49,430] is applied.                                   | + |
| 125894 | <i>C.2.6.1 Tilde Expansion</i>                                                                                  |   |
| 125895 | Tilde expansion generally occurs only at the beginning of words, but an exception based on                      |   |
| 125896 | historical practice has been included:                                                                          |   |
| 125897 | <code>PATH=/posix/bin:~dgtk/bin</code>                                                                          |   |
| 125898 | This is eligible for tilde expansion because <tilde> follows a <colon> and none of the relevant                 |   |
| 125899 | characters is quoted. Consideration was given to prohibiting this behavior because any of the                   |   |
| 125900 | following are reasonable substitutes:                                                                           |   |
| 125901 | <code>PATH=\$(printf %s ~karels/bin : ~bostic/bin)</code>                                                       |   |
| 125902 | <code>for Dir in ~maat/bin ~srb/bin ...</code>                                                                  |   |
| 125903 | <code>do</code>                                                                                                 |   |
| 125904 | <code>    PATH=\${PATH:+\$PATH:}\$Dir</code>                                                                    |   |
| 125905 | <code>done</code>                                                                                               |   |
| 125906 | In the first command, explicit <colon> characters are used for each directory. In all cases, the                |   |
| 125907 | shell performs tilde expansion on each directory because all are separate words to the shell.                   |   |
| 125908 | Note that expressions in operands such as:                                                                      |   |
| 125909 | <code>make -k mumble LIBDIR=~chet/lib</code>                                                                    |   |
| 125910 | do not qualify as shell variable assignments, and tilde expansion is not performed (unless the                  |   |
| 125911 | command does so itself, which <i>make</i> does not).                                                            |   |
| 125912 | Because of the requirement that the word is not quoted, the following are not equivalent; only                  |   |
| 125913 | the last causes tilde expansion:                                                                                |   |
| 125914 | <code>\~hlj/    ~h\lj/    ~"hlj"/    ~hlj\    ~hlj/</code>                                                      |   |
| 125915 | In an early proposal, tilde expansion occurred following any unquoted <equals-sign> or                          |   |
| 125916 | <colon>, but this was removed because of its complexity and to avoid breaking commands such                     |   |
| 125917 | as:                                                                                                             |   |
| 125918 | <code>rcp hostname:~marc/.profile .</code>                                                                      |   |
| 125919 | System administrators on systems where // has an implementation-defined meaning which is                        | + |
| 125920 | different to /, should not create users with a home directory of / or //, since this may lead to                | + |
| 125921 | unexpected filename resolution on those systems.                                                                | + |
| 125922 | A suggestion was made that the special sequence "\$~" should be allowed to force tilde                          |   |
| 125923 | expansion anywhere. Since this is not historical practice, it has been left for future                          |   |
| 125924 | implementations to evaluate. (The description in XCU <a href="#">Section 2.2</a> (on page 2320) requires that a |   |
| 125925 | <dollar-sign> be quoted to represent itself, so the "\$~" combination is already unspecified.)                  |   |
| 125926 | The results of giving <tilde> with an unknown login name are undefined because the KornShell                    |   |
| 125927 | "~+" and "~-" constructs make use of this condition, but in general it is an error to give an                   |   |
| 125928 | incorrect login name with <tilde>. The results of having <i>HOME</i> unset are unspecified because              |   |
| 125929 | some historical shells treat this as an error.                                                                  |   |
| 125930 | Historically, the Korn shell performed field splitting and pathname expansion on the results of                 |   |
| 125931 | tilde expansion, and earlier versions of this standard reflected this. However, tilde expansion                 |   |
| 125932 | results in a pathname, and performing field splitting and pathname expansion on something                       |   |
| 125933 | that is already a pathname is at best redundant and at worst will change the value from the                     |   |
| 125934 | correct pathname to one or more incorrect ones. Later versions of the Korn shell do not perform                 |   |
| 125935 | these expansions and POSIX.1-2008 has been updated to match. Note that although pathname                        |   |

125936 expansion is not performed on the results of tilde expansion, this does not prevent other parts of  
 125937 the same word from being expanded. For example, ~/a\* expands to all files in \$HOME  
 125938 beginning with 'a'.

#### 125939 C.2.6.2 Parameter Expansion

125940 The rule for finding the closing '}' in "\${...}" is the one used in the KornShell and is  
 125941 upwardly-compatible with the Bourne shell, which does not determine the closing '}' until the  
 125942 word is expanded. The advantage of this is that incomplete expansions, such as:

125943 \${foo

125944 can be determined during tokenization, rather than during expansion.

125945 For rationale regarding expansion of "\${...}" within double-quotes, see [Section C.2.2.3](#) (on +  
 125946 page 3676). +

125947 The string length and substring capabilities were included because of the demonstrated need for  
 125948 them, based on their usage in other shells, such as C shell and KornShell.

125949 Historical versions of the KornShell have not performed tilde expansion on the word part of  
 125950 parameter expansion; however, it is more consistent to do so.

125951 POSIX.1-2008, Technical Corrigendum 1, XCU/TC1-2008/0004 [458], XCU/TC1-2008/0005 +  
 125952 [458], XCU/TC1-2008/0006 [457], XCU/TC1-2008/0007 [457], XCU/TC1-2008/0008 [417], +  
 125953 XCU/TC1-2008/0009 [457], XCU/TC1-2008/0010 [457], XCU/TC1-2008/0011 [457], +  
 125954 XCU/TC1-2008/0012 [457], XCU/TC1-2008/0013 [457], XCU/TC1-2008/0014 [457], +  
 125955 XCU/TC1-2008/0015 [457], XCU/TC1-2008/0016 [457], XCU/TC1-2008/0017 [457], and +  
 125956 XCU/TC1-2008/0018 [458] are applied.

#### 125957 C.2.6.3 Command Substitution

125958 The "\$()" form of command substitution solves a problem of inconsistent behavior when using  
 125959 backquotes. For example:

| 125960 | Command              | Output |
|--------|----------------------|--------|
| 125961 | echo '\\$x'          | \\$x   |
| 125962 | echo `echo '\\$x'`   | \$x    |
| 125963 | echo \$(echo '\\$x') | \\$x   |

125964 Additionally, the backquoted syntax has historical restrictions on the contents of the embedded  
 125965 command. While the newer "\$()" form can process any kind of valid embedded script, the  
 125966 backquoted form cannot handle some valid scripts that include backquotes. For example, these  
 125967 otherwise valid embedded scripts do not work in the left column, but do work on the right:

|        |                             |                             |
|--------|-----------------------------|-----------------------------|
| 125968 | echo `                      | echo \$(                    |
| 125969 | cat <<\eof                  | cat <<\eof                  |
| 125970 | a here-doc with `           | a here-doc with )           |
| 125971 | eof                         | eof                         |
| 125972 | `                           | )                           |
| 125973 | echo `                      | echo \$(                    |
| 125974 | echo abc # a comment with ` | echo abc # a comment with ) |
| 125975 | `                           | )                           |
| 125976 | echo `                      | echo \$(                    |
| 125977 | echo ` `                    | echo ` `                    |

Because of these inconsistent behaviors, the backquoted variety of command substitution is not recommended for new applications that nest command substitutions or attempt to embed complex scripts.

The KornShell feature:

If *command* is of the form *<word*, *word* is expanded to generate a pathname, and the value of the command substitution is the contents of this file with any trailing <newline> characters deleted.

was omitted from the Shell and Utilities volume of POSIX.1-2008 because *\$(cat word)* is an appropriate substitute. However, to prevent breaking numerous scripts relying on this feature, it is unspecified to have a script within "*\$ ( )*" that has only redirections.

Arithmetic expansions have precedence over command substitutions. That is, if the shell can parse an expansion beginning with "*\$ ( (*" as an arithmetic expansion then it will do so. It will only parse the expansion as a command substitution (that starts with a subshell) if it determines that it cannot parse the expansion as an arithmetic expansion. If the syntax is valid for neither type of expansion, then it is unspecified what kind of syntax error the shell reports.

How well the shell performs this determination is a quality of implementation issue. Current shell implementations use heuristics. In particular, the shell need not evaluate nested expansions when determining whether it can parse an expansion beginning with "*\$ ( (*" as an arithmetic expansion. For example:

```
$ ((a $op b))
```

is always an arithmetic expansion if "*\$op*" expands to, say, *' ++ '*, but if "*\$op*" expands to *' ( '* then the shell might still parse the expansion as an arithmetic expansion (resulting in a syntax error due to unbalanced parentheses) or it might perform a command substitution.

This standard requires that conforming applications always separate the "*\$ (*" and *' (* with white space when a command substitution starts with a subshell. This is because implementations may support extensions in arithmetic expressions which could result in the shell parsing the input as an arithmetic expansion even though a minimally conforming shell would not. For example, many shells support arrays with the array index (which can be an expression) in square brackets. Therefore, the presence of "*myfile[0-9]*" within an expansion beginning "*\$ ( (*" is no guarantee that it will be parsed as a command substitution.

The ambiguity is not restricted to the simple case of a single subshell. More complicated ambiguous cases are possible (even with just the standard shell syntax), such as:

```
$ ((cat <<EOH
+ ((
EOH
) && (cat <<EOH
)) + 1 +
EOH
))
```

This can be parsed as an arithmetic expansion, with *cat* and *EOH* as the names of shell variables. Ambiguous cases also exist where the end of the expansion is at a different location for the arithmetic expansion and the command substitution:

```
$ ((cat <<EOF
+ (((
EOF
```

|        |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |   |
|--------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---|
| 126024 | ) && (                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |   |
| 126025 | cat <<EOF                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |   |
| 126026 | +                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |   |
| 126027 | EOF                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |   |
| 126028 | ) )                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |   |
| 126029 | This is an incomplete arithmetic expansion, but would have been a (complete) command substitution if it could not have been parsed as an arithmetic expansion. If this expansion occurs at the end of input then the shell reports a syntax error; it does not parse it as a command substitution.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |   |
| 126030 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |   |
| 126031 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |   |
| 126032 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |   |
| 126033 | IEEE Std 1003.1-2001/Cor 1-2002, item XCU/TC1/D6/4 is applied, changing the text from: "If a command substitution occurs inside double-quotes, it shall not be performed on the results of the substitution." to: "If a command substitution occurs inside double-quotes, field splitting and pathname expansion shall not be performed on the results of the substitution.". The replacement text taken from the ISO POSIX-2: 1993 standard is clearer about the items that are not performed.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |   |
| 126034 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |   |
| 126035 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |   |
| 126036 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |   |
| 126037 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |   |
| 126038 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |   |
| 126039 | SD5-XCU-ERN-84 is applied, clarifying how the search for the matching backquote is satisfied.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |   |
| 126040 | POSIX.1-2008, Technical Corrigendum 1, XCU/TC1-2008/0019 [217] is applied.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                | + |
| 126041 | <i>C.2.6.4 Arithmetic Expansion</i>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |   |
| 126042 | The standard developers agreed that there was a strong desire for some kind of arithmetic evaluator to provide functionality similar to <i>expr</i> , that relating it to ' \$ ' makes it work well with the standard shell language and provides access to arithmetic evaluation in places where accessing a utility would be inconvenient.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |   |
| 126043 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |   |
| 126044 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |   |
| 126045 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |   |
| 126046 | The syntax and semantics for arithmetic were revised for the ISO/IEC 9945-2:1993 standard. The language represents a simple subset of the previous arithmetic language (which was derived from the KornShell " ( ) " construct). The syntax was changed from that of a command denoted by (( <i>expression</i> )) to an expansion denoted by \$( <i>expression</i> )). The new form is a dollar expansion ( ' \$ ' ) that evaluates the expression and substitutes the resulting value. Objections to the previous style of arithmetic included that it was too complicated, did not fit in well with the use of variables in the shell, and its syntax conflicted with subshells. The justification for the new syntax is that the shell is traditionally a macro language, and if a new feature is to be added, it should be accomplished by extending the capabilities presented by the current model of the shell, rather than by inventing a new one outside the model; adding a new dollar expansion was perceived to be the most intuitive and least destructive way to add such a new capability. |   |
| 126047 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |   |
| 126048 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |   |
| 126049 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |   |
| 126050 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |   |
| 126051 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |   |
| 126052 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |   |
| 126053 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |   |
| 126054 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |   |
| 126055 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |   |
| 126056 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |   |
| 126057 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |   |
| 126058 | The standard requires assignment operators to be supported (as listed in XCU <a href="#">Section 1.1.2</a> , on page 2305), and since arithmetic expansions are not specified to be evaluated in a subshell environment, changes to variables there have to be in effect after the arithmetic expansion, just as in the parameter expansion "\${x=value}" .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |   |
| 126059 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |   |
| 126060 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |   |
| 126061 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |   |
| 126062 | Note, however, that "\$ ( ( x=5 ) )" need not be equivalent to "\$ ( ( \$x=5 ) )" . If the value of the environment variable <i>x</i> is the string " <i>y</i> =", the expansion of "\$ ( ( x=5 ) )" would set <i>x</i> to 5 and output 5, but "\$ ( ( \$x=5 ) )" would output 0 if the value of the environment variable <i>y</i> is not 5 and would output 1 if the environment variable <i>y</i> is 5. Similarly, if the value of the environment variable is 4, the expansion of "\$ ( ( x=5 ) )" would still set <i>x</i> to 5 and output 5, but "\$ ( ( \$x=5 ) )" (which would be equivalent to "\$ ( ( 4=5 ) )" ) would yield a syntax error.                                                                                                                                                                                                                                                                                                                                                                                                                                                     |   |
| 126063 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |   |
| 126064 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |   |
| 126065 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |   |
| 126066 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |   |
| 126067 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |   |
| 126068 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |   |
| 126069 | In early proposals, a form <code>\$(<i>expression</i>)</code> was used. It was functionally equivalent to the "\$ ( ) " of the current text, but objections were lodged that the 1988 KornShell had already implemented                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |   |
| 126070 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |   |

126071 "\$ ( ) )" and there was no compelling reason to invent yet another syntax. Furthermore, the  
 126072 "\$ [ ]" syntax had a minor incompatibility involving the patterns in **case** statements.

126073 The portion of the ISO C standard arithmetic operations selected corresponds to the operations  
 126074 historically supported in the KornShell. In addition to the exceptions listed in XCU [Section 2.6.4](#)  
 126075 (on page 2332), the use of the following are explicitly outside the scope of the rules defined in  
 126076 XCU [Section 1.1.2.1](#) (on page 2305):

- 126077 • The prefix operator ' & ' and the " [ ] ", "-> ", and ' . ' operators.
- 126078 • Casts

126079 It was concluded that the *test* command (**I**) was sufficient for the majority of relational arithmetic  
 126080 tests, and that tests involving complicated relational expressions within the shell are rare, yet  
 126081 could still be accommodated by testing the value of "\$ ( ) )" itself. For example:

```
126082 # a complicated relational expression
126083 while [$((($x + $y)/($a * $b)) < ($foo*$bar))) -ne 0]
```

126084 or better yet, the rare script that has many complex relational expressions could define a  
 126085 function like this:

```
126086 val() {
126087 return $(!$1)
126088 }
```

126089 and complicated tests would be less intimidating:

```
126090 while val $((($x + $y)/($a * $b)) < ($foo*$bar)))
126091 do
126092 # some calculations
126093 done
```

126094 A suggestion that was not adopted was to modify *true* and *false* to take an optional argument,  
 126095 and *true* would exit true only if the argument was non-zero, and *false* would exit false only if the  
 126096 argument was non-zero:

```
126097 while true $(($x > 5 && $y <= 25))
```

126098 There is a minor portability concern with the new syntax. The example "\$ ( (2+2) )" could have  
 126099 been intended to mean a command substitution of a utility named "2+2" in a subshell. The  
 126100 standard developers considered this to be obscure and isolated to some KornShell scripts  
 126101 (because "\$ ( )" command substitution existed previously only in the KornShell). The text on  
 126102 command substitution requires that the "\$ ( " and ' ( ' be separate tokens if this usage is  
 126103 needed.

126104 An example such as:

```
126105 echo $((echo hi);(echo there))
```

126106 should not be misinterpreted by the shell as arithmetic because attempts to balance the  
 126107 parentheses pairs would indicate that they are subshells. However, as indicated by XBD [Section 3.113](#)  
 126108 (on page 51), a conforming application must separate two adjacent parentheses with white  
 126109 space to indicate nested subshells.

126110 The standard is intentionally silent about how a variable's numeric value in an expression is  
 126111 determined from its normal "sequence of bytes" value. It could be done as a text substitution, as  
 126112 a conversion like that performed by *strtol()*, or even recursive evaluation. Therefore, the only  
 126113 cases for which the standard is clear are those for which both conversions produce the same  
 126114 result. The cases where they give the same result are those where the sequence of bytes form a  
 126115 valid integer constant. Therefore, if a variable does not contain a valid integer constant, the

126116 behavior is unspecified.

126117 For the commands:

126118 `x=010; echo $((x += 1))`

126119 the output must be 9.

126120 For the commands:

126121 `x=' 1'; echo $((x += 1))`

126122 the results are unspecified.

126123 For the commands:

126124 `x=1+1; echo $((x += 1))`

126125 the results are unspecified.

126126 Although the ISO/IEC 9899:1999 standard now requires support for **long long** and allows  
 126127 extended integer types with higher ranks, POSIX.1-2008 only requires arithmetic expansions to  
 126128 support **signed long** integer arithmetic. Implementations are encouraged to support signed  
 126129 integer values at least as large as the size of the largest file allowed on the implementation.

126130 Implementations are also allowed to perform floating-point evaluations as long as an  
 126131 application won't see different results for expressions that would not overflow **signed long**  
 126132 integer expression evaluation. (This includes appropriate truncation of results to integer values.)

126133 Changes made in response to IEEE PASC Interpretation 1003.2 #208 removed the requirement  
 126134 that the integer constant suffixes `l` and `L` had to be recognized. The ISO POSIX-2: 1993 standard  
 126135 did not require the `u`, `ul`, `uL`, `U`, `Ul`, `UL`, `lu`, `lU`, `Lu`, and `LU` suffixes since only signed integer  
 126136 arithmetic was required. Since all arithmetic expressions were treated as handling **signed long**  
 126137 integer types anyway, the `l` and `L` suffixes were redundant. No known scripts used them and  
 126138 some historic shells did not support them. When the ISO/IEC 9899:1999 standard was used as  
 126139 the basis for the description of arithmetic processing, the `ll` and `LL` suffixes and combinations  
 126140 were also not required. Implementations are still free to accept any or all of these suffixes, but  
 126141 are not required to do so.

126142 There was also some confusion as to whether the shell was required to recognize character  
 126143 constants. Syntactically, character constants were required to be recognized, but the  
 126144 requirements for the handling of `<backslash>` and single-quote characters (needed to specify  
 126145 character constants) within an arithmetic expansion were ambiguous. Furthermore, no known  
 126146 shells supported them. Changes made in response to IEEE PASC Interpretation 1003.2 #208  
 126147 removed the requirement to support them (if they were indeed required before). POSIX.1-2008  
 126148 clearly does not require support for character constants.

126149 IEEE Std 1003.1-2001/Cor 2-2004, item XCU/TC2/D6/3 is applied, clarifying arithmetic  
 126150 expressions.

126151 POSIX.1-2008, Technical Corrigendum 1, XCU/TC1-2008/0020 [50] is applied.

+

#### 126152 C.2.6.5 *Field Splitting*

126153 The operation of field splitting using *IFS*, as described in early proposals, was based on the way  
 126154 the KornShell splits words, but it is incompatible with other common versions of the shell.  
 126155 However, each has merit, and so a decision was made to allow both. If the *IFS* variable is unset  
 126156 or is `<space><tab><newline>`, the operation is equivalent to the way the System V shell splits  
 126157 words. Using characters outside the `<space><tab><newline>` set yields the KornShell behavior,  
 126158 where each of the non-`<space><tab><newline>`s is significant. This behavior, which affords the

126159 most flexibility, was taken from the way the original *awk* handled field splitting.

126160 Rule (3) can be summarized as a pseudo-ERE:

126161  $(s^*ns^* | s^+)$

126162 where *s* is an *IFS* white-space character and *n* is a character in the *IFS* that is not white space.  
 126163 Any string matching that ERE delimits a field, except that the *s+* form does not delimit fields at  
 126164 the beginning or the end of a line. For example, if *IFS* is <space>/<comma>/<tab>, the string:

126165 <space><space>red<space><space>, <space>white<space>blue

126166 yields the three colors as the delimited fields.

#### 126167 C.2.6.6 Pathname Expansion

126168 There is no additional rationale provided for this section.

#### 126169 C.2.6.7 Quote Removal

126170 There is no additional rationale provided for this section.

### 126171 C.2.7 Redirection

126172 In the System Interfaces volume of POSIX.1-2008, file descriptors are integers in the range  
 126173 0–(OPEN\_MAX–1). The file descriptors discussed in XCU [Section 2.7](#) (on page 2334) are that  
 126174 same set of small integers.

126175 Having multi-digit file descriptor numbers for I/O redirection can cause some obscure  
 126176 compatibility problems. Specifically, scripts that depend on an example command:

126177 `echo 22>/dev/null`

126178 echoing "2" to standard error or "22" to standard output are no longer portable. However, the  
 126179 file descriptor number must still be delimited from the preceding text. For example:

126180 `cat file2>foo`

126181 writes the contents of **file2**, not the contents of **file**.

126182 The ">|" format of output redirection was adopted from the KornShell. Along with the  
 126183 *noclobber* option, *set -C*, it provides a safety feature to prevent inadvertent overwriting of  
 126184 existing files. (See the RATIONALE for the *pathchk* utility for why this step was taken.) The  
 126185 restriction on regular files is historical practice.

126186 The System V shell and the KornShell have differed historically on pathname expansion of *word*;  
 126187 the former never performed it, the latter only when the result was a single field (file). As a  
 126188 compromise, it was decided that the KornShell functionality was useful, but only as a shorthand  
 126189 device for interactive users. No reasonable shell script would be written with a command such  
 126190 as:

126191 `cat foo > a*`

126192 Thus, shell scripts are prohibited from doing it, while interactive users can select the shell with  
 126193 which they are most comfortable.

126194 The construct "2>&1" is often used to redirect standard error to the same file as standard  
 126195 output. Since the redirections take place beginning to end, the order of redirections is significant.  
 126196 For example:

126197 `ls > foo 2>&1`

126198 directs both standard output and standard error to file **foo**. However:

126199 `ls 2>&1 > foo`

126200 only directs standard output to file **foo** because standard error was duplicated as standard  
126201 output before standard output was directed to file **foo**.

126202 Applications should not use the `[n]<&-` or `[n]>&-` operators to execute a utility or application +  
126203 with file descriptor 0 not open for reading or with file descriptor 1 or 2 not open for writing, as +  
126204 this might cause the executed program (or shell built-in) to misbehave. In order not to pass on +  
126205 these file descriptors to an executed utility or application, applications should not just close +  
126206 them but should reopen them on, for example, **/dev/null**. Some implementations may reopen +  
126207 them automatically, but applications should not rely on this being done. +

126208 The "`<>`" operator could be useful in writing an application that worked with several terminals,  
126209 and occasionally wanted to start up a shell. That shell would in turn be unable to run  
126210 applications that run from an ordinary controlling terminal unless it could make use of "`<>`"  
126211 redirection. The specific example is a historical version of the pager *more*, which reads from  
126212 standard error to get its commands, so standard input and standard output are both available  
126213 for their usual usage. There is no way of saying the following in the shell without "`<>`":

126214 `cat food | more - >/dev/tty03 2<>/dev/tty03`

126215 Another example of "`<>`" is one that opens **/dev/tty** on file descriptor 3 for reading and writing:

126216 `exec 3<> /dev/tty`

126217 An example of creating a lock file for a critical code region:

126218 `set -C`  
126219 `until 2> /dev/null > lockfile`  
126220 `do sleep 30`  
126221 `done`  
126222 `set +C`  
126223 `perform critical function`  
126224 `rm lockfile`

126225 Since **/dev/null** is not a regular file, no error is generated by redirecting to it in *noclobber* mode.

126226 Tilde expansion is not performed on a here-document because the data is treated as if it were  
126227 enclosed in double-quotes.

#### 126228 C.2.7.1 *Redirecting Input*

126229 There is no additional rationale provided for this section.

#### 126230 C.2.7.2 *Redirecting Output*

126231 There is no additional rationale provided for this section.



126232 C.2.7.3 *Appending Redirected Output*

126233 Note that when a file is opened (even with the `O_APPEND` flag set), the initial file offset for that  
 126234 file is set to the beginning of the file. Some historic shells set the file offset to the current end-of-  
 126235 file when **append** mode shell redirection was used, but this is not allowed by POSIX.1-2008.

126236 C.2.7.4 *Here-Document*

126237 There is no additional rationale provided for this section.

126238 C.2.7.5 *Duplicating an Input File Descriptor*

126239 There is no additional rationale provided for this section.

126240 C.2.7.6 *Duplicating an Output File Descriptor*

126241 There is no additional rationale provided for this section.

126242 C.2.7.7 *Open File Descriptors for Reading and Writing*

126243 There is no additional rationale provided for this section.

126244 **C.2.8 Exit Status and Errors**

126245 There is no additional rationale provided for this section. +

126246 C.2.8.1 *Consequences of Shell Errors*

126247 There is no additional rationale provided for this section.

126248 C.2.8.2 *Exit Status for Commands*

126249 There is a historical difference in *sh* and *ksh* non-interactive error behavior. When a command  
 126250 named in a script is not found, some implementations of *sh* exit immediately, but *ksh* continues  
 126251 with the next command. Thus, the Shell and Utilities volume of POSIX.1-2008 says that the shell  
 126252 “may” exit in this case. This puts a small burden on the programmer, who has to test for  
 126253 successful completion following a command if it is important that the next command not be  
 126254 executed if the previous command was not found. If it is important for the command to have  
 126255 been found, it was probably also important for it to complete successfully. The test for successful  
 126256 completion would not need to change.

126257 Historically, shells have returned an exit status of  $128+n$ , where  $n$  represents the signal number.  
 126258 Since signal numbers are not standardized, there is no portable way to determine which signal  
 126259 caused the termination. Also, it is possible for a command to exit with a status in the same range  
 126260 of numbers that the shell would use to report that the command was terminated by a signal.  
 126261 Implementations are encouraged to choose exit values greater than 256 to indicate programs that  
 126262 terminate by a signal so that the exit status cannot be confused with an exit status generated by a  
 126263 normal termination.

126264 Historical shells make the distinction between “utility not found” and “utility found but cannot  
 126265 execute” in their error messages. By specifying two seldomly used exit status values for these  
 126266 cases, 127 and 126 respectively, this gives an application the opportunity to make use of this

distinction without having to parse an error message that would probably change from locale to locale. The *command*, *env*, *nohup*, and *xargs* utilities in the Shell and Utilities volume of POSIX.1-2008 have also been specified to use this convention.

When a command fails during word expansion or redirection, most historical implementations exit with a status of 1. However, there was some sentiment that this value should probably be much higher so that an application could distinguish this case from the more normal exit status values. Thus, the language “greater than zero” was selected to allow either method to be implemented.

## 126275 C.2.9 Shell Commands

A description of an “empty command” was removed from an early proposal because it is only relevant in the cases of *sh -c ""*, *system("")*, or an empty shell-script file (such as the implementation of *true* on some historical systems). Since it is no longer mentioned in the Shell and Utilities volume of POSIX.1-2008, it falls into the silently unspecified category of behavior where implementations can continue to operate as they have historically, but conforming applications do not construct empty commands. (However, note that *sh* does explicitly state an exit status for an empty string or file.) In an interactive session or a script with other commands, extra <newline> or <semicolon> characters, such as:

```
126284 $ false
126285 $
126286 $ echo $?
126287 1
```

would not qualify as the empty command described here because they would be consumed by other parts of the grammar.

### 126290 C.2.9.1 Simple Commands

The enumerated list is used only when the command is actually going to be executed. For example, in:

```
126293 true || $foo *
```

no expansions are performed.

The following example illustrates both how a variable assignment without a command name affects the current execution environment, and how an assignment with a command name only affects the execution environment of the command:

```
126298 $ x=red
126299 $ echo $x
126300 red
126301 $ export x
126302 $ sh -c 'echo $x'
126303 red
126304 $ x=blue sh -c 'echo $x'
126305 blue
126306 $ echo $x
126307 red
```

This next example illustrates that redirections without a command name are still performed:

```
126309 $ ls foo
```

```

126310 ls: foo: no such file or directory
126311 $ > foo
126312 $ ls foo
126313 foo

```

126314 A command without a command name, but one that includes a command substitution, has an  
 126315 exit status of the last command substitution that the shell performed. For example:

```

126316 if x=$(command)
126317 then ...
126318 fi

```

126319 An example of redirections without a command name being performed in a subshell shows that  
 126320 the here-document does not disrupt the standard input of the **while** loop:

```

126321 IFS=:
126322 while read a b
126323 do echo $a
126324 <<-eof
126325 Hello
126326 eof
126327 done </etc/passwd

```

126328 Following are examples of commands without command names in AND-OR lists:

```

126329 > foo || {
126330 echo "error: foo cannot be created" >&2
126331 exit 1
126332 }
126333 # set saved if /vmunix.save exists
126334 test -f /vmunix.save && saved=1

```

126335 Command substitution and redirections without command names both occur in subshells, but  
 126336 they are not necessarily the same ones. For example, in:

```

126337 exec 3> file
126338 var=$(echo foo >&3) 3>&1

```

126339 it is unspecified whether **foo** is echoed to the file or to standard output.

126340 POSIX.1-2008, Technical Corrigendum 1, XCU/TC1-2008/0021 [255] is applied. +

## 126341 Command Search and Execution

126342 This description requires that the shell can execute shell scripts directly, even if the underlying  
 126343 system does not support the common "#!" interpreter convention. That is, if file **foo** contains  
 126344 shell commands and is executable, the following executes **foo**:

```

126345 ./foo

```

126346 The command search shown here does not match all historical implementations. A more typical  
 126347 sequence has been:

- 126348 • Any built-in (special or regular)
- 126349 • Functions
- 126350 • Path search for executable files

126351 But there are problems with this sequence. Since the programmer has no idea in advance which  
 126352 utilities might have been built into the shell, a function cannot be used to override portably a

utility of the same name. (For example, a function named *cd* cannot be written for many historical systems.) Furthermore, the *PATH* variable is partially ineffective in this case, and only a pathname with a <slash> can be used to ensure a specific executable file is invoked.

After the *execve()* failure described, the shell normally executes the file as a shell script. Some implementations, however, attempt to detect whether the file is actually a script and not an executable from some other architecture. The method used by the KornShell is allowed by the text that indicates non-text files may be bypassed.

The sequence selected for the Shell and Utilities volume of POSIX.1-2008 acknowledges that special built-ins cannot be overridden, but gives the programmer full control over which versions of other utilities are executed. It provides a means of suppressing function lookup (via the *command* utility) for the user's own functions and ensures that any regular built-ins or functions provided by the implementation are under the control of the path search. The mechanisms for associating built-ins or functions with executable files in the path are not specified by the Shell and Utilities volume of POSIX.1-2008, but the wording requires that if either is implemented, the application is not able to distinguish a function or built-in from an executable (other than in terms of performance, presumably). The implementation ensures that all effects specified by the Shell and Utilities volume of POSIX.1-2008 resulting from the invocation of the regular built-in or function (interaction with the environment, variables, traps, and so on) are identical to those resulting from the invocation of an executable file.

IEEE Std 1003.1-2001/Cor 2-2004, item XCU/TC2/D6/4 is applied, updating the case where *execve()* fails due to an error equivalent to the [ENOEXEC] error.

POSIX.1-2008, Technical Corrigendum 1, XCU/TC1-2008/0022 [168], XCU/TC1-2008/0023 + [168], XCU/TC1-2008/0024 [168], XCU/TC1-2008/0025 [168], XCU/TC1-2008/0026 [168,430], + XCU/TC1-2008/0027 [168,430], and XCU/TC1-2008/0028 [173] are applied.

## Examples

Consider three versions of the *ls* utility:

1. The application includes a shell function named *ls*.
2. The user writes a utility named *ls* and puts it in **/fred/bin**.
3. The example implementation provides *ls* as a regular shell built-in that is invoked (either by the shell or directly by *exec*) when the path search reaches the directory **/posix/bin**.

If *PATH*=**/posix/bin**, various invocations yield different versions of *ls*:

| Invocation                                             | Version of <i>ls</i> |
|--------------------------------------------------------|----------------------|
| <i>ls</i> (from within application script)             | (1) function         |
| <i>command ls</i> (from within application script)     | (3) built-in         |
| <i>ls</i> (from within makefile called by application) | (3) built-in         |
| <i>system("ls")</i>                                    | (3) built-in         |
| <i>PATH="/fred/bin:\$PATH" ls</i>                      | (2) user's version   |

### C.2.9.2 Pipelines

Because pipeline assignment of standard input or standard output or both takes place before redirection, it can be modified by redirection. For example:

```
$ command1 2>&1 | command2
```

sends both the standard output and standard error of *command1* to the standard input of *command2*.

|        |                                                                                                                                                         |   |
|--------|---------------------------------------------------------------------------------------------------------------------------------------------------------|---|
| 126396 | The reserved word <code>!</code> allows more flexible testing using AND and OR lists.                                                                   |   |
| 126397 | It was suggested that it would be better to return a non-zero value if any command in the                                                               |   |
| 126398 | pipeline terminates with non-zero status (perhaps the bitwise-inclusive OR of all return values).                                                       |   |
| 126399 | However, the choice of the last-specified command semantics are historical practice and would                                                           |   |
| 126400 | cause applications to break if changed. An example of historical behavior:                                                                              |   |
| 126401 | <code>\$ sleep 5   (exit 4)</code>                                                                                                                      |   |
| 126402 | <code>\$ echo \$?</code>                                                                                                                                |   |
| 126403 | <code>4</code>                                                                                                                                          |   |
| 126404 | <code>\$ (exit 4)   sleep 5</code>                                                                                                                      |   |
| 126405 | <code>\$ echo \$?</code>                                                                                                                                |   |
| 126406 | <code>0</code>                                                                                                                                          |   |
| 126407 | POSIX.1-2008, Technical Corrigendum 1, XCU/TC1-2008/0029 [205] is applied.                                                                              | + |
| 126408 | <b>Exit Status</b>                                                                                                                                      | + |
| 126409 | POSIX.1-2008, Technical Corrigendum 1, XCU/TC1-2008/0030 [52] is applied.                                                                               |   |
| 126410 | <b>C.2.9.3 Lists</b>                                                                                                                                    |   |
| 126411 | The equal precedence of " <code>&amp;&amp;</code> " and " <code>  </code> " is historical practice. The standard developers                             |   |
| 126412 | evaluated the model used more frequently in high-level programming languages, such as C, to                                                             |   |
| 126413 | allow the shell logical operators to be used for complex expressions in an unambiguous way, but                                                         |   |
| 126414 | they could not allow historical scripts to break in the subtle way unequal precedence might                                                             |   |
| 126415 | cause. Some arguments were posed concerning the " <code>{ }</code> " or " <code>()</code> " groupings that are required                                 |   |
| 126416 | historically. There are some disadvantages to these groupings:                                                                                          |   |
| 126417 | <ul style="list-style-type: none"> <li>• The "<code>()</code>" can be expensive, as they spawn other processes on some implementations. This</li> </ul> |   |
| 126418 | performance concern is primarily an implementation issue.                                                                                               |   |
| 126419 | <ul style="list-style-type: none"> <li>• The "<code>{ }</code>" braces are not operators (they are reserved words) and require a trailing</li> </ul>    |   |
| 126420 | <space> after each ' <code>{</code> ', and a <semicolon> before each ' <code>}</code> '. Most programmers (and                                          |   |
| 126421 | certainly interactive users) have avoided braces as grouping constructs because of the                                                                  |   |
| 126422 | problematic syntax required. Braces were not changed to operators because that would                                                                    |   |
| 126423 | generate compatibility issues even greater than the precedence question; braces appear                                                                  |   |
| 126424 | outside the context of a keyword in many shell scripts.                                                                                                 |   |
| 126425 | IEEE PASC Interpretation 1003.2 #204 is applied, clarifying that the operators " <code>&amp;&amp;</code> " and " <code>  </code> "                      |   |
| 126426 | are evaluated with left associativity.                                                                                                                  |   |
| 126427 | POSIX.1-2008, Technical Corrigendum 1, XCU/TC1-2008/0031 [45] and XCU/TC1-2008/0032                                                                     | + |
| 126428 | [45] are applied.                                                                                                                                       |   |
| 126429 | <b>Asynchronous Lists</b>                                                                                                                               |   |
| 126430 | The grammar treats a construct such as:                                                                                                                 |   |
| 126431 | <code>foo &amp; bar &amp; bam &amp;</code>                                                                                                              |   |
| 126432 | as one "asynchronous list", but since the status of each element is tracked by the shell, the term                                                      |   |
| 126433 | "element of an asynchronous list" was introduced to identify just one of the <b>foo</b> , <b>bar</b> , or <b>bam</b>                                    |   |
| 126434 | portions of the overall list.                                                                                                                           |   |
| 126435 | Unless the implementation has an internal limit, such as <code>{CHILD_MAX}</code> , on the retained process                                             |   |
| 126436 | IDs, it would require unbounded memory for the following example:                                                                                       |   |
| 126437 | <code>while true</code>                                                                                                                                 |   |

126438       do       foo & echo \$!  
126439       done

126440       The treatment of the signals SIGINT and SIGQUIT with asynchronous lists is described in XCU  
126441       [Section 2.11](#) (on page 2353).

126442       Since the connection of the input to the equivalent of **/dev/null** is considered to occur before  
126443       redirections, the following script would produce no output:

126444       exec < /etc/passwd  
126445       cat <&0 &  
126446       wait

#### 126447       **Sequential Lists**

126448       There is no additional rationale provided for this section.

#### 126449       **AND Lists**

126450       There is no additional rationale provided for this section.

#### 126451       **OR Lists**

126452       There is no additional rationale provided for this section.

### 126453   C.2.9.4   *Compound Commands*

#### 126454       **Grouping Commands**

126455       The semicolon shown in *{compound-list;}* is an example of a control operator delimiting the }  
126456       reserved word. Other delimiters are possible, as shown in XCU [Section 2.10](#) (on page 2347);  
126457       <newline> is frequently used.

126458       A proposal was made to use the **<do-done>** construct in all cases where command grouping in  
126459       the current process environment is performed, identifying it as a construct for the grouping  
126460       commands, as well as for shell functions. This was not included because the shell already has a  
126461       grouping construct for this purpose ("**{ }**"), and changing it would have been counter-  
126462       productive.

126463       The requirement for conforming applications to separate two leading ' ( ' characters with white +  
126464       space if a grouping command would be parsed as an arithmetic expansion if preceded by a ' \$' +  
126465       is to allow shells which implement the "**(( arithmetic expression ))**" extension to +  
126466       apply the same disambiguation rules consistently to **\$((...))** and **((...))**. See [Section](#) +  
126467       [C.2.6.3](#) (on page 3684). +

126468       POSIX.1-2008, Technical Corrigendum 1, XCU/TC1-2008/0033 [217] is applied.

#### 126469       **For Loop**

126470       The format is shown with generous usage of <newline> characters. See the grammar in XCU  
126471       [Section 2.10](#) (on page 2347) for a precise description of where <newline> and <semicolon>  
126472       characters can be interchanged.

126473       Some historical implementations support '**{**' and '**}**' as substitutes for **do** and **done**. The  
126474       standard developers chose to omit them, even as an obsolescent feature. (Note that these  
126475       substitutes were only for the **for** command; the **while** and **until** commands could not use them  
126476       historically because they are followed by compound-lists that may contain "**{ . . . }**" grouping  
126477       commands themselves.)

126478 The reserved word pair **do ... done** was selected rather than **do ... od** (which would have  
 126479 matched the spirit of **if ... fi** and **case ... esac**) because *od* is already the name of a standard  
 126480 utility.

126481 PASC Interpretation 1003.2 #169 has been applied changing the grammar.

#### 126482 **Case Conditional Construct**

126483 An optional <left-parenthesis> before *pattern* was added to allow numerous historical KornShell  
 126484 scripts to conform. At one time, using the leading parenthesis was required if the **case** statement  
 126485 was to be embedded within a "\$ ( ) " command substitution; this is no longer the case with the  
 126486 POSIX shell. Nevertheless, many historical scripts use the <left-parenthesis>, if only because it  
 126487 makes matching-parenthesis searching easier in *vi* and other editors. This is a relatively simple  
 126488 implementation change that is upwards-compatible for all scripts.

126489 Consideration was given to requiring *break* inside the *compound-list* to prevent falling through to  
 126490 the next pattern action list. This was rejected as being nonexistent practice. An interesting  
 126491 undocumented feature of the KornShell is that using ";" &" instead of ";;" as a terminator  
 126492 causes the exact opposite behavior—the flow of control continues with the next *compound-list*.

126493 The pattern ' \* ', given as the last pattern in a **case** construct, is equivalent to the default case in  
 126494 a C-language **switch** statement.

126495 The grammar shows that reserved words can be used as patterns, even if one is the first word on  
 126496 a line. Obviously, the reserved word **esac** cannot be used in this manner.

#### 126497 **If Conditional Construct**

126498 The precise format for the command syntax is described in XCU [Section 2.10](#) (on page 2347).

#### 126499 **While Loop**

126500 The precise format for the command syntax is described in XCU [Section 2.10](#) (on page 2347).

#### 126501 **Until Loop**

126502 The precise format for the command syntax is described in XCU [Section 2.10](#) (on page 2347).

#### 126503 C.2.9.5 *Function Definition Command*

126504 The description of functions in an early proposal was based on the notion that functions should  
 126505 behave like miniature shell scripts; that is, except for sharing variables, most elements of an  
 126506 execution environment should behave as if they were a new execution environment, and  
 126507 changes to these should be local to the function. For example, traps and options should be reset  
 126508 on entry to the function, and any changes to them do not affect the traps or options of the caller.  
 126509 There were numerous objections to this basic idea, and the opponents asserted that functions  
 126510 were intended to be a convenient mechanism for grouping common commands that were to be  
 126511 executed in the current execution environment, similar to the execution of the *dot* special  
 126512 built-in.

126513 It was also pointed out that the functions described in that early proposal did not provide a local  
 126514 scope for everything a new shell script would, such as the current working directory, or *umask*,  
 126515 but instead provided a local scope for only a few select properties. The basic argument was that  
 126516 if a local scope is needed for the execution environment, the mechanism already existed: the  
 126517 application can put the commands in a new shell script and call that script. All historical shells  
 126518 that implemented functions, other than the KornShell, have implemented functions that operate  
 126519 in the current execution environment. Because of this, traps and options have a global scope

within a shell script. Local variables within a function were considered and included in another early proposal (controlled by the special built-in *local*), but were removed because they do not fit the simple model developed for functions and because there was some opposition to adding yet another new special built-in that was not part of historical practice. Implementations should reserve the identifier *local* (as well as *typeset*, as used in the KornShell) in case this local variable mechanism is adopted in a future version of this standard.

A separate issue from the execution environment of a function is the availability of that function to child shells. A few objectors maintained that just as a variable can be shared with child shells by exporting it, so should a function. In early proposals, the *export* command therefore had a *-f* flag for exporting functions. Functions that were exported were to be put into the environment as *name()*=*value* pairs, and upon invocation, the shell would scan the environment for these and automatically define these functions. This facility was strongly opposed and was omitted. Some of the arguments against exportable functions were as follows:

- There was little historical practice. The Ninth Edition shell provided them, but there was controversy over how well it worked.
- There are numerous security problems associated with functions appearing in the environment of a user and overriding standard utilities or the utilities owned by the application.
- There was controversy over requiring *make* to import functions, where it has historically used an *exec* function for many of its command line executions.
- Functions can be big and the environment is of a limited size. (The counter-argument was that functions are no different from variables in terms of size: there can be big ones, and there can be small ones—and just as one does not export huge variables, one does not export huge functions. However, this might not apply to the average shell-function writer, who typically writes much larger functions than variables.)

As far as can be determined, the functions in the Shell and Utilities volume of POSIX.1-2008 match those in System V. Earlier versions of the KornShell had two methods of defining functions:

```
function fname { compound-list }
```

and:

```
fname() { compound-list }
```

The latter used the same definition as the Shell and Utilities volume of POSIX.1-2008, but differed in semantics, as described previously. The current edition of the KornShell aligns the latter syntax with the Shell and Utilities volume of POSIX.1-2008 and keeps the former as is.

The name space for functions is limited to that of a *name* because of historical practice. Complications in defining the syntactic rules for the function definition command and in dealing with known extensions such as the "*@()*" usage in the KornShell prevented the name space from being widened to a *word*. Using functions to support synonyms such as the "*!!*" and "*%'*" usage in the C shell is thus disallowed to conforming applications, but acceptable as an extension. For interactive users, the aliasing facilities in the Shell and Utilities volume of POSIX.1-2008 should be adequate for this purpose. It is recognized that the name space for utilities in the file system is wider than that currently supported for functions, if the portable filename character set guidelines are ignored, but it did not seem useful to mandate extensions in systems for so little benefit to conforming applications.

The "*()*" in the function definition command consists of two operators. Therefore, intermixing <blank> characters with the *fname*, '*'*', and '*)*' is allowed, but unnecessary.



126566 An example of how a function definition can be used wherever a simple command is allowed:

```
126567 # If variable i is equal to "yes",
126568 # define function foo to be ls -l
126569 #
126570 ["$i" = yes] && foo() {
126571 ls -l
126572 }
```

126573 POSIX.1-2008, Technical Corrigendum 1, XCU/TC1-2008/0034 [383] and XCU/TC1-2008/0035 +  
126574 [214] are applied.

## 126575 C.2.10 Shell Grammar

126576 There are several subtle aspects of this grammar where conventional usage implies rules about  
126577 the grammar that in fact are not true.

126578 For *compound\_list*, only the forms that end in a *separator* allow a reserved word to be recognized,  
126579 so usually only a *separator* can be used where a compound list precedes a reserved word (such as  
126580 **Then**, **Else**, **Do**, and **Rbrace**). Explicitly requiring a separator would disallow such valid (if rare)  
126581 statements as:

```
126582 if (false) then (echo x) else (echo y) fi
```

126583 See the Note under special grammar rule (1).

126584 Concerning the third sentence of rule (1) (“Also, if the parser ...”):

- 126585 • This sentence applies rather narrowly: when a compound list is terminated by some clear  
126586 delimiter (such as the closing **fi** of an inner **if\_clause**) then it would apply; where the  
126587 compound list might continue (as in after a ‘;’), rule (7a) (and consequently the first  
126588 sentence of rule (1)) would apply. In many instances the two conditions are identical, but  
126589 this part of rule (1) does not give license to treating a **WORD** as a reserved word unless it  
126590 is in a place where a reserved word has to appear.
- 126591 • The statement is equivalent to requiring that when the LR(1) lookahead set contains  
126592 exactly one reserved word, it must be recognized if it is present. (Here “LR(1)” refers to the  
126593 theoretical concepts, not to any real parser generator.)

126594 For example, in the construct below, and when the parser is at the point marked with ‘^’,  
126595 the only next legal token is **then** (this follows directly from the grammar rules):

```
126596 if if...fi then ... fi
126597 ^
```

126598 At that point, the **then** must be recognized as a reserved word.

126599 (Depending on the parser generator actually used, “extra” reserved words may be in some  
126600 lookahead sets. It does not really matter if they are recognized, or even if any possible  
126601 reserved word is recognized in that state, because if it is recognized and is not in the  
126602 (theoretical) LR(1) lookahead set, an error is ultimately detected. In the example above, if  
126603 some other reserved word (for example, **while**) is also recognized, an error occurs later.

126604 This is approximately equivalent to saying that reserved words are recognized after other  
126605 reserved words (because it is after a reserved word that this condition occurs), but avoids  
126606 the “except for ...” list that would be required for **case**, **for**, and so on. (Reserved words  
126607 are of course recognized anywhere a *simple\_command* can appear, as well. Other rules take  
126608 care of the special cases of non-recognition, such as rule (4) for **case** statements.)

126609 Note that the body of here-documents are handled by token recognition (see XCU [Section 2.3](#), on  
 126610 page 2321) and do not appear in the grammar directly. (However, the here-document I/O  
 126611 redirection operator is handled as part of the grammar.)

126612 The start symbol of the grammar (**complete\_command**) represents either input from the  
 126613 command line or a shell script. It is repeatedly applied by the interpreter to its input and  
 126614 represents a single “chunk” of that input as seen by the interpreter.

#### 126615 C.2.10.1 Shell Grammar Lexical Conventions

126616 There is no additional rationale provided for this section.

#### 126617 C.2.10.2 Shell Grammar Rules

126618 POSIX.1-2008, Technical Corrigendum 1, XCU/TC1-2008/0036 [44] is applied. |

### 126619 C.2.11 Signals and Error Handling

126620 SD5-XCU-ERN-93 is applied, updating the first paragraph of XCU [Section 2.11](#) (on page 2353).

### 126621 C.2.12 Shell Execution Environment

126622 Some implementations have implemented the last stage of a pipeline in the current environment  
 126623 so that commands such as:

126624 `command | read foo`

126625 set variable **foo** in the current environment. This extension is allowed, but not required;  
 126626 therefore, a shell programmer should consider a pipeline to be in a subshell environment, but  
 126627 not depend on it.

126628 In early proposals, the description of execution environment failed to mention that each  
 126629 command in a multiple command pipeline could be in a subshell execution environment. For  
 126630 compatibility with some historical shells, the wording was phrased to allow an implementation  
 126631 to place any or all commands of a pipeline in the current environment. However, this means that  
 126632 a POSIX application must assume each command is in a subshell environment, but not depend  
 126633 on it.

126634 The wording about shell scripts is meant to convey the fact that describing “trap actions” can  
 126635 only be understood in the context of the shell command language. Outside of this context, such  
 126636 as in a C-language program, signals are the operative condition, not traps.

126637 POSIX.1-2008, Technical Corrigendum 1, XCU/TC1-2008/0037 [238] is applied. +

### 126638 C.2.13 Pattern Matching Notation

126639 Pattern matching is a simpler concept and has a simpler syntax than REs, as the former is  
 126640 generally used for the manipulation of filenames, which are relatively simple collections of  
 126641 characters, while the latter is generally used to manipulate arbitrary text strings of potentially  
 126642 greater complexity. However, some of the basic concepts are the same, so this section points  
 126643 liberally to the detailed descriptions in XBD [Chapter 9](#) (on page 181).

126644 C.2.13.1 *Patterns Matching a Single Character*

126645 Both quoting and escaping are described here because pattern matching must work in three  
126646 separate circumstances:

- 126647 1. Calling directly upon the shell, such as in pathname expansion or in a **case** statement. All  
126648 of the following match the string or file **abc**:

126649 `abc "abc" a"b" c a\bc a[b]c a["b"]c a[\b]c a["\b"]c a?c a*c`

126650 The following do not:

126651 `"a?c" a*c a\[b]c`

- 126652 2. Calling a utility or function without going through a shell, as described for *find* and the  
126653 *fnmatch()* function defined in the System Interfaces volume of POSIX.1-2008.

- 126654 3. Calling utilities such as *find*, *cpio*, *tar*, or *pax* through the shell command line. In this case,  
126655 shell quote removal is performed before the utility sees the argument. For example, in:

126656 `find /bin -name "e\c[\h]o" -print`

126657 after quote removal, the <backslash> characters are presented to *find* and it treats them as  
126658 escape characters. Both precede ordinary characters, so the *c* and *h* represent themselves  
126659 and *echo* would be found on many historical systems (that have it in **/bin**). To find a  
126660 filename that contained shell special characters or pattern characters, both quoting and  
126661 escaping are required, such as:

126662 `pax -r ... "*a\(\?"`

126663 to extract a filename ending with "a(?".

126664 Conforming applications are required to quote or escape the shell special characters (sometimes  
126665 called metacharacters). If used without this protection, syntax errors can result or  
126666 implementation extensions can be triggered. For example, the KornShell supports a series of  
126667 extensions based on parentheses in patterns.

126668 The restriction on a <circumflex> in a bracket expression is to allow implementations that  
126669 support pattern matching using the <circumflex> as the negation character in addition to the  
126670 <exclamation-mark>. A conforming application must use something like "[\^!]" to match  
126671 either character.

126672 C.2.13.2 *Patterns Matching Multiple Characters*

126673 Since each <asterisk> matches zero or more occurrences, the patterns "a\*b" and "a\*\*b" have  
126674 identical functionality.

126675 **Examples**

126676 `a[bc]` Matches the strings "ab" and "ac".

126677 `a*d` Matches the strings "ad", "abd", and "abcd", but not the string "abc".

126678 `a*d*` Matches the strings "ad", "abcd", "abcdef", "aaaad", and "adddd".

126679 `*a*d` Matches the strings "ad", "abcd", "efabcd", "aaaad", and "adddd".

126680 C.2.13.3 *Patterns Used for Filename Expansion*

126681 The caveat about a <slash> within a bracket expression is derived from historical practice. The  
 126682 pattern "a[b/c]d" does not match such pathnames as **abd** or **a/d**. On some implementations  
 126683 (including those conforming to the Single UNIX Specification), it matched a pathname of  
 126684 literally "a[b/c]d". On other systems, it produced an undefined condition (an unescaped '['  
 126685 used outside a bracket expression). In this version, the XSI behavior is now required.

126686 Filenames beginning with a <period> historically have been specially protected from view on  
 126687 UNIX systems. A proposal to allow an explicit <period> in a bracket expression to match a  
 126688 leading <period> was considered; it is allowed as an implementation extension, but a  
 126689 conforming application cannot make use of it. If this extension becomes popular in the future, it  
 126690 will be considered for a future version of the Shell and Utilities volume of POSIX.1-2008.

126691 Historical systems have varied in their permissions requirements. To match **f\*/bar** has required  
 126692 read permissions on the **f\*** directories in the System V shell, but the Shell and Utilities volume of  
 126693 POSIX.1-2008, the C shell, and KornShell require only search permissions.

126694 **C.2.14 Special Built-In Utilities**

126695 See the RATIONALE sections on the individual reference pages.

126696 **C.3 Batch Environment Services and Utilities**126697 **Scope of the Batch Environment Services and Utilities Option**

126698 This section summarizes the deliberations of the IEEE P1003.15 (Batch Environment) working  
 126699 group in the development of the Batch Environment Services and Utilities option, which covers  
 126700 a set of services and utilities defining a batch processing system.

126701 This informative section contains historical information concerning the contents of the  
 126702 amendment and describes why features were included or discarded by the working group.

126703 **History of Batch Systems**

126704 The supercomputing technical committee began as a "Birds Of a Feather" (BOF) at the January  
 126705 1987 Usenix meeting. There was enough general interest to form a supercomputing attachment  
 126706 to the /usr/group working groups. Several subgroups rapidly formed. Of those subgroups, the  
 126707 batch group was the most ambitious. The first early meetings were spent evaluating user needs  
 126708 and existing batch implementations.

126709 To evaluate user needs, individuals from the supercomputing community came and presented  
 126710 their needs. Common requests were flexibility, interoperability, control of resources, and ease-of-  
 126711 use. Backward-compatibility was not an issue. The working group then evaluated some existing  
 126712 systems. The following different systems were evaluated:

- 126713 • PROD
- 126714 • Convex Distributed Batch
- 126715 • NQS
- 126716 • CTSS
- 126717 • MDQS from Ballistics Research Laboratory (BRL)

126718 Finally, NQS was chosen as a model because it satisfied not only the most user requirements, but

126719 because it was public domain, already implemented on a variety of hardware platforms, and  
126720 network-based.

### 126721 **Historical Implementations of Batch Systems**

126722 Deferred processing of work under the control of a scheduler has been a feature of most  
126723 proprietary operating systems from the earliest days of multi-user systems in order to maximize  
126724 utilization of the computer.

126725 The arrival of UNIX systems proved to be a dilemma to many hardware providers and users  
126726 because it did not include the sophisticated batch facilities offered by the proprietary systems.  
126727 This omission was rectified in 1986 by NASA Ames Research Center who developed the  
126728 Network Queuing System (NQS) as a portable UNIX application that allowed the routing and  
126729 processing of batch “jobs” in a network. To encourage its usage, the product was later put into  
126730 the public domain. It was promptly picked up by UNIX hardware providers, and ported and  
126731 developed for their respective hardware and UNIX implementations.

126732 Many major vendors, who traditionally offer a batch-dominated environment, ported the  
126733 public-domain product to their systems, customized it to support the capabilities of their  
126734 systems, and added many customer-requested features.

126735 Due to the strong hardware provider and customer acceptance of NQS, it was decided to use  
126736 NQS as the basis for the POSIX Batch Environment amendment in 1987. Other batch systems  
126737 considered at the time included CTSS, MDQS (a forerunner of NQS from the Ballistics Research  
126738 Laboratory), and PROD (a Los Alamos Labs development). None were thought to have both the  
126739 functionality and acceptability of NQS.

### 126740 **NQS Differences from the *at* utility**

126741 The base standard *at* and *batch* utilities are not sufficient to meet the batch processing needs in a  
126742 supercomputing environment and additional functionality in the areas of resource management,  
126743 job scheduling, system management, and control of output is required.

### 126744 **Batch Environment Services and Utilities Option Definitions**

126745 The concept of a batch job is closely related to a session with a session leader. The main  
126746 difference is that a batch job does not have a controlling terminal. There has been much debate  
126747 over whether to use the term “request” or “job”. Job was the final choice because of the  
126748 historical use of this term in the batch environment.

126749 The current definition for job identifiers is not sufficient with the model of destinations. The  
126750 current definition is:

126751 `sequence_number.originating_host`

126752 Using the model of destination, a host may include multiple batch nodes, the location of which  
126753 is identified uniquely by a name or directory service. If the current definition is used, batch  
126754 nodes running on the same host would have to coordinate their use of sequence numbers, as  
126755 sequence numbers are assigned by the originating host. The alternative is to use the originating  
126756 batch node name instead of the originating host name.

126757 The reasons for wishing to run more than one batch system per host could be the following.

126758 A test and production batch system are maintained on a single host. This is most likely in a  
126759 development facility, but could also arise when a site is moving from one version to another. The  
126760 new batch system could be installed as a test version that is completely separate from the  
126761 production batch system, so that problems can be isolated to the test system. Requiring the batch  
126762 nodes to coordinate their use of sequence numbers creates a dependency between the two

- 126763 nodes, and that defeats the purpose of running two nodes.
- 126764 A site has multiple departments using a single host, with different management policies. An  
126765 example of contention might be in job selection algorithms. One group might want a FIFO type  
126766 of selection, while another group wishes to use a more complex algorithm based on resource  
126767 availability. Again, requiring the batch nodes to coordinate is an unnecessary binding.
- 126768 The proposal eventually accepted was to replace originating host with originating batch node.  
126769 This supplies sufficient granularity to ensure unique job identifiers. If more than one batch node  
126770 is on a particular host, they each have their own unique name.
- 126771 The queue portion of a destination is not part of the job identifier as these are not required to be  
126772 unique between batch nodes. For instance, two batch nodes may both have queues called small,  
126773 medium, and large. It is only the batch node name that is uniquely identifiable throughout the  
126774 batch system. The queue name has no additional function in this context.
- 126775 Assume there are three batch nodes, each of which has its own name server. On batch node one,  
126776 there are no queues. On batch node two, there are fifty queues. On batch node three, there are  
126777 forty queues. The system administrator for batch node one does not have to configure queues,  
126778 because there are none implemented. However, if a user wishes to send a job to either batch  
126779 node two or three, the system administrator for batch node one must configure a destination  
126780 that maps to the appropriate batch node and queue. If every queue is to be made accessible from  
126781 batch node one, the system administrator has to configure ninety destinations.
- 126782 To avoid requiring this, there should be a mechanism to allow a user to separate the destination  
126783 into a batch node name and a queue name. Then, an implementation that is configured to get to  
126784 all the batch nodes does not need any more configuration to allow a user to get to all of the  
126785 queues on all of the batch nodes. The node name is used to locate the batch node, while the  
126786 queue name is sent unchanged to that batch node.
- 126787 The following are requirements that a destination identifier must be capable of providing:
- 126788 • The ability to direct a job to a queue in a particular batch node.
  - 126789 • The ability to direct a job to a particular batch node.
  - 126790 • The ability to group at a higher level than just one queue. This includes grouping similar  
126791 queues across multiple batch nodes (this is a pipe queue).
  - 126792 • The ability to group batch nodes. This allows a user to submit a job to a group name with  
126793 no knowledge of the batch node configuration. This also provides aliasing as a special  
126794 case. Aliasing is a group containing only one batch node name. The group name is the  
126795 alias.
- 126796 In addition, the administrator has the following requirements:
- 126797 • The ability to control access to the queues.
  - 126798 • The ability to control access to the batch nodes.
  - 126799 • The ability to control access to groups of queues (pipe queues).
  - 126800 • The ability to configure retry time intervals and durations.
- 126801 The requirements of the user are met by destination as explained in the following.
- 126802 The user has the ability to specify a queue name, which is known only to the batch node  
126803 specified. There is no configuration of these queues required on the submitting node.
- 126804 The user has the ability to specify a batch node whose name is network-unique. The  
126805 configuration required is that the batch node be defined as an application, just as other  
126806 applications such as FTP are configured.

126807 Once a job reaches a queue, it can again become a user of the batch system. The batch node can  
 126808 choose to send the job to another batch node or queue or both. In other words, the routing is at  
 126809 an application level, and it is up to the batch system to choose where the job will be sent.  
 126810 Configuration is up to the batch node where the queue resides. This provides grouping of  
 126811 queues across batch nodes or within a batch node. The user submits the job to a queue, which by  
 126812 definition routes the job to other queues or nodes or both.

126813 A node name may be given to a naming service, which returns multiple addresses as opposed to  
 126814 just one. This provides grouping at a batch node level. This is a local issue, meaning that the  
 126815 batch node must choose only one of these addresses. The list of addresses is not sent with the  
 126816 job, and once the job is accepted on another node, there is no connection between the list and the  
 126817 job. The requirements of the administrator are met by destination as explained in the following.

126818 The control of queues is a batch system issue, and will be done using the batch administrative  
 126819 utilities.

126820 The control of nodes is a network issue, and will be done through whatever network facilities  
 126821 are available.

126822 The control of access to groups of queues (pipe queues) is covered by the control of any other  
 126823 queue. The fact that the job may then be sent to another destination is not relevant.

126824 The propagation of a job across more than one point-to-point connection was dropped because  
 126825 of its complexity and because all of the issues arising from this capability could not be resolved.  
 126826 It could be provided as additional functionality at some time in the future.

126827 The addition of *network* as a defined term was done to clarify the difference between a network  
 126828 of batch nodes as opposed to a network of hosts. A network of batch nodes is referred to as a  
 126829 batch system. The network refers to the actual host configuration. A single host may have  
 126830 multiple batch nodes.

126831 In the absence of a standard network naming convention, this option establishes its own  
 126832 convention for the sake of consistency and expediency. This is subject to change, should a future  
 126833 working group develop a standard naming convention for network pathnames.

### 126834 C.3.1 Batch General Concepts

126835 During the development of the Batch Environment Services and Utilities option, a number of  
 126836 topics were discussed at length which influenced the wording of the normative text but could  
 126837 not be included in the final text. The following items are some of the most significant terms and  
 126838 concepts of those discussed:

- 126839 • Small and Consistent Command Set

126840 Often, conventional utilities from UNIX systems have a very complicated utility syntax  
 126841 and usage. This can often result in confusion and errors when trying to use them. The  
 126842 Batch Environment Services and Utilities option utility set, on the other hand, has been  
 126843 paired to a small set of robust utilities with an orthogonal calling sequence.

- 126844 • Checkpoint/Restart

126845 This feature permits an already executing process to checkpoint or save its contents. Some  
 126846 implementations permit this at both the batch utility level (for example, checkpointing this  
 126847 job upon its abnormal termination) or from within the job itself via a system call. Support  
 126848 of checkpoint/restart is optional. A conscious, careful effort was made to make the *qsub*  
 126849 utility consistently refer to checkpoint/restart as optional functionality.

- 126850
- Rerunability
- 126851 When a user submits a job for batch processing, they can designate it “rerunnable” in that  
126852 it will automatically resume execution from the start of the job if the machine on which it  
126853 was executing crashes for some reason. The decision on whether the job will be rerun or  
126854 not is entirely up to the submitter of the job and no decisions will be made within the batch  
126855 system. A job that is rerunnable and has been submitted with the proper  
126856 checkpoint/restart switch will first be checkpointed and execution begun from that point.  
126857 Furthermore, use of the implementation-defined checkpoint/restart feature will not be  
126858 defined in this context.
- 126859
- Error Codes
- 126860 All utilities exit with error status zero (0) if successful, one (1) if a user error occurred, and  
126861 two (2) for an internal Batch Environment Services and Utilities option error.
- 126862
- Level of Portability
- 126863 Portability is specified at both the user, operator, and administrator levels. A conforming  
126864 batch implementation prevents identical functionality and behavior at all these levels.  
126865 Additionally, portable batch shell scripts with embedded Batch Environment Services and  
126866 Utilities option utilities add an additional level of portability.
- 126867
- Resource Specification
- 126868 A small set of globally understood resources, such as memory and CPU time, is specified.  
126869 All conforming batch implementations are able to process them in a manner consistent  
126870 with the yet-to-be-developed resource management model. Resources not in this  
126871 amendment set are ignored and passed along as part of the argument stream of the utility.
- 126872
- Queue Position
- 126873 Queue position is the place a job occupies in a queue. It is dependent on a variety of factors  
126874 such as submission time and priority. Since priority may be affected by the implementation  
126875 of fair share scheduling, the definition of queue position is implementation-defined.
- 126876
- Queue ID
- 126877 A numerical queue ID is an external requirement for purposes of accounting. The  
126878 identification number was chosen over queue name for processing convenience.
- 126879
- Job ID
- 126880 A common notion of “jobs” is a collection of processes whose process group cannot be  
126881 altered and is used for resource management and accounting. This concept is  
126882 implementation-defined and, as such, has been omitted from the batch amendment.
- 126883
- Bytes *versus* Words
- 126884 Except for one case, bytes are used as the standard unit for memory size. Furthermore, the  
126885 definition of a word varies from machine to machine. Therefore, bytes will be the default  
126886 unit of memory size.
- 126887
- Regular Expressions
- 126888 The standard definition of regular expressions is much too broad to be used in the batch  
126889 utility syntax. All that is needed is a simple concept of “all”; for example, delete all my jobs  
126890 from the named queue. For this reason, regular expressions have been eliminated from the  
126891 batch amendment.



- 126892 • Display Privacy
  - 126893 How much data should be displayed locally through functions? Local policy dictates the
  - 126894 amount of privacy. Library functions must be used to create and enforce local policy.
  - 126895 Network and local *qstats* must reflect the policy of the server machine.
- 126896 • Remote Host Naming Convention
  - 126897 It was decided that host names would be a maximum of 255 characters in length, with at
  - 126898 most 15 characters being shown in displays. The 255 character limit was chosen because it
  - 126899 is consistent with BSD. The 15-character limit was an arbitrary decision.
- 126900 • Network Administration
  - 126901 Network administration is important, but is outside the scope of the batch amendment.
  - 126902 Network administration could be done with *rsh*. However, authentication becomes two-
  - 126903 sided.
- 126904 • Network Administration Philosophy
  - 126905 Keep it simple. Centralized management should be possible. For example, Los Alamos
  - 126906 needs a dumb set of CPUs to be managed by a central system *versus* several
  - 126907 independently-managed systems as is the general case for the Batch Environment Services
  - 126908 and Utilities option.
- 126909 • Operator Utility Defaults (that is, Default Host, User, Account, and so on)
  - 126910 It was decided that usability would override orthogonality and syntactic consistency.
- 126911 • The Batch System Manager and Operator Distinction
  - 126912 The distinction between manager and operator is that operators can only control the flow
  - 126913 of jobs. A manager can alter the batch system configuration in addition to job flow. POSIX
  - 126914 makes a distinction between user and system administrator but goes no further. The
  - 126915 concepts of manager and operator privileges fall under local policy. The distinction
  - 126916 between manager and operator is historical in batch environments, and the Batch
  - 126917 Environment Services and Utilities option has continued that distinction.
- 126918 • The Batch System Administrator
  - 126919 An administrator is equivalent to a batch system manager.

### 126920 C.3.2 Batch Services

- 126921 This rationale is provided as informative rather than normative text, to avoid placing
- 126922 requirements on implementors regarding the use of symbolic constants, but at the same time to
- 126923 give implementors a preferred practice for assigning values to these constants to promote
- 126924 interoperability.
- 126925 The *Checkpoint* and *Minimum\_Cpu\_Interval* attributes induce a variety of behavior depending
- 126926 upon their values. Some jobs cannot or should not be checkpointed. Other users will simply
- 126927 need to ensure job continuation across planned downtimes; for example, scheduled preventive
- 126928 maintenance. For users consuming expensive resources, or for jobs that run longer than the
- 126929 mean time between failures, however, periodic checkpointing may be essential. However,
- 126930 system administrators must be able to set minimum checkpoint intervals on a queue-by-queue
- 126931 basis to guard against, for example, naive users specifying interval values too small on memory-
- 126932 intensive jobs. Otherwise, system overhead would adversely affect performance.
- 126933 The use of symbolic constants, such as *NO\_CHECKPOINT*, was introduced to lend a degree of
- 126934 formalism and portability to this option.

- 126935 Support for checkpointing is optional for servers. However, clients must provide for the `-c`  
 126936 option, since in a distributed environment the job may run on a server that does provide such  
 126937 support, even if the host of the client does not support the checkpoint feature.
- 126938 If the user does not specify the `-c` option, the default action is left unspecified by this option.  
 126939 Some implementations may wish to do checkpointing by default; others may wish to checkpoint  
 126940 only under an explicit request from the user.
- 126941 The *Priority* attribute has been made non-optional. All clients already had been required to  
 126942 support the `-p` option. The concept of prioritization is common in historical implementations.  
 126943 The default priority is left to the server to establish.
- 126944 The *Hold\_Types* attribute has been modified to allow for implementation-defined hold types to  
 126945 be passed to a batch server.
- 126946 It was the intent of the IEEE P1003.15 working group to mandate the support for the  
 126947 *Resource\_List* attribute in this option by referring to another amendment, specifically the  
 126948 IEEE P1003.1a draft standard. However, during the development of the IEEE P1003.1a draft  
 126949 standard this was excluded. As such this requirement has been removed from the normative  
 126950 text.
- 126951 The *Shell\_Path* attribute has been modified to accept a list of shell paths that are associated with  
 126952 a host. The name of the attribute has been changed to *Shell\_Path\_List*.

### 126953 C.3.3 Common Behavior for Batch Environment Utilities

- 126954 This section was defined to meet the goal of a “Small and Consistent Command Set” for this  
 126955 option.

## 126956 C.4 Utilities

- 126957 For the utilities included in POSIX.1-2008, see the RATIONALE sections on the individual  
 126958 reference pages.

### 126959 C.4.1 Utilities Removed in this Version

- 126960 None.

### 126961 C.4.2 Utilities Removed in the Previous Version

- 126962 The following utilities were removed in the previous version of this standard:

|        |                 |               |               |              |               |
|--------|-----------------|---------------|---------------|--------------|---------------|
| 126963 | <i>calendar</i> | <i>cu</i>     | <i>line</i>   | <i>pcat</i>  | <i>unpack</i> |
| 126964 | <i>cancel</i>   | <i>dircmp</i> | <i>lint</i>   | <i>pg</i>    | <i>uulog</i>  |
| 126965 | <i>cc</i>       | <i>dis</i>    | <i>lpstat</i> | <i>spell</i> | <i>uuname</i> |
| 126966 | <i>col</i>      | <i>egrep</i>  | <i>mail</i>   | <i>sum</i>   | <i>uupick</i> |
| 126967 | <i>cpio</i>     | <i>fgrep</i>  | <i>pack</i>   | <i>tar</i>   | <i>uuto</i>   |

126968 **C.4.3 Exclusion of Utilities**

126969 The set of utilities contained in POSIX.1-2008 is drawn from the base documents, with one  
 126970 addition: the *c99* utility. This section contains rationale for some of the deliberations that led to  
 126971 this set of utilities, and why certain utilities were excluded.

126972 Many utilities were evaluated by the standard developers; more historical utilities were  
 126973 excluded from the base documents than included. The following list contains many common  
 126974 UNIX system utilities that were not included as mandatory utilities, in the User Portability  
 126975 Utilities option, in the XSI option, or in one of the software development groups. It is logistically  
 126976 difficult for this rationale to distribute correctly the reasons for not including a utility among the  
 126977 various utility options. Therefore, this section covers the reasons for all utilities not included in  
 126978 POSIX.1-2008.

126979 This rationale is limited to a discussion of only those utilities actively or indirectly evaluated by  
 126980 the standard developers of the base documents, rather than the list of all known UNIX utilities  
 126981 from all its variants.

126982 *adb* The intent of the various software development utilities was to assist in the  
 126983 installation (rather than the actual development and debugging) of applications.  
 126984 This utility is primarily a debugging tool. Furthermore, many useful aspects of *adb*  
 126985 are very hardware-specific.

126986 *as* Assemblers are hardware-specific and are included implicitly as part of the  
 126987 compilers in POSIX.1-2008.

126988 *banner* The only known use of this command is as part of the *lp* printer header pages. It  
 126989 was decided that the format of the header is implementation-defined, so this utility  
 126990 is superfluous to application portability.

126991 *calendar* This reminder service program is not useful to conforming applications.

126992 *cancel* The *lp* (line printer spooling) system specified is the most basic possible and did  
 126993 not need this level of application control.

126994 *chroot* This is primarily of administrative use, requiring superuser privileges.

126995 *col* No utilities defined in POSIX.1-2008 produce output requiring such a filter. The  
 126996 *nroff* text formatter is present on many historical systems and will continue to  
 126997 remain as an extension; *col* is expected to be shipped by all the systems that ship  
 126998 *nroff*.

126999 *cpio* This has been replaced by *pax*, for reasons explained in the rationale for that utility.

127000 *cpp* This is subsumed by *c99*.

127001 *cu* This utility is terminal-oriented and is not useful from shell scripts or typical  
 127002 application programs.

127003 *dc* The functionality of this utility can be provided by the *bc* utility; *bc* was selected  
 127004 because it was easier to use and had superior functionality. Although the historical  
 127005 versions of *bc* are implemented using *dc* as a base, POSIX.1-2008 prescribes the  
 127006 interface and not the underlying mechanism used to implement it.

127007 *dircmp* Although a useful concept, the historical output of this directory comparison  
 127008 program is not suitable for processing in application programs. Also, the *diff -r*  
 127009 command gives equivalent functionality.

127010 *dis* Disassemblers are hardware-specific.

|        |               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|--------|---------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 127011 | <i>emacs</i>  | The community of <i>emacs</i> editing enthusiasts was adamant that the full <i>emacs</i> editor not be included in the base documents because they were concerned that an attempt to standardize this very powerful environment would encourage vendors to ship versions conforming strictly to the standard, but lacking the extensibility required by the community. The author of the original <i>emacs</i> program also expressed his desire to omit the program. Furthermore, there were a number of historical UNIX systems that did not include <i>emacs</i> , or included it without supporting it, but there were very few that did not include and support <i>vi</i> . |
| 127012 |               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 127013 |               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 127014 |               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 127015 |               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 127016 |               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 127017 |               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 127018 |               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 127019 | <i>ld</i>     | This is subsumed by <i>c99</i> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| 127020 | <i>line</i>   | The functionality of <i>line</i> can be provided with <i>read</i> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 127021 | <i>lint</i>   | This technology is partially subsumed by <i>c99</i> . It is also hard to specify the degree of checking for possible error conditions in programs in any compiler, and specifying what <i>lint</i> would do in these cases is equally difficult.                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| 127022 |               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 127023 |               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 127024 |               | It is fairly easy to specify what a compiler does. It requires specifying the language, what it does with that language, and stating that the interpretation of any incorrect program is unspecified. Unfortunately, any description of <i>lint</i> is required to specify what to do with erroneous programs. Since the number of possible errors and questionable programming practices is infinite, one cannot require <i>lint</i> to detect all errors of any given class.                                                                                                                                                                                                   |
| 127025 |               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 127026 |               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 127027 |               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 127028 |               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 127029 |               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 127030 |               | Additionally, some vendors complained that since many compilers are distributed in a binary form without a <i>lint</i> facility (because the ISO C standard does not require one), implementing the standard as a stand-alone product will be much harder. Rather than being able to build upon a standard compiler component (simply by providing <i>c99</i> as an interface), source to that compiler would most likely need to be modified to provide the <i>lint</i> functionality. This was considered a major burden on system providers for a very small gain to developers (users).                                                                                      |
| 127031 |               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 127032 |               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 127033 |               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 127034 |               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 127035 |               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 127036 |               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 127037 | <i>login</i>  | This utility is terminal-oriented and is not useful from shell scripts or typical application programs.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 127038 |               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 127039 | <i>lorder</i> | This utility is an aid in creating an implementation-defined detail of object libraries that the standard developers did not feel required standardization.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 127040 |               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 127041 | <i>lpstat</i> | The <i>lp</i> system specified is the most basic possible and did not need this level of application control.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 127042 |               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 127043 | <i>mail</i>   | This utility was omitted in favor of <i>mailx</i> because there was a considerable functionality overlap between the two.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 127044 |               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 127045 | <i>mknod</i>  | This was omitted in favor of <i>mkfifo</i> , as <i>mknod</i> has too many implementation-defined functions.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 127046 |               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 127047 | <i>news</i>   | This utility is terminal-oriented and is not useful from shell scripts or typical application programs.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 127048 |               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 127049 | <i>pack</i>   | This compression program was considered inferior to <i>compress</i> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 127050 | <i>passwd</i> | This utility was proposed in a historical draft of the base documents but met with too many objections to be included. There were various reasons:                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| 127051 |               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 127052 |               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 127053 |               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 127054 |               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |

- Changing a password should not be viewed as a command, but as part of the login sequence. Changing a password should only be done while a trusted path is in effect.

|        |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|--------|--------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 127055 |              | <ul style="list-style-type: none"> <li>• Even though the text in early drafts was intended to allow a variety of implementations to conform, the security policy for one site may differ from another site running with identical hardware and software. One site might use password authentication while the other did not. Vendors could not supply a <i>passwd</i> utility that would conform to POSIX.1-2008 for all sites using their system.</li> <li>• This is really a subject for a system administration working group or a security working group.</li> </ul> |
| 127056 |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 127057 |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 127058 |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 127059 |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 127060 |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 127061 |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 127062 |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 127063 | <i>pcat</i>  | This compression program was considered inferior to <i>zcat</i> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 127064 | <i>pg</i>    | This duplicated many of the features of the <i>more</i> pager, which was preferred by the standard developers.                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| 127065 |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 127066 | <i>prof</i>  | The intent of the various software development utilities was to assist in the installation (rather than the actual development and debugging) of applications. This utility is primarily a debugging tool.                                                                                                                                                                                                                                                                                                                                                               |
| 127067 |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 127068 |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 127069 | RCS          | RCS was originally considered as part of a version control utilities portion of the scope. However, this aspect was abandoned by the standard developers. SCCS is now included as an optional part of the XSI option.                                                                                                                                                                                                                                                                                                                                                    |
| 127070 |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 127071 |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 127072 | <i>red</i>   | Restricted editor. This was not considered by the standard developers because it never provided the level of security restriction required.                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 127073 |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 127074 | <i>rsh</i>   | Restricted shell. This was not considered by the standard developers because it does not provide the level of security restriction that is implied by historical documentation.                                                                                                                                                                                                                                                                                                                                                                                          |
| 127075 |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 127076 |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 127077 | <i>sdb</i>   | The intent of the various software development utilities was to assist in the installation (rather than the actual development and debugging) of applications. This utility is primarily a debugging tool. Furthermore, some useful aspects of <i>sdb</i> are very hardware-specific.                                                                                                                                                                                                                                                                                    |
| 127078 |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 127079 |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 127080 |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 127081 | <i>sdiff</i> | The “side-by-side <i>diff</i> ” utility from System V was omitted because it is used infrequently, and even less so by conforming applications. Despite being in System V, it is not in the SVID or XPG.                                                                                                                                                                                                                                                                                                                                                                 |
| 127082 |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 127083 |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 127084 | <i>shar</i>  | Any of the numerous “shell archivers” were excluded because they did not meet the requirement of existing practice.                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 127085 |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 127086 | <i>shl</i>   | This utility is terminal-oriented and is not useful from shell scripts or typical application programs. The job control aspects of the shell command language are generally more useful.                                                                                                                                                                                                                                                                                                                                                                                 |
| 127087 |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 127088 |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 127089 | <i>size</i>  | The intent of the various software development utilities was to assist in the installation (rather than the actual development and debugging) of applications. This utility is primarily a debugging tool.                                                                                                                                                                                                                                                                                                                                                               |
| 127090 |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 127091 |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 127092 | <i>spell</i> | This utility is not useful from shell scripts or typical application programs. The <i>spell</i> utility was considered, but was omitted because there is no known technology that can be used to make it recognize general language for user-specified input without providing a complete dictionary along with the input file.                                                                                                                                                                                                                                          |
| 127093 |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 127094 |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 127095 |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 127096 | <i>su</i>    | This utility is not useful from shell scripts or typical application programs. (There was also sentiment to avoid security-related utilities.)                                                                                                                                                                                                                                                                                                                                                                                                                           |
| 127097 |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |

|        |               |                                                                                                                                                   |
|--------|---------------|---------------------------------------------------------------------------------------------------------------------------------------------------|
| 127098 | <i>sum</i>    | This utility was renamed <i>cksum</i> .                                                                                                           |
| 127099 | <i>tar</i>    | This has been replaced by <i>pax</i> , for reasons explained in the rationale for that utility.                                                   |
| 127100 | <i>unpack</i> | This compression program was considered inferior to <i>uncompress</i> .                                                                           |
| 127101 | <i>wall</i>   | This utility is terminal-oriented and is not useful in shell scripts or typical applications. It is generally used only by system administrators. |
| 127102 |               |                                                                                                                                                   |

127103

# *Rationale (Informative)*

127104

## **Part D:**

127105

## **Portability Considerations**

127106

*The Open Group*

127107

*The Institute of Electrical and Electronics Engineers, Inc.*





127108

127109

## Portability Considerations (Informative)

127110

This section contains information to satisfy various international requirements:

127111

- [Section D.1](#) describes perceived user requirements.

127112

- [Section D.2](#) (on page 3719) indicates how the facilities of POSIX.1-2008 satisfy those requirements.

127113

127114

- [Section D.3](#) (on page 3727) offers guidance to writers of profiles on how the configurable options, limits, and optional behavior of POSIX.1-2008 should be cited in profiles.

127115

### D.1 User Requirements

127117

This section describes the user requirements that were perceived by the standard developers. The primary source for these requirements was an analysis of historical practice in widespread use, as typified by the base documents listed in [Section A.1.1](#) (on page 3439).

127118

127119

127120

POSIX.1-2008 addresses the needs of users requiring open systems solutions for source code portability of applications. It currently addresses users requiring open systems solutions for source-code portability of applications involving multi-programming and process management (creating processes, signaling, and so on); access to files and directories in a hierarchy of file systems (opening, reading, writing, deleting files, and so on); access to asynchronous communications ports and other special devices; access to information about other users of the system; facilities supporting applications requiring bounded (realtime) response.

127121

127122

127123

127124

127125

127126

127127

The following users are identified for POSIX.1-2008:

127128

- Those employing applications written in high-level languages, such as C, Ada, or FORTRAN.

127129

127130

- Users who desire conforming applications that do not necessarily require the characteristics of high-level languages (for example, the speed of execution of compiled languages or the relative security of source code intellectual property inherent in the compilation process).

127131

127132

127133

127134

- Users who desire conforming applications that can be developed quickly and can be modified readily without the use of compilers and other system components that may be unavailable on small systems or those without special application development capabilities.

127135

127136

127137

127138

- Users who interact with a system to achieve general-purpose time-sharing capabilities common to most business or government offices or academic environments: editing, filing, inter-user communications, printing, and so on.

127139

127140

127141

- Users who develop applications for POSIX-conformant systems.

127142

- Users who develop applications for UNIX systems.

127143

An acknowledged restriction on applicable users is that they are limited to the group of individuals who are familiar with the style of interaction characteristic of historically-derived systems based on one of the UNIX operating systems (as opposed to other historical systems

127144

127145

127146 with different models, such as MS/DOS, Macintosh, VMS, MVS, and so on). Typical users  
127147 would include program developers, engineers, or general-purpose time-sharing users.

127148 The requirements of users of POSIX.1-2008 can be summarized as a single goal: *application source*  
127149 *portability*. The requirements of the user are stated in terms of the requirements of portability of  
127150 applications. This in turn becomes a requirement for a standardized set of syntax and semantics  
127151 for operations commonly found on many operating systems.

127152 The following sections list the perceived requirements for application portability.

#### 127153 **D.1.1 Configuration Interrogation**

127154 An application must be able to determine whether and how certain optional features are  
127155 provided and to identify the system upon which it is running, so that it may appropriately adapt  
127156 to its environment.

127157 Applications must have sufficient information to adapt to varying behaviors of the system.

#### 127158 **D.1.2 Process Management**

127159 An application must be able to manage itself, either as a single process or as multiple processes.  
127160 Applications must be able to manage other processes when appropriate.

127161 Applications must be able to identify, control, create, and delete processes, and there must be  
127162 communication of information between processes and to and from the system.

127163 Applications must be able to use multiple flows of control with a process (threads) and  
127164 synchronize operations between these flows of control.

#### 127165 **D.1.3 Access to Data**

127166 Applications must be able to operate on the data stored on the system, access it, and transmit it  
127167 to other applications. Information must have protection from unauthorized or accidental access  
127168 or modification.

#### 127169 **D.1.4 Access to the Environment**

127170 Applications must be able to access the external environment to communicate their input and  
127171 results.

#### 127172 **D.1.5 Access to Determinism and Performance Enhancements**

127173 Applications must have sufficient control of resource allocation to ensure the timeliness of  
127174 interactions with external objects.

**127175 D.1.6 Operating System-Dependent Profile**

127176 The capabilities of the operating system may make certain optional characteristics of the base  
127177 language in effect no longer optional, and this should be specified.

**127178 D.1.7 I/O Interaction**

127179 The interaction between the C language I/O subsystem (*stdio*) and the I/O subsystem of  
127180 POSIX.1-2008 must be specified.

**127181 D.1.8 Internationalization Interaction**

127182 The effects of the environment of POSIX.1-2008 on the internationalization facilities of the C  
127183 language must be specified.

**127184 D.1.9 C-Language Extensions**

127185 Certain functions in the C language must be extended to support the additional capabilities  
127186 provided by POSIX.1-2008.

**127187 D.1.10 Command Language**

127188 Users should be able to define procedures that combine simple tools and/or applications into  
127189 higher-level components that perform to the specific needs of the user. The user should be able  
127190 to store, recall, use, and modify these procedures. These procedures should employ a powerful  
127191 command language that is used for recurring tasks in conforming applications (scripts) in the  
127192 same way that it is used interactively to accomplish one-time tasks. The language and the  
127193 utilities that it uses must be consistent between systems to reduce errors and retraining.

**127194 D.1.11 Interactive Facilities**

127195 Use the system to accomplish individual tasks at an interactive terminal. The interface should be  
127196 consistent, intuitive, and offer usability enhancements to increase the productivity of terminal  
127197 users, reduce errors, and minimize retraining costs. Online documentation or usage assistance  
127198 should be available.

**127199 D.1.12 Accomplish Multiple Tasks Simultaneously**

127200 Access applications and interactive facilities from a single terminal without requiring serial  
127201 execution: switch between multiple interactive tasks; schedule one-time or periodic background  
127202 work; display the status of all work in progress or scheduled; influence the priority scheduling  
127203 of work, when authorized.

**127204 D.1.13 Complex Data Manipulation**

127205 Manipulate data in files in complex ways: sort, merge, compare, translate, edit, format, pattern  
127206 match, select subsets (strings, columns, fields, rows, and so on). These facilities should be  
127207 available to both conforming applications and interactive users.

**127208 D.1.14 File Hierarchy Manipulation**

127209 Create, delete, move/rename, copy, backup/archive, and display files and directories. These  
127210 facilities should be available to both conforming applications and interactive users.

**127211 D.1.15 Locale Configuration**

127212 Customize applications and interactive sessions for the cultural and language conventions of the  
127213 user. Employ a wide variety of standard character encodings. These facilities should be available  
127214 to both conforming applications and interactive users.

**127215 D.1.16 Inter-User Communication**

127216 Send messages or transfer files to other users on the same system or other systems on a network.  
127217 These facilities should be available to both conforming applications and interactive users.

**127218 D.1.17 System Environment**

127219 Display information about the status of the system (activities of users and their interactive and  
127220 background work, file system utilization, system time, configuration, and presence of optional  
127221 facilities) and the environment of the user (terminal characteristics, and so on). Inform the  
127222 system operator/administrator of problems. Control access to user files and other resources.

**127223 D.1.18 Printing**

127224 Output files on a variety of output device classes, accessing devices on local or network-  
127225 connected systems. Control (or influence) the formatting, priority scheduling, and output  
127226 distribution of work. These facilities should be available to both conforming applications and  
127227 interactive users.

**127228 D.1.19 Software Development**

127229 Develop (create and manage source files, compile/interpret, debug) portable open systems  
127230 applications and package them for distribution to, and updating of, other systems.

## D.2 Portability Capabilities

This section describes the significant portability capabilities of POSIX.1-2008 and indicates how the user requirements listed in [Section D.1](#) (on page 3715) are addressed. The capabilities are listed in the same format as the preceding user requirements; they are summarized below:

- Configuration Interrogation
- Process Management
- Access to Data
- Access to the Environment
- Access to Determinism and Performance Enhancements
- Operating System-Dependent Profile
- I/O Interaction
- Internationalization Interaction
- C-Language Extensions
- Command Language
- Interactive Facilities
- Accomplish Multiple Tasks Simultaneously
- Complex Data Manipulation
- File Hierarchy Manipulation
- Locale Configuration
- Inter-User Communication
- System Environment
- Printing
- Software Development

### D.2.1 Configuration Interrogation

The *uname()* operation provides basic identification of the system. The *sysconf()*, *pathconf()*, and *fpathconf()* functions and the *getconf* utility provide means to interrogate the implementation to determine how to adapt to the environment in which it is running. These values can be either static (indicating that all instances of the implementation have the same value) or dynamic (indicating that different instances of the implementation have the different values, or that the value may vary for other reasons, such as reconfiguration).

#### Unsatisfied Requirements

None directly. However, as new areas are added, there will be a need for additional capability in this area.

## D.2.2 Process Management

The *fork()*, *exec* family, *posix\_spawn()*, and *posix\_spawnp()* functions provide for the creation of new processes or the insertion of new applications into existing processes. The *\_Exit()*, *\_exit()*, *exit()*, and *abort()* functions allow for the termination of a process by itself. The *wait()*, *waitid()*, and *waitpid()* functions allow one process to deal with the termination of another.

The *times()* function allows for basic measurement of times used by a process. Various functions, including *fstat()*, *getegid()*, *geteuid()*, *getgid()*, *getgrgid()*, *getgrnam()*, *getlogin()*, *getpid()*, *getppid()*, *getpwnam()*, *getpwuid()*, *getuid()*, *lstat()*, and *stat()*, provide for access to the identifiers of processes and the identifiers and names of owners of processes (and files).

The various functions operating on environment variables provide for communication of information (primarily user-configurable defaults) from a parent to child processes.

The operations on the current working directory control and interrogate the directory from which relative pathname searches start. The *umask()* function controls the default protections applied to files created by the process.

The *alarm()*, *pause()*, *sleep()*, *ualarm()*, and *usleep()* operations allow the process to suspend until a timer has expired or to be notified when a period of time has elapsed. The *time()* operation interrogates the current time and date.

The signal mechanism provides for communication of events either from other processes or from the environment to the application, and the means for the application to control the effect of these events. The mechanism provides for external termination of a process and for a process to suspend until an event occurs. The mechanism also provides for a value to be associated with an event.

Job control provides a means to group processes and control them as groups, and to control their access to the function between the user and the system (the “controlling terminal”). It also provides the means to suspend and resume processes.

The Process Scheduling option provides control of the scheduling and priority of a process.

The Message Passing option provides a means for interprocess communication involving small amounts of data.

The Memory Management facilities provide control of memory resources and for the sharing of memory. This functionality is mandatory on POSIX-conforming systems.

The Threads facilities provide multiple flows of control with a process (threads), synchronization between threads (including mutexes, barriers, and spin locks), association of data with threads, and controlled cancellation of threads.

The XSI interprocess communications functionality provide an alternate set of facilities to manipulate semaphores, message queues, and shared memory. These are provided on XSI-conformant systems to support conforming applications developed to run on UNIX systems.

## D.2.3 Access to Data

The *open()*, *close()*, *fclose()*, *fopen()*, and *pipe()* functions provide for access to files and data. Such files may be regular files, interprocess data channels (pipes), or devices. Additional types of objects in the file system are permitted and are being contemplated for standardization.

The *access()*, *chmod()*, *chown()*, *dup()*, *dup2()*, *fchmod()*, *fcntl()*, *fstat()*, *ftruncate()*, *lstat()*, *readlink()*, *realpath()*, *stat()*, and *utime()* functions allow for control and interrogation of file and file-related objects (including symbolic links), and their ownership, protections, and timestamps.

127307 The *fgetc()*, *fputc()*, *fread()*, *fseek()*, *fsetpos()*, *fwrite()*, *getc()*, *getchar()*, *lseek()*, *putchar()*, *putc()*,  
 127308 *read()*, and *write()* functions provide for data transfer from the application to files (in all their  
 127309 forms).

127310 The *closedir()*, *link()*, *mkdir()*, *opendir()*, *readdir()*, *rename()*, *rmdir()*, *rewinddir()*, and *unlink()*  
 127311 functions provide for a complete set of operations on directories. Directories can arbitrarily  
 127312 contain other directories, and a single file can be mentioned in more than one directory.

127313 The *faccessat()*, *openat()*, *fchmodat()*, *fchownat()*, *fstatat()*, *linkat()*, *renameat()*, *readlinkat()*,  
 127314 *symlinkat()*, and *unlinkat()* functions allow for race-free and thread-safe file access. The  
 127315 motivation for the introduction of these functions was as follows:

- 127316 • Interfaces taking a pathname may be limited by the maximum length of a pathname  
 127317 ({PATH\_MAX}). The absolute path of files can far exceed this length. The alternative  
 127318 solution of changing the working directory and using relative pathnames is not thread-  
 127319 safe.
- 127320 • A second motivation is that files accessed outside the current working directory are subject  
 127321 to attacks caused by the race condition created by changing any of the elements of the  
 127322 pathnames used.
- 127323 • A third motivation is to allow application code which makes use of a virtual current  
 127324 working directory for each individual thread. In the alternative model there is only one  
 127325 current working directory for all threads.

127326 The file-locking mechanism provides for advisory locking (protection during transactions) of  
 127327 ranges of bytes (in effect, records) in a file.

127328 The *confstr()*, *fpathconf()*, *pathconf()*, and *sysconf()* functions provide for enquiry as to the  
 127329 behavior of the system where variability is permitted.

127330 The asynchronous input and output functions *aio\_cancel()*, *aio\_error()*, *aio\_fsync()*, *aio\_read()*,  
 127331 *aio\_return()*, *aio\_suspend()*, *aio\_write()*, and *lio\_listio()* provide for initiation and control of  
 127332 asynchronous data transfers.

127333 The Synchronized Input and Output option provides for assured commitment of data to media.

#### 127334 D.2.4 Access to the Environment

127335 The operations and types in XBD are provided for access to asynchronous serial devices. The  
 127336 primary intended use for these is the controlling terminal for the application (the interaction  
 127337 point between the user and the system). They are general enough to be used to control any  
 127338 asynchronous serial device. The functions are also general enough to be used with many other  
 127339 device types as a user interface when some emulation is provided.

127340 Less detailed access is provided for other device types, but in many instances an application  
 127341 need not know whether an object in the file system is a device or a regular file to operate  
 127342 correctly.

127343 **Unsatisfied Requirements**

127344 Detailed control of common device classes, specifically magnetic tape, is not provided.

127345 **D.2.5 Bounded (Realtime) Response**

127346 The realtime signal functions *sigqueue()*, *sigtimedwait()*, and *sigwaitinfo()* provide queued signals  
127347 and the prioritization of the handling of signals.

127348 The SCHED\_FIFO, SCHED\_SPORADIC, and SCHED\_RR scheduling policies provide control  
127349 over processor allocation.

127350 The semaphore functions *sem\_close()*, *sem\_destroy()*, *sem\_getvalue()*, *sem\_init()*, *sem\_open()*,  
127351 *sem\_post()*, *sem\_timedwait()*, *sem\_trywait()*, *sem\_unlink()*, and *sem\_wait()* provide high-  
127352 performance synchronization.

127353 The memory management functions provide memory locking for control of memory allocation,  
127354 file mapping for high performance, and shared memory for high-performance interprocess  
127355 communication. The Message Passing option provides for interprocess communication without  
127356 being dependent on shared memory.

127357 The timers functions *clock\_getres()*, *clock\_gettime()*, *clock\_settime()*, *nanosleep()*, *timer\_create()*,  
127358 *timer\_delete()*, *timer\_getoverrun()*, *timer\_gettime()*, and *timer\_settime()* provide functionality to  
127359 manipulate clocks and timers and include a high resolution function called *nanosleep()* with a  
127360 finer resolution than the *sleep()* function.

127361 The timeout functions — *pthread\_mutex\_timedlock()*, *pthread\_rwlock\_timedrdlock()*,  
127362 *pthread\_rwlock\_timedwrlock()*, and *sem\_timedwait()* — the Typed Memory Objects option and the  
127363 Monotonic Clock option provide further facilities for applications to use to obtain predictable  
127364 bounded response.

127365 **D.2.6 Operating System-Dependent Profile**

127366 POSIX.1-2008 makes no distinction between text and binary files. The values of EXIT\_SUCCESS  
127367 and EXIT\_FAILURE are further defined.

127368 **Unsatisfied Requirements**

127369 None known, but the ISO C standard may contain some additional options that could be  
127370 specified.

127371 **D.2.7 I/O Interaction**

127372 POSIX.1-2008 defines how each of the ISO C standard *stdio* functions interact with the POSIX.1  
127373 operations, typically specifying the behavior in terms of POSIX.1 operations.



127374 **Unsatisfied Requirements**

127375 None.

127376 **D.2.8 Internationalization Interaction**

127377 The POSIX.1-2008 environment operations provide a means to define the environment for  
 127378 *setlocale()* and time functions such as *ctime()*. The *tzset()* function is provided to set time  
 127379 conversion information.

127380 The *nl\_langinfo()* function is provided to query locale-specific cultural settings.

127381 The multiple concurrent locale functions *duplocale()*, *freelocale()*, *is\*\_l()*, *newlocale()*,  
 127382 *strcasecmp\_l()*, *strcoll\_l()*, *strfmon\_l()*, *strncasecmp\_l()*, *strxfrm\_l()*, *tolower\_l()*, *toupper\_l()*,  
 127383 *towctrans\_l()*, *towlower\_l()*, *towupper\_l()*, *uselocale\_l()*, *wscasecmp\_l()*, *wscoll\_l()*, *wcsncasecmp\_l()*,  
 127384 *wcsxfrm\_l()*, *wctrans\_l()*, and *wctype\_l()* are provide to support per-thread locale information.

127385 **Unsatisfied Requirements**

127386 None.

127387 **D.2.9 C-Language Extensions**

127388 The *setjmp()* and *longjmp()* functions are not defined to be cognizant of the signal masks defined  
 127389 for POSIX.1. The *sigsetjmp()* and *siglongjmp()* functions are provided to fill this gap.

127390 The *\_setjmp()* and *\_longjmp()* functions are provided as XSI options to support historic practice.

127391 **Unsatisfied Requirements**

127392 None.

127393 **D.2.10 Command Language**

127394 The shell command language, as described in XCU [Chapter 2](#) (on page 2319), is a common  
 127395 language useful in batch scripts, through an API to high-level languages (for the C-Language  
 127396 Binding option, *system()* and *popen()*) and through an interactive terminal (see the *sh* utility).  
 127397 The shell language has many of the characteristics of a high-level language, but it has been  
 127398 designed to be more suitable for user terminal entry and includes interactive debugging  
 127399 facilities. Through the use of pipelining, many complex commands can be constructed from  
 127400 combinations of data filters and other common components. Shell scripts can be created, stored,  
 127401 recalled, and modified by the user with simple editors.

127402 In addition to the basic shell language, the following utilities offer features that simplify and  
 127403 enhance programmatic access to the utilities and provide features normally found only in high-  
 127404 level languages: *basename*, *bc*, *command*, *dirname*, *echo*, *env*, *expr*, *false*, *printf*, *read*, *sleep*, *tee*, *test*,  
 127405 *time\**,<sup>9</sup> *true*, *wait*, *xargs*, and all of the special built-in utilities in XCU [Section 2.14](#) (on page 2356).

---

127406 9. The utilities listed with an asterisk here and later in this section are present only on systems which support the User Portability Utilities  
 127407 option. There may be further restrictions on the utilities offered with various configuration option combinations; see the individual utility  
 127408 descriptions.

127409 **Unsatisfied Requirements**

127410 None.

127411 **D.2.11 Interactive Facilities**

127412 The utilities offer a common style of command-line interface through conformance to the Utility  
127413 Syntax Guidelines (see XBD [Section 12.2](#), on page 215) and the common utility defaults (see XCU  
127414 [Section 1.4](#), on page 2310). The *sh* utility offers an interactive command-line history and editing  
127415 facility.

127416 The following utilities can be used interactively as well as by scripts; *alias*, *fc*, *mailx*, *unalias*, and  
127417 *write*.

127418 The following utilities in the User Portability Utilities option provide for interactive use: *ex*, *more*,  
127419 and *vi*; the *man* utility offers online access to system documentation.

127420 **Unsatisfied Requirements**

127421 The command line interface to individual utilities is as intuitive and consistent as historical  
127422 practice allows. Work underway based on graphical user interfaces may be more suitable for  
127423 novice or occasional users of the system.

127424 **D.2.12 Accomplish Multiple Tasks Simultaneously**

127425 The shell command language offers background processing through the asynchronous list  
127426 command form; see XCU [Section 2.9](#) (on page 2338).

127427 The *nohup* utility makes background processing more robust and usable.

127428 The *kill* utility can terminate background jobs.

127429 The following utilities support periodic job scheduling, control, and display: *at*, *batch*, *crontab*,  
127430 *nice*, *ps*, and *renice*.

127431 When the User Portability Utilities option is supported, the following utilities allow  
127432 manipulation of jobs: *bg*, *fg*, and *jobs*.

127433 **Unsatisfied Requirements**

127434 Terminals with multiple windows may be more suitable for some multi-tasking interactive uses  
127435 than the job control approach in POSIX.1-2008. See the comments on graphical user interfaces in  
127436 [Section D.2.11](#). The *nice* and *renice* utilities do not necessarily take advantage of complex system  
127437 scheduling algorithms that are supported by the realtime options within POSIX.1-2008.

127438 **D.2.13 Complex Data Manipulation**

127439 The following utilities address user requirements in this area: *asa*, *awk*, *bc*, *cmp*, *comm*, *csplit*, *cut*,  
127440 *dd*, *diff*, *ed*, *ex\**, *expand*, *expr*, *find*, *fold*, *grep*, *head*, *join*, *od*, *paste*, *pr*, *printf*, *sed*, *sort*, *split*, *tabs*, *tail*, *tr*,  
127441 *unexpand*, *uniq*, *uudecode*, *uuencode*, and *wc*.

127442 **Unsatisfied Requirements**

127443 Sophisticated text formatting utilities, such as *troff* or *TeX*, are not included. Standards work in  
 127444 the area of SGML may satisfy this.

127445 **D.2.14 File Hierarchy Manipulation**

127446 The following utilities address user requirements in this area: *basename*, *cd*, *chgrp*, *chmod*, *chown*,  
 127447 *cksum*, *cp*, *dd*, *df*, *diff*, *dirname*, *du*, *find*, *ls*, *ln*, *mkdir*, *mkfifo*, *mv*, *patch*, *pathchk*, *pax*, *pwd*, *rm*, *rmdir*,  
 127448 *test*, and *touch*.

127449 **Unsatisfied Requirements**

127450 Some graphical user interfaces offer more intuitive file manager components that allow file  
 127451 manipulation through the use of icons for novice users.

127452 **D.2.15 Locale Configuration**

127453 The standard utilities are affected by the various *LC\_* variables to achieve locale-dependent  
 127454 operation: character classification, collation sequences, regular expressions and shell pattern  
 127455 matching, date and time formats, numeric formatting, and monetary formatting. When the  
 127456 POSIX2\_LOCALEDEF option is supported, applications can provide their own locale definition  
 127457 files.

127458 The following utilities address user requirements in this area: *date*, *ed*, *ex\**, *find*, *grep*, *locale*,  
 127459 *localedef*, *more\**, *sed*, *sh*, *sort*, *tr*, *uniq*, and *vi\**.

127460 The *iconv()*, *iconv\_close()*, and *iconv\_open()* functions are available to allow an application to  
 127461 convert character data between supported character sets.

127462 The *gencat* utility and the *catopen()*, *catclose()*, and *catgets()* functions provide for message  
 127463 catalog manipulation.

127464 **Unsatisfied Requirements**

127465 Some aspects of multi-byte character and state-encoded character encodings have not yet been  
 127466 addressed. The C-language functions, such as *getopt()*, are generally limited to single-byte  
 127467 characters. The effect of the *LC\_MESSAGES* variable on message formats is only suggested at  
 127468 this time.

127469 **D.2.16 Inter-User Communication**

127470 The following utilities address user requirements in this area: *cksum*, *mailx*, *mesg*, *patch*, *pax*, *talk*,  
 127471 *uudecode*, *uuencode*, *who*, and *write*.

127472 The historical UUCP utilities are included as a separate UUCP Utilities option.

127473 **Unsatisfied Requirements**

127474 None.

127475 **D.2.17 System Environment**

127476 The following utilities address user requirements in this area: *chgrp*, *chmod*, *chown*, *df*, *du*, *env*,  
127477 *getconf*, *id*, *logger*, *logname*, *mesg*, *newgrp*, *ps*, *stty*, *tput*, *tty*, *umask*, *uname*, and *who*.

127478 The *closelog()*, *openlog()*, *setlogmask()*, and *syslog()* functions provide system logging facilities on  
127479 XSI-conformant systems; these are analogous to the *logger* utility.

127480 **Unsatisfied Requirements**

127481 None.

127482 **D.2.18 Printing**

127483 The following utilities address user requirements in this area: *pr* and *lp*.

127484 **Unsatisfied Requirements**

127485 There are no features to control the formatting or scheduling of the print jobs.

127486 **D.2.19 Software Development**

127487 The following utilities address user requirements in this area: *ar*, *asa*, *awk*, *c99*, *ctags*, *fort77*,  
127488 *getconf*, *getopts*, *lex*, *localedef*, *make*, *nm*, *od*, *patch*, *pax*, *strings*, *strip*, *time*, and *yacc*.

127489 The *system()*, *popen()*, *pclose()*, *regcomp()*, *regex()*, *regerror()*, *regfree()*, *fnmatch()*, *getopt()*,  
127490 *glob()*, *globfree()*, *wordexp()*, and *wordfree()* functions allow C-language programmers to access  
127491 some of the interfaces used by the utilities, such as argument processing, regular expressions,  
127492 and pattern matching.

127493 The SCCS source-code control system utilities are available on systems supporting the XSI  
127494 Development option.

127495 **Unsatisfied Requirements**

127496 There are no language-specific development tools related to languages other than C and  
127497 FORTRAN. The C tools are more complete and varied than the FORTRAN tools. There is no  
127498 data dictionary or other CASE-like development tools.

127499 **D.2.20 Future Growth**

127500 It is arguable whether or not all functionality to support applications is potentially within the  
127501 scope of POSIX.1-2008. As a simple matter of practicality, it cannot be. Areas such as graphics,  
127502 application domain-specific functionality, windowing, and so on, should be in unique standards.  
127503 As such, they are properly “Unsatisfied Requirements” in terms of providing fully conforming  
127504 applications, but ones which are outside the scope of POSIX.1-2008.

127505 However, as the standards evolve, certain functionality once considered “exotic” enough to be  
127506 part of a separate standard become common enough to be included in a core standard such as  
127507 this. Realtime and networking, for example, have both moved from separate standards (with

127508 much difficult cross-referencing) into this standard over time, and although no specific areas  
 127509 have been identified for inclusion in a future version, such inclusions seem likely.

### 127510 **D.3 Profiling Considerations**

127511 This section offers guidance to writers of profiles on how the configurable options, limits, and  
 127512 optional behavior of POSIX.1-2008 should be cited in profiles. Profile writers should consult the  
 127513 general guidance in POSIX.0 when writing POSIX Standardized Profiles.

127514 The information in this section is an inclusive list of features that should be considered by profile  
 127515 writers. Subsetting of POSIX.1-2008 should follow XBD [Section 2.1.5.1](#) (on page 20). A set of  
 127516 profiling options is described in [Appendix E](#) (on page 3741).

#### 127517 **D.3.1 Configuration Options**

127518 There are two set of options suggested by POSIX.1-2008: those for POSIX-conforming systems  
 127519 and those for X/Open System Interface (XSI) conformance. The requirements for XSI  
 127520 conformance are documented in the Base Definitions volume of POSIX.1-2008 and not discussed  
 127521 further here, as they superset the POSIX conformance requirements.

#### 127522 **D.3.2 Configuration Options (Shell and Utilities)**

127523 There are three broad optional configurations for the Shell and Utilities volume of POSIX.1-2008:  
 127524 basic execution system, development system, and user portability interactive system. The  
 127525 options to support these, and other minor configuration options, are listed in XBD [Chapter 2](#) (on  
 127526 page 15). Profile writers should consult the following list and the comments concerning user  
 127527 requirements addressed by various components in [Section D.2](#) (on page 3719).

127528 POSIX2\_UPE

127529 The system supports the User Portability Utilities option.

127530 This option is a requirement for a user portability interactive system. It is required  
 127531 frequently except for those systems, such as embedded realtime or dedicated application  
 127532 systems, that support little or no interactive time-sharing work by users or operators. XSI-  
 127533 conformant systems support this option.

127534 POSIX2\_SW\_DEV

127535 The system supports the Software Development Utilities option.

127536 This option is required by many systems, even those in which actual software development  
 127537 does not occur. The *make* utility, in particular, is required by many application software  
 127538 packages as they are installed onto the system. If POSIX2\_C\_DEV is supported,  
 127539 POSIX2\_SW\_DEV is almost a mandatory requirement because of *ar* and *make*.

127540 POSIX2\_C\_BIND

127541 The system supports the C-Language Bindings option.

127542 This option is required on some implementations developing complex C applications or on  
 127543 any system installing C applications in source form that require the functions in this option.  
 127544 The *system()* and *popen()* functions, in particular, are widely used by applications; the  
 127545 others are rather more specialized.

- 127546 POSIX2\_C\_DEV  
 127547 The system supports the C-Language Development Utilities option.
- 127548 This option is required by many systems, even those in which actual C-language software  
 127549 development does not occur. The *c99* utility, in particular, is required by many application  
 127550 software packages as they are installed onto the system. The *lex* and *yacc* utilities are used  
 127551 less frequently.
- 127552 POSIX2\_FORT\_DEV  
 127553 The system supports the FORTRAN Development Utilities option
- 127554 As with C, this option is needed on any system developing or installing FORTRAN  
 127555 applications in source form.
- 127556 POSIX2\_FORT\_RUN  
 127557 The system supports the FORTRAN Runtime Utilities option.
- 127558 This option is required for some FORTRAN applications that need the *asa* utility to convert  
 127559 Hollerith printing statement output. It is unknown how frequently this occurs.
- 127560 POSIX2\_LOCALEDEF  
 127561 The system supports the creation of locales.
- 127562 This option is needed if applications require their own customized locale definitions to  
 127563 operate. It is presently unknown whether many applications are dependent on this.  
 127564 However, the option is virtually mandatory for systems in which internationalized  
 127565 applications are developed.
- 127566 XSI-conformant systems support this option.
- 127567 POSIX2\_PBS  
 127568 The system supports the Batch Environment Services and Utilities option.
- 127569 POSIX2\_PBS\_ACCOUNTING  
 127570 The system supports the optional feature of accounting within the Batch Environment  
 127571 Services and Utilities option. It will be required in servers that implement the optional  
 127572 feature of accounting.
- 127573 POSIX2\_PBS\_CHECKPOINT  
 127574 The system supports the optional feature of checkpoint/restart within the Batch  
 127575 Environment Services and Utilities option.
- 127576 POSIX2\_PBS\_LOCATE  
 127577 The system supports the optional feature of locating batch jobs within the Batch  
 127578 Environment Services and Utilities option.
- 127579 POSIX2\_PBS\_MESSAGE  
 127580 The system supports the optional feature of sending messages to batch jobs within the Batch  
 127581 Environment Services and Utilities option.
- 127582 POSIX2\_PBS\_TRACK  
 127583 The system supports the optional feature of tracking batch jobs within the Batch  
 127584 Environment Services and Utilities option.
- 127585 POSIX2\_CHAR\_TERM  
 127586 The system supports at least one terminal type capable of all operations described in  
 127587 POSIX.1-2008.
- 127588 On systems with POSIX2\_UPE, this option is almost always required. It was developed  
 127589 solely to allow certain specialized vendors and user applications to bypass the requirement  
 127590 for general-purpose asynchronous terminal support. For example, an application and

127591 system that was suitable for block-mode terminals, such as IBM 3270s, would not need this  
 127592 option.  
 127593 XSI-conformant systems support this option.

### 127594 D.3.3 Configurable Limits

127595 Very few of the limits need to be increased for profiles. No profile can cite lower values.  
 127596 {POSIX2\_BC\_BASE\_MAX}  
 127597 {POSIX2\_BC\_DIM\_MAX}  
 127598 {POSIX2\_BC\_SCALE\_MAX}  
 127599 {POSIX2\_BC\_STRING\_MAX}  
 127600 No increase is anticipated for any of these *bc* values, except for very specialized applications  
 127601 involving huge numbers.  
 127602 {POSIX2\_COLL\_WEIGHTS\_MAX}  
 127603 Some natural languages with complex collation requirements require an increase from the  
 127604 default 2 to 4; no higher numbers are anticipated.  
 127605 {POSIX2\_EXPR\_NEST\_MAX}  
 127606 No increase is anticipated.  
 127607 {POSIX2\_LINE\_MAX}  
 127608 This number is much larger than most historical applications have been able to use. At some  
 127609 future time, applications may be rewritten to take advantage of even larger values.  
 127610 {POSIX2\_RE\_DUP\_MAX}  
 127611 No increase is anticipated.  
 127612 {POSIX2\_VERSION}  
 127613 This is actually not a limit, but a standard version stamp. Generally, a profile should specify  
 127614 XCU [Chapter 2](#) (on page 2319) by name in the normative references section, not this value.

### 127615 D.3.4 Configuration Options (System Interfaces)

127616 {NGROUPS\_MAX}  
 127617 A non-zero value indicates that the implementation supports supplementary groups.  
 127618 This option is needed where there is a large amount of shared use of files, but where a  
 127619 certain amount of protection is needed. Many profiles<sup>10</sup> are known to require this option; it  
 127620 should only be required if needed, but it should never be prohibited.  
 127621 \_POSIX\_ADVISORY\_INFO  
 127622 The system provides advisory information for file management.  
 127623 This option allows the application to specify advisory information that can be used to  
 127624 achieve better or even deterministic response time in file manager or input and output  
 127625 operations.  
 127626 \_POSIX\_ASYNCHRONOUS\_IO  
 127627 Support for asynchronous input and output is mandatory in POSIX.1-2008.

---

127628 10. There are no formally approved profiles of POSIX.1-2008 at the time of publication; the reference here is to various profiles generated by  
 127629 private bodies or governments.

127630 `_POSIX_BARRIERS`

127631 Support for barrier synchronization is mandatory in POSIX.1-2008.

127632 This facility allows efficient synchronization of multiple parallel threads in multi-processor  
127633 systems in which the operation is supported in part by the hardware architecture.127634 `_POSIX_CHOWN_RESTRICTED`127635 The system restricts the right to “give away” files to other users. It is mandatory that an  
127636 implementation be able to support this facility in POSIX.1-2008; however, it is recognized  
127637 that implementations need not enable the functionality by default.127638 Some applications expect that they can change the ownership of files in this way. It is  
127639 provided where either security or system account requirements cause this ability to be a  
127640 problem. It is also known to be specified in many profiles.127641 `_POSIX_CLOCK_SELECTION`

127642 Support for clock selection is mandatory in POSIX.1-2008.

127643 This facility allows applications to request a high resolution sleep in order to suspend a  
127644 thread during a relative time interval, or until an absolute time value, using the desired  
127645 clock. It also allows the application to select the clock used in a *pthread\_cond\_timedwait()*  
127646 function call.127647 `_POSIX_CPUTIME`

127648 The system supports the Process CPU-Time Clocks option.

127649 This option allows applications to use a new clock that measures the execution times of  
127650 processes or threads, and the possibility to create timers based upon these clocks, for  
127651 runtime detection (and treatment) of execution time overruns.127652 `_POSIX_FSYNC`

127653 The system supports file synchronization requests.

127654 This option was created to support historical systems that did not provide the feature.  
127655 Applications that are expecting guaranteed completion of their input and output operations  
127656 should require the `_POSIX_SYNC_IO` option. This option should never be prohibited.

127657 XSI-conformant systems support this option.

127658 `_POSIX_IPV6`

127659 The system supports facilities related to Internet Protocol Version 6 (IPv6).

127660 This option was created to allow systems to transition to IPv6.

127661 `_POSIX_JOB_CONTROL`

127662 Support for job control is mandatory in POSIX.1-2008.

127663 Most applications that use it can run when it is not present, although with a degraded level  
127664 of user convenience.127665 `_POSIX_MAPPED_FILES`

127666 Support for memory mapped files is mandatory in POSIX.1-2008.

127667 This facility provides for the mapping of regular files into the process address space.

127668 Both this facility and the Shared Memory Objects option provide shared access to memory  
127669 objects in the process address space. The *mmap()* and *munmap()* functions provide the  
127670 functionality of existing practice for mapping regular files. This functionality was deemed  
127671 unnecessary, if not inappropriate, for embedded systems applications and is expected to be  
127672 optional in subprofiles.



127673 `_POSIX_MEMLOCK`

127674 The system supports the locking of the address space.

127675 This option was created to support historical systems that did not provide the feature. It  
127676 should only be required if needed, but it should never be prohibited.127677 `_POSIX_MEMLOCK_RANGE`

127678 The system supports the locking of specific ranges of the address space.

127679 For applications that have well-defined sections that need to be locked and others that do  
127680 not, POSIX.1-2008 supports an optional set of functions to lock or unlock a range of process  
127681 addresses. The following are two reasons for having a means to lock down a specific range:

- 127682 1. An asynchronous event handler function that must respond to external events in a  
127683 deterministic manner such that page faults cannot be tolerated
- 127684 2. An input/output “buffer” area that is the target for direct-to-process I/O, and the  
127685 overhead of implicit locking and unlocking for each I/O call cannot be tolerated

127686 It should only be required if needed, but it should never be prohibited.

127687 `_POSIX_MEMORY_PROTECTION`

127688 Support for memory protection is mandatory in POSIX.1-2008.

127689 The provision of this facility typically imposes additional hardware requirements.

127690 `_POSIX_PRIORITIZED_IO`

127691 The system provides prioritization for input and output operations.

127692 The use of this option may interfere with the ability of the system to optimize input and  
127693 output throughput. It should only be required if needed, but it should never be prohibited.127694 `_POSIX_MESSAGE_PASSING`

127695 The system supports the passing of messages between processes.

127696 This option was created to support historical systems that did not provide the feature. The  
127697 functionality adds a high-performance XSI interprocess communication facility for local  
127698 communication. It should only be required if needed, but it should never be prohibited.127699 `_POSIX_MONOTONIC_CLOCK`

127700 The system supports the Monotonic Clock option.

127701 This option allows realtime applications to rely on a monotonically increasing clock that  
127702 does not jump backwards, and whose value does not change except for the regular ticking  
127703 of the clock.127704 `_POSIX_PRIORITY_SCHEDULING`

127705 The system provides priority-based process scheduling.

127706 Support of this option provides predictable scheduling behavior, allowing applications to  
127707 determine the order in which processes that are ready to run are granted access to a  
127708 processor. It should only be required if needed, but it should never be prohibited.127709 `_POSIX_REALTIME_SIGNALS`

127710 Support for realtime signals is mandatory in POSIX.1-2008.

127711 This facility provides prioritized, queued signals with associated data values.

127712 `_POSIX_REGEX`

127713 Support for regular expression facilities is mandatory in POSIX.1-2008.

- 127714 `_POSIX_SAVED_IDS`  
 127715 Support for this feature is mandatory in POSIX.1-2008.
- 127716 Certain classes of applications rely on it for proper operation, and there is no alternative  
 127717 short of giving the application root privileges on most implementations that did not provide  
 127718 `_POSIX_SAVED_IDS`.
- 127719 `_POSIX_SEMAPHORES`  
 127720 Support for counting semaphores is mandatory in POSIX.1-2008.
- 127721 `_POSIX_SHARED_MEMORY_OBJECTS`  
 127722 The system supports the mapping of shared memory objects into the process address space.
- 127723 Both this option and the Memory Mapped Files option provide shared access to memory  
 127724 objects in the process address space. The functions defined under this option provide the  
 127725 functionality of existing practice for shared memory objects. This functionality was deemed  
 127726 appropriate for embedded systems applications and, hence, is provided under this option.  
 127727 It should only be required if needed, but it should never be prohibited.
- 127728 `_POSIX_SHELL`  
 127729 Support for the *sh* utility command line interpreter is mandatory in POSIX.1-2008.
- 127730 `_POSIX_SPAWN`  
 127731 The system supports the spawn option.
- 127732 This option provides applications with an efficient mechanism to spawn execution of a new  
 127733 process.
- 127734 `_POSIX_SPINLOCKS`  
 127735 Support for spin locks is mandatory in POSIX.1-2008.
- 127736 This facility provides a simple and efficient synchronization mechanism for threads  
 127737 executing in multi-processor systems.
- 127738 `_POSIX_SPORADIC_SERVER`  
 127739 The system supports the sporadic server scheduling policy.
- 127740 This option provides applications with a new scheduling policy for scheduling aperiodic  
 127741 processes or threads in hard realtime applications.
- 127742 `_POSIX_SYNCHRONIZED_IO`  
 127743 The system supports guaranteed file synchronization.
- 127744 This option was created to support historical systems that did not provide the feature.  
 127745 Applications that are expecting guaranteed completion of their input and output operations  
 127746 should require this option, rather than the File Synchronization option. It should only be  
 127747 required if needed, but it should never be prohibited.
- 127748 `_POSIX_THREADS`  
 127749 Support for multiple threads of control within a single process is mandatory in  
 127750 POSIX.1-2008.
- 127751 `_POSIX_THREAD_ATTR_STACKADDR`  
 127752 The system supports specification of the stack address for a created thread.
- 127753 Applications may take advantage of support of this option for performance benefits, but  
 127754 dependence on this feature should be minimized. This option should never be prohibited.
- 127755 XSI-conformant systems support this option.

|        |                                                                                               |
|--------|-----------------------------------------------------------------------------------------------|
| 127756 | <code>_POSIX_THREAD_ATTR_STACKSIZE</code>                                                     |
| 127757 | The system supports specification of the stack size for a created thread.                     |
| 127758 | Applications may require this option in order to ensure proper execution, but such usage      |
| 127759 | limits portability and dependence on this feature should be minimized. It should only be      |
| 127760 | required if needed, but it should never be prohibited.                                        |
| 127761 | XSI-conformant systems support this option.                                                   |
| 127762 | <code>_POSIX_THREAD_PRIORITY_SCHEDULING</code>                                                |
| 127763 | The system provides priority-based thread scheduling.                                         |
| 127764 | Support of this option provides predictable scheduling behavior, allowing applications to     |
| 127765 | determine the order in which threads that are ready to run are granted access to a processor. |
| 127766 | It should only be required if needed, but it should never be prohibited.                      |
| 127767 | <code>_POSIX_THREAD_PRIO_INHERIT</code>                                                       |
| 127768 | The system provides mutual-exclusion operations with priority inheritance.                    |
| 127769 | Support of this option provides predictable scheduling behavior, allowing applications to     |
| 127770 | determine the order in which threads that are ready to run are granted access to a processor. |
| 127771 | It should only be required if needed, but it should never be prohibited.                      |
| 127772 | <code>_POSIX_THREAD_PRIO_PROTECT</code>                                                       |
| 127773 | The system supports a priority ceiling emulation protocol for mutual-exclusion operations.    |
| 127774 | Support of this option provides predictable scheduling behavior, allowing applications to     |
| 127775 | determine the order in which threads that are ready to run are granted access to a processor. |
| 127776 | It should only be required if needed, but it should never be prohibited.                      |
| 127777 | <code>_POSIX_THREAD_PROCESS_SHARED</code>                                                     |
| 127778 | The system provides shared access among multiple processes to synchronization objects.        |
| 127779 | This option was created to support historical systems that did not provide the feature. It    |
| 127780 | should only be required if needed, but it should never be prohibited.                         |
| 127781 | XSI-conformant systems support this option.                                                   |
| 127782 | <code>_POSIX_THREAD_SAFE_FUNCTIONS</code>                                                     |
| 127783 | Support for thread-safe functions is mandatory in POSIX.1-2008.                               |
| 127784 | <code>_POSIX_THREAD_SPORADIC_SERVER</code>                                                    |
| 127785 | The system supports the thread sporadic server scheduling policy.                             |
| 127786 | Support for this option provides applications with a new scheduling policy for scheduling     |
| 127787 | aperiodic threads in hard realtime applications.                                              |
| 127788 | <code>_POSIX_TIMEOUTS</code>                                                                  |
| 127789 | Support for timeouts for some blocking services is mandatory in POSIX.1-2008.                 |
| 127790 | <code>_POSIX_TIMERS</code>                                                                    |
| 127791 | Support for higher resolution clocks with multiple timers per process is mandatory in         |
| 127792 | POSIX.1-2008.                                                                                 |
| 127793 | This facility is appropriate for applications requiring higher resolution timestamps or       |
| 127794 | needing to control the timing of multiple activities.                                         |
| 127795 | <code>_POSIX_TRACE</code>                                                                     |
| 127796 | The system supports the Trace option.                                                         |
| 127797 | This option was created to allow applications to perform tracing.                             |

127798 `_POSIX_TRACE_EVENT_FILTER`  
 127799 The system supports the Trace Event Filter option.  
 127800 This option is dependent on support of the Trace option.  
 127801 `_POSIX_TRACE_INHERIT`  
 127802 The system supports the Trace Inherit option.  
 127803 This option is dependent on support of the Trace option.  
 127804 `_POSIX_TRACE_LOG`  
 127805 The system supports the Trace Log option.  
 127806 This option is dependent on support of the Trace option.  
 127807 `_POSIX_TYPED_MEMORY_OBJECTS`  
 127808 The system supports the Typed Memory Objects option.  
 127809 This option was created to allow realtime applications to access different kinds of physical  
 127810 memory, and allow processes in these applications to share portions of this memory.

### 127811 **D.3.5 Configurable Limits**

127812 In general, the configurable limits in the `<limits.h>` header defined in the Base Definitions  
 127813 volume of POSIX.1-2008 have been set to minimal values; many applications or implementations  
 127814 may require larger values. No profile can cite lower values.

127815 `{AIO_LISTIO_MAX}`  
 127816 The current minimum is likely to be inadequate for most applications. It is expected that  
 127817 this value will be increased by profiles requiring support for list input and output  
 127818 operations.

127819 `{AIO_MAX}`  
 127820 The current minimum is likely to be inadequate for most applications. It is expected that  
 127821 this value will be increased by profiles requiring support for asynchronous input and  
 127822 output operations.

127823 `{AIO_PRIO_DELTA_MAX}`  
 127824 The functionality associated with this limit is needed only by sophisticated applications. It  
 127825 is not expected that this limit would need to be increased under a general-purpose profile.

127826 `{ARG_MAX}`  
 127827 The current minimum is likely to need to be increased for profiles, particularly as larger  
 127828 amounts of information are passed through the environment. Many implementations are  
 127829 believed to support larger values.

127830 `{CHILD_MAX}`  
 127831 The current minimum is suitable only for systems where a single user is not running  
 127832 applications in parallel. It is significantly too low for any system also requiring windows,  
 127833 and if `_POSIX_JOB_CONTROL` is specified, it should be raised.

127834 `{CLOCKRES_MIN}`  
 127835 It is expected that profiles will require a finer granularity clock, perhaps as fine as 1  $\mu$ s,  
 127836 represented by a value of 1 000 for this limit.

127837 `{DELAYTIMER_MAX}`  
 127838 It is believed that most implementations will provide larger values.

|        |                                                                                                 |
|--------|-------------------------------------------------------------------------------------------------|
| 127839 | {LINK_MAX}                                                                                      |
| 127840 | For most applications and usage, the current minimum is adequate. Many implementations          |
| 127841 | have a much larger value, but this should not be used as a basis for raising the value unless   |
| 127842 | the applications to be used require it.                                                         |
| 127843 | {LOGIN_NAME_MAX}                                                                                |
| 127844 | This is not actually a limit, but an implementation parameter. No profile should impose a       |
| 127845 | requirement on this value.                                                                      |
| 127846 | {MAX_CANON}                                                                                     |
| 127847 | For most purposes, the current minimum is adequate. Unless high-speed burst serial              |
| 127848 | devices are used, it should be left as is.                                                      |
| 127849 | {MAX_INPUT}                                                                                     |
| 127850 | See {MAX_CANON}.                                                                                |
| 127851 | {MQ_OPEN_MAX}                                                                                   |
| 127852 | The current minimum should be adequate for most profiles.                                       |
| 127853 | {MQ_PRIO_MAX}                                                                                   |
| 127854 | The current minimum corresponds to the required number of process scheduling priorities.        |
| 127855 | Many realtime practitioners believe that the number of message priority levels ought to be      |
| 127856 | the same as the number of execution scheduling priorities.                                      |
| 127857 | {NAME_MAX}                                                                                      |
| 127858 | Many implementations now support larger values, and many applications and users                 |
| 127859 | assume that larger names can be used. Many existing profiles also specify a larger value.       |
| 127860 | Specifying this value will reduce the number of conforming implementations, although this       |
| 127861 | might not be a significant consideration over time. Values greater than 255 should not be       |
| 127862 | required.                                                                                       |
| 127863 | {NGROUPS_MAX}                                                                                   |
| 127864 | The value selected will typically be 8 or larger.                                               |
| 127865 | {OPEN_MAX}                                                                                      |
| 127866 | The historically common value for this has been 20. Many implementations support larger         |
| 127867 | values. If applications that use larger values are anticipated, an appropriate value should be  |
| 127868 | specified.                                                                                      |
| 127869 | {PAGESIZE}                                                                                      |
| 127870 | This is not actually a limit, but an implementation parameter. No profile should impose a       |
| 127871 | requirement on this value.                                                                      |
| 127872 | {PATH_MAX}                                                                                      |
| 127873 | Historically, the minimum has been either 1024 or indefinite, depending on the                  |
| 127874 | implementation. Few applications actually require values larger than 256, but some users        |
| 127875 | may create file hierarchies that must be accessed with longer paths. This value should only     |
| 127876 | be changed if there is a clear requirement.                                                     |
| 127877 | {PIPE_BUF}                                                                                      |
| 127878 | The current minimum is adequate for most applications. Historically, it has been larger. If     |
| 127879 | applications that write single transactions larger than this are anticipated, it should be      |
| 127880 | increased. Applications that write lines of text larger than this probably do not need it       |
| 127881 | increased, as the text line is delimited by a <newline>.                                        |
| 127882 | {POSIX_VERSION}                                                                                 |
| 127883 | This is actually not a limit, but a standard version stamp. Generally, a profile should specify |
| 127884 | POSIX.1-2008 by a name in the normative references section, not this value.                     |

|        |                                                                                               |
|--------|-----------------------------------------------------------------------------------------------|
| 127885 | {PTHREAD_DESTRUCTOR_ITERATIONS}                                                               |
| 127886 | It is unlikely that applications will need larger values to avoid loss of memory resources.   |
| 127887 | {PTHREAD_KEYS_MAX}                                                                            |
| 127888 | The current value should be adequate for most profiles.                                       |
| 127889 | {PTHREAD_STACK_MIN}                                                                           |
| 127890 | This should not be treated as an actual limit, but as an implementation parameter. No         |
| 127891 | profile should impose a requirement on this value.                                            |
| 127892 | {PTHREAD_THREADS_MAX}                                                                         |
| 127893 | It is believed that most implementations will provide larger values.                          |
| 127894 | {RTSIG_MAX}                                                                                   |
| 127895 | The current limit was chosen so that the set of POSIX.1 signal numbers can fit within a       |
| 127896 | 32-bit field. It is recognized that most existing implementations define many more signals    |
| 127897 | than are specified in POSIX.1 and, in fact, many implementations have already exceeded 32     |
| 127898 | signals (including the “null signal”). Support of {_POSIX_RTSIG_MAX} additional signals       |
| 127899 | may push some implementations over the single 32-bit word line, but is unlikely to push       |
| 127900 | any implementations that are already over that line beyond the 64 signal line.                |
| 127901 | {SEM_NSEMS_MAX}                                                                               |
| 127902 | The current value should be adequate for most profiles.                                       |
| 127903 | {SEM_VALUE_MAX}                                                                               |
| 127904 | The current value should be adequate for most profiles.                                       |
| 127905 | {SSIZE_MAX}                                                                                   |
| 127906 | This limit reflects fundamental hardware characteristics (the size of an integer), and should |
| 127907 | not be specified unless it is clearly required. Extreme care should be taken to assure that   |
| 127908 | any value that might be specified does not unnecessarily eliminate implementations            |
| 127909 | because of accidents of hardware design.                                                      |
| 127910 | {STREAM_MAX}                                                                                  |
| 127911 | This limit is very closely related to {OPEN_MAX}. It should never be larger than              |
| 127912 | {OPEN_MAX}, but could reasonably be smaller for application areas where most files are        |
| 127913 | not accessed through <i>stdio</i> . Some implementations may limit {STREAM_MAX} to 20 but     |
| 127914 | allow {OPEN_MAX} to be considerably larger. Such implementations should be allowed for        |
| 127915 | if the applications permit.                                                                   |
| 127916 | {TIMER_MAX}                                                                                   |
| 127917 | The current limit should be adequate for most profiles, but it may need to be larger for      |
| 127918 | applications with a large number of asynchronous operations.                                  |
| 127919 | {TTY_NAME_MAX}                                                                                |
| 127920 | This is not actually a limit, but an implementation parameter. No profile should impose a     |
| 127921 | requirement on this value.                                                                    |
| 127922 | {TZNAME_MAX}                                                                                  |
| 127923 | The minimum has been historically adequate, but if longer timezone names are anticipated      |
| 127924 | (particularly such values as UTC−1), this should be increased.                                |

**D.3.6 Optional Behavior**

127925  
127926  
127927  
127928  
127929  
127930  
127931  
127932

In POSIX.1-2008, there are no instances of the terms unspecified, undefined, implementation-defined, or with the verbs “may” or “need not”, that the standard developers anticipate or sanction as suitable for profile or test method citation. All of these are merely warnings to conforming applications to avoid certain areas that can vary from system to system, and even over time on the same system. In many cases, these terms are used explicitly to support extensions, but profiles should not anticipate and require such extensions; future versions of this standard may do so.





127933

# *Rationale (Informative)*

127934

## **Part E:**

127935

## **Subprofiling Considerations**

127936

*The Open Group*

127937

*The Institute of Electrical and Electronics Engineers, Inc.*



## Subprofiling Considerations (Informative)

127940  
127941  
127942  
127943

This section contains further information to satisfy the requirement that the project scope enable subprofiling of POSIX.1-2008. The approach taken is to include a general requirement in normative text regarding subprofiling and to include an informative section (here) containing a proposed set of subprofiling options.

### 127944 E.1 Subprofiling Option Groups

127945  
127946  
127947  
127948

The following Option Groups<sup>11</sup> are defined to support profiling. Systems claiming support to POSIX.1-2008 need not implement these options apart from the requirements stated in XBD [Section 2.1.3](#) (on page 16). These Option Groups allow profiles to subset the System Interfaces volume of POSIX.1-2008 by collecting sets of related functions.

127949  
127950  
127951

POSIX\_ASYNCHRONOUS\_IO: Asynchronous Input and Output Functions

*aio\_cancel()*, *aio\_error()*, *aio\_fsync()*, *aio\_read()*, *aio\_return()*, *aio\_suspend()*, *aio\_write()*,  
*lio\_listio()*

127952  
127953

POSIX\_BARRIERS: Barriers

*pthread\_barrier\_destroy()*, *pthread\_barrier\_init()*, *pthread\_barrier\_wait()*, *pthread\_barrierattr()*

127954  
127955

POSIX\_C\_LANG\_JUMP: Jump Functions

*longjmp()*, *setjmp()*

127956

POSIX\_C\_LANG\_MATH: Maths Library

127957  
127958  
127959  
127960  
127961  
127962  
127963  
127964  
127965  
127966  
127967  
127968  
127969  
127970  
127971  
127972  
127973  
127974  
127975  
127976

*acos()*, *acosf()*, *acosh()*, *acoshf()*, *acoshl()*, *acosl()*, *asin()*, *asinf()*, *asinh()*, *asinhf()*, *asinhf()*, *asinhl()*,  
*asinl()*, *atan()*, *atan2()*, *atan2f()*, *atan2l()*, *atanf()*, *atanh()*, *atanhf()*, *atanhl()*, *atanl()*, *cabs()*,  
*cabsf()*, *cabsl()*, *cacos()*, *cacosf()*, *cacosh()*, *cacoshf()*, *cacoshl()*, *cacosl()*, *carg()*, *cargf()*, *cargl()*,  
*casin()*, *casinf()*, *casinh()*, *casinhf()*, *casinhf()*, *casinl()*, *catan()*, *catanf()*, *catanh()*, *catanhf()*,  
*catanhf()*, *catanhl()*, *catanl()*, *cbrt()*, *cbrtf()*, *cbrtl()*, *ccos()*, *ccosf()*, *ccosh()*, *ccoshf()*, *ccoshl()*, *ccosl()*,  
*ceil()*, *ceilf()*, *ceilf()*, *cexp()*, *cexpf()*, *cexpl()*, *cimag()*, *cimagf()*, *cimagl()*, *clog()*, *clogf()*, *clogl()*,  
*conj()*, *conjf()*, *conjl()*, *copysign()*, *copysignf()*, *copysignl()*, *cos()*, *cosf()*, *cosh()*, *coshf()*,  
*coshl()*, *cosl()*, *cpow()*, *cpowf()*, *cpowl()*, *cproj()*, *cprojf()*, *cprojl()*, *creal()*, *crealf()*, *creall()*,  
*csin()*, *csinf()*, *csinh()*, *csinhf()*, *csinhf()*, *csinl()*, *csqrt()*, *csqrtf()*, *csqrtl()*, *ctan()*, *ctanf()*,  
*ctanh()*, *ctanhf()*, *ctanhf()*, *ctanl()*, *erf()*, *erfc()*, *erfcf()*, *erfcl()*, *erff()*, *erfl()*, *exp()*, *exp2()*,  
*exp2f()*, *exp2l()*, *expf()*, *expl()*, *expm1()*, *expm1f()*, *expm1l()*, *fabs()*, *fabsf()*, *fabsf()*, *fdim()*,  
*fdimf()*, *fdiml()*, *floor()*, *floorf()*, *floorl()*, *fma()*, *fmaf()*, *fmal()*, *fmax()*, *fmaxf()*, *fmaxl()*, *fmin()*,  
*fminf()*, *fminl()*, *fmod()*, *fmodf()*, *fmodl()*, *fpclassify()*, *frexp()*, *frexpf()*, *frexpl()*, *hypot()*,  
*hypotf()*, *hypotl()*, *ilogb()*, *ilogbf()*, *ilogbl()*, *isfinite()*, *isgreater()*, *isgreaterequal()*, *isinf()*,  
*isless()*, *islessequal()*, *islessgreater()*, *isnan()*, *isnormal()*, *isunordered()*, *ldexp()*, *ldexpf()*,  
*ldexpl()*, *lgamma()*, *lgammaf()*, *lgammal()*, *llrint()*, *llrintf()*, *llrintl()*, *llround()*, *llroundf()*,  
*llroundl()*, *log()*, *log10()*, *log10f()*, *log10l()*, *log1p()*, *log1pf()*, *log1pl()*, *log2()*, *log2f()*, *log2l()*,  
*logb()*, *logbf()*, *logbl()*, *logf()*, *logl()*, *lrint()*, *lrintf()*, *lrintl()*, *lround()*, *lroundf()*, *lroundl()*,  
*modf()*, *modff()*, *modfl()*, *nan()*, *nanf()*, *nanl()*, *nearbyint()*, *nearbyintf()*, *nearbyintl()*,  
*nextafter()*, *nextafterf()*, *nextafterl()*, *nexttoward()*, *nexttowardf()*, *nexttowardl()*, *pow()*, *powf()*,

127977 11. These are modeled on the Units of Functionality from IEEE Std 1003.13-1998.

127978 *powl()*, *remainder()*, *remainderf()*, *remainderl()*, *remquo()*, *remquoof()*, *remquol()*, *rint()*, *rintf()*,  
 127979 *rintl()*, *round()*, *roundf()*, *roundl()*, *scalbln()*, *scalblnf()*, *scalblnl()*, *scalbn()*, *scalbnf()*,  
 127980 *scalbnl()*, *signbit()*, *sin()*, *sinf()*, *sinh()*, *sinhf()*, *sinhl()*, *sinl()*, *sqrt()*, *sqrtf()*, *sqrtl()*, *tan()*,  
 127981 *tanf()*, *tanh()*, *tanhf()*, *tanhf()*, *tanl()*, *tgamma()*, *tgammaf()*, *tgammaf()*, *trunc()*, *truncf()*,  
 127982 *truncl()*

#### 127983 POSIX\_C\_LANG\_SUPPORT: General ISO C Library

127984 *abs()*, *asctime()*, *atof()*, *atoi()*, *atol()*, *atoll()*, *bsearch()*, *calloc()*, *ctime()*, *difftime()*, *div()*,  
 127985 *feclearexcept()*, *fegetenv()*, *fegetexceptflag()*, *fegetround()*, *fehldexcept()*, *feraiseexcept()*,  
 127986 *fesetenv()*, *fesetexceptflag()*, *fesetround()*, *fetestexcept()*, *feupdateenv()*, *free()*, *gmtime()*,  
 127987 *imaxabs()*, *imaxdiv()*, *isalnum()*, *isalpha()*, *isblank()*, *iscntrl()*, *isdigit()*, *isgraph()*, *islower()*,  
 127988 *isprint()*, *ispunct()*, *isspace()*, *isupper()*, *isxdigit()*, *labs()*, *ldiv()*, *llabs()*, *lldiv()*, *localeconv()*,  
 127989 *localtime()*, *malloc()*, *memchr()*, *memcmp()*, *memcpy()*, *memmove()*, *memset()*, *mktime()*,  
 127990 *qsort()*, *rand()*, *realloc()*, *setlocale()*, *snprintf()*, *sprintf()*, *srand()*, *sscanf()*, *strcat()*, *strchr()*,  
 127991 *strcmp()*, *strcoll()*, *strcpy()*, *strcspn()*, *strerror()*, *strtime()*, *strlen()*, *strncat()*, *strncmp()*,  
 127992 *strncpy()*, *strpbrk()*, *strrchr()*, *strspn()*, *strstr()*, *strtod()*, *strtof()*, *strtoimax()*, *strtok()*, *strtol()*,  
 127993 *strtold()*, *strtoll()*, *strtoul()*, *strtoull()*, *strtoumax()*, *strxfrm()*, *time()*, *tolower()*, *toupper()*,  
 127994 *tzname()*, *tzset()*, *va\_arg()*, *va\_copy()*, *va\_end()*, *va\_start()*, *vsprintf()*, *vsprintf()*, *vsscanf()*

#### 127995 POSIX\_C\_LANG\_SUPPORT\_R: Thread-Safe General ISO C Library

127996 *asctime\_r()*, *ctime\_r()*, *gmtime\_r()*, *localtime\_r()*, *rand\_r()*, *strerror\_r()*, *strtok\_r()*

#### 127997 POSIX\_C\_LANG\_WIDE\_CHAR: Wide-Character ISO C Library

127998 *btowc()*, *iswalnum()*, *iswalalpha()*, *iswblank()*, *iswcntrl()*, *iswctype()*, *iswdigit()*, *iswgraph()*,  
 127999 *iswlower()*, *iswprint()*, *iswpunct()*, *iswspace()*, *iswupper()*, *iswxdigit()*, *mblen()*, *mbrlen()*,  
 128000 *mbrtowc()*, *mbsinit()*, *mbsrtowcs()*, *mbstowcs()*, *mbtowc()*, *swprintf()*, *swscanf()*, *towctrans()*,  
 128001 *towlower()*, *towupper()*, *vwprintf()*, *vwscanf()*, *wcrtomb()*, *wcscat()*, *wcschr()*, *wcscmp()*,  
 128002 *wcscoll()*, *wcscpy()*, *wcscspn()*, *wcsftime()*, *wcslen()*, *wcsncat()*, *wcsncmp()*, *wcsncpy()*,  
 128003 *wcspbrk()*, *wcsrchr()*, *wcrtombs()*, *wcspn()*, *wcsstr()*, *wcstod()*, *wcstof()*, *wcstoimax()*,  
 128004 *wcstok()*, *wcstol()*, *wcstold()*, *wcstoll()*, *wcstombs()*, *wcstoul()*, *wcstoull()*, *wcstoumax()*,  
 128005 *wcsxfrm()*, *wctob()*, *wctomb()*, *wctrans()*, *wctype()*, *wmemchr()*, *wmemcmp()*, *wmemcpy()*,  
 128006 *wmemmove()*, *wmemset()*

#### 128007 POSIX\_C\_LANG\_WIDE\_CHAR\_EXT: Extended Wide-Character ISO C Library

128008 *mbsnrtowcs()*, *wcpcpy()*, *wcpncpy()*, *wcscasecmp()*, *wcsdup()*, *wcsncasecmp()*, *wcsnlen()*,  
 128009 *wcsnrtombs()*

#### 128010 POSIX\_C\_LIB\_EXT: General C Library Extension

128011 *fnmatch()*, *getopt()*, *getsubopt()*, *optarg*, *opterr*, *optind*, *optopt*, *stpcpy()*, *stpncpy()*, *strcasecmp()*,  
 128012 *strdup()*, *strfmon()*, *strncasecmp()*, *strndup()*, *strnlen()*

#### 128013 POSIX\_CLOCK\_SELECTION: Clock Selection

128014 *clock\_nanosleep()*, *pthread\_condattr\_getclock()*, *pthread\_condattr\_setclock()*

#### 128015 POSIX\_DEVICE\_IO: Device Input and Output

128016 *FD\_CLR()*, *FD\_ISSET()*, *FD\_SET()*, *FD\_ZERO()*, *clearerr()*, *close()*, *fclose()*, *fdopen()*, *feof()*,  
 128017 *ferror()*, *fflush()*, *fgetc()*, *fgets()*, *fileno()*, *fopen()*, *fprintf()*, *fputc()*, *fputs()*, *fread()*, *freopen()*,  
 128018 *fscanf()*, *fwrite()*, *getc()*, *getchar()*, *gets()*, *open()*, *perror()*, *poll()*, *printf()*, *pread()*, *pselect()*,  
 128019 *putc()*, *putchar()*, *puts()*, *pwrite()*, *read()*, *scanf()*, *select()*, *setbuf()*, *setvbuf()*, *stderr*, *stdin*,  
 128020 *stdout*, *ungetc()*, *vfprintf()*, *vfsanf()*, *vprintf()*, *vscanf()*, *write()*

#### 128021 POSIX\_DEVICE\_IO\_EXT: Extended Device Input and Output

128022 *dprintf()*, *fmemopen()*, *open\_memstream()*, *vdprintf()*

#### 128023 POSIX\_DEVICE\_SPECIFIC: General Terminal

128024 *cfgetispeed()*, *cfgetospeed()*, *cfsetispeed()*, *cfsetospeed()*, *ctermid()*, *isatty()*, *tcdrain()*, *tcflow()*,  
 128025 *tcflush()*, *tcgetattr()*, *tcsendbreak()*, *tcsetattr()*, *ttyname()*

|        |                                                                                                                                                                                                   |
|--------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 128026 | POSIX_DEVICE_SPECIFIC_R: Thread-Safe General Terminal                                                                                                                                             |
| 128027 | <i>ttyname_r()</i>                                                                                                                                                                                |
| 128028 | POSIX_DYNAMIC_LINKING: Dynamic Linking                                                                                                                                                            |
| 128029 | <i>dlclose()</i> , <i>dlderror()</i> , <i>dlopen()</i> , <i>dlsym()</i>                                                                                                                           |
| 128030 | POSIX_FD_MGMT: File Descriptor Management                                                                                                                                                         |
| 128031 | <i>dup()</i> , <i>dup2()</i> , <i>fcntl()</i> , <i>fgetpos()</i> , <i>fseek()</i> , <i>fseeko()</i> , <i>fsetpos()</i> , <i>ftell()</i> , <i>ftello()</i> , <i>ftruncate()</i> , <i>lseek()</i> , |
| 128032 | <i>rewind()</i>                                                                                                                                                                                   |
| 128033 | POSIX_FIFO: FIFO                                                                                                                                                                                  |
| 128034 | <i>mkfifo()</i>                                                                                                                                                                                   |
| 128035 | POSIX_FIFO_FD: FIFO File Descriptor Routines                                                                                                                                                      |
| 128036 | <i>mkfifoat()</i> , <i>mknodat()</i>                                                                                                                                                              |
| 128037 | POSIX_FILE_ATTRIBUTES: File Attributes                                                                                                                                                            |
| 128038 | <i>chmod()</i> , <i>chown()</i> , <i>fchmod()</i> , <i>fchown()</i> , <i>umask()</i>                                                                                                              |
| 128039 | POSIX_FILE_ATTRIBUTES_FD: File Attributes File Descriptor Routines                                                                                                                                |
| 128040 | <i>fchmodat()</i> , <i>fchownat()</i>                                                                                                                                                             |
| 128041 | POSIX_FILE_LOCKING: Thread-Safe Stdio Locking                                                                                                                                                     |
| 128042 | <i>flockfile()</i> , <i>ftrylockfile()</i> , <i>funlockfile()</i> , <i>getc_unlocked()</i> , <i>getchar_unlocked()</i> , <i>putc_unlocked()</i> ,                                                 |
| 128043 | <i>putchar_unlocked()</i>                                                                                                                                                                         |
| 128044 | POSIX_FILE_SYSTEM: File System                                                                                                                                                                    |
| 128045 | <i>access()</i> , <i>chdir()</i> , <i>closedir()</i> , <i>creat()</i> , <i>fchdir()</i> , <i>fpathconf()</i> , <i>fstat()</i> , <i>fstatvfs()</i> , <i>getcwd()</i> , <i>link()</i> ,             |
| 128046 | <i>mkdir()</i> , <i>mkstemp()</i> , <i>opendir()</i> , <i>pathconf()</i> , <i>readdir()</i> , <i>remove()</i> , <i>rename()</i> , <i>rewinddir()</i> , <i>rmdir()</i> ,                           |
| 128047 | <i>stat()</i> , <i>statvfs()</i> , <i>tmpfile()</i> , <i>tmpnam()</i> , <i>truncate()</i> , <i>unlink()</i> , <i>utime()</i>                                                                      |
| 128048 | POSIX_FILE_SYSTEM_EXT: File System Extensions                                                                                                                                                     |
| 128049 | <i>alphasort()</i> , <i>dirfd()</i> , <i>getdelim()</i> , <i>getline()</i> , <i>mkdtemp()</i> , <i>scandir()</i>                                                                                  |
| 128050 | POSIX_FILE_SYSTEM_FD: File System File Descriptor Routines                                                                                                                                        |
| 128051 | <i>faccessat()</i> , <i>fdopendir()</i> , <i>fstatat()</i> , <i>linkat()</i> , <i>mkdirat()</i> , <i>openat()</i> , <i>renameat()</i> , <i>unlinkat()</i> ,                                       |
| 128052 | <i>utimensat()</i>                                                                                                                                                                                |
| 128053 | POSIX_FILE_SYSTEM_GLOB: File System Glob Expansion                                                                                                                                                |
| 128054 | <i>glob()</i> , <i>globfree()</i>                                                                                                                                                                 |
| 128055 | POSIX_FILE_SYSTEM_R: Thread-Safe File System                                                                                                                                                      |
| 128056 | <i>readdir_r()</i>                                                                                                                                                                                |
| 128057 | POSIX_I18N: Internationalization                                                                                                                                                                  |
| 128058 | <i>catclose()</i> , <i>catgets()</i> , <i>catopen()</i> , <i>iconv()</i> , <i>iconv_close()</i> , <i>iconv_open()</i> , <i>nl_langinfo()</i>                                                      |
| 128059 | POSIX_JOB_CONTROL: Job Control                                                                                                                                                                    |
| 128060 | <i>setpgid()</i> , <i>tcgetpgrp()</i> , <i>tcsetpgrp()</i> , <i>tcgetsid()</i>                                                                                                                    |
| 128061 | POSIX_MAPPED_FILES: Memory Mapped Files                                                                                                                                                           |
| 128062 | <i>mmap()</i> , <i>munmap()</i>                                                                                                                                                                   |
| 128063 | POSIX_MEMORY_PROTECTION: Memory Protection                                                                                                                                                        |
| 128064 | <i>mprotect()</i>                                                                                                                                                                                 |
| 128065 | POSIX_MULTI_CONCURRENT_LOCALES: Multiple Concurrent Locales                                                                                                                                       |
| 128066 | <i>duplocale()</i> , <i>freelocale()</i> , <i>isalnum_l()</i> , <i>isalpha_l()</i> , <i>isblank_l()</i> , <i>iscntrl_l()</i> , <i>isdigit_l()</i> , <i>isgraph_l()</i> ,                          |
| 128067 | <i>islower_l()</i> , <i>isprint_l()</i> , <i>ispunct_l()</i> , <i>isspace_l()</i> , <i>isupper_l()</i> , <i>iswalnum_l()</i> , <i>iswalalpha_l()</i> ,                                            |
| 128068 | <i>iswblank_l()</i> , <i>iswcntrl_l()</i> , <i>iswctype_l()</i> , <i>iswdigit_l()</i> , <i>iswgraph_l()</i> , <i>iswlower_l()</i> , <i>iswprint_l()</i> ,                                         |
| 128069 | <i>iswpunct_l()</i> , <i>iswspace_l()</i> , <i>iswupper_l()</i> , <i>iswxdigit_l()</i> , <i>isxdigit_l()</i> , <i>newlocale()</i> , <i>strcasecmp_l()</i> ,                                       |

128070 *strcoll\_l()*, *strfmon\_l()*, *strncasecmp\_l()*, *strxfrm\_l()*, *tolower\_l()*, *toupper\_l()*, *towctrans\_l()*,  
 128071 *towlower()*, *towupper()*, *uselocale()*, *wscasecmp\_l()*, *wscoll\_l()*, *wcsncasecmp\_l()*, *wcsxfrm\_l()*,  
 128072 *wctrans\_l()*, *wctype\_l()*

128073 POSIX\_MULTI\_PROCESS: Multiple Processes  
 128074 *\_Exit()*, *\_exit()*, *assert()*, *atexit()*, *clock()*, *execl()*, *execle()*, *execlp()*, *execv()*, *execve()*, *execvp()*,  
 128075 *exit()*, *fork()*, *getpgrp()*, *getpgid()*, *getpid()*, *getppid()*, *getsid()*, *setsid()*, *sleep()*, *times()*, *wait()*,  
 128076 *waitid()*, *waitpid()*

128077 POSIX\_MULTI\_PROCESS\_FD: Multiple Processes File Descriptor Routines  
 128078 *fxexecve()*

128079 POSIX\_NETWORKING: Networking  
 128080 *accept()*, *bind()*, *connect()*, *endhostent()*, *endnetent()*, *endprotoent()*, *endservent()*,  
 128081 *freeaddrinfo()*, *gai\_strerror()*, *getaddrinfo()*, *gethostent()*, *gethostname()*, *getnameinfo()*,  
 128082 *getnetbyaddr()*, *getnetbyname()*, *getnetent()*, *getpeername()*, *getprotobyname()*,  
 128083 *getprotobynumber()*, *getprotoent()*, *getservbyname()*, *getservbyport()*, *getservent()*,  
 128084 *getsockname()*, *getsockopt()*, *htonl()*, *htons()*, *if\_freenameindex()*, *if\_indextoname()*,  
 128085 *if\_nameindex()*, *if\_nametoindex()*, *inet\_addr()*, *inet\_ntoa()*, *inet\_ntop()*, *inet\_pton()*, *listen()*,  
 128086 *ntohl()*, *ntohs()*, *recv()*, *recvfrom()*, *recvmsg()*, *send()*, *sendmsg()*, *sendto()*, *sethostent()*,  
 128087 *setnetent()*, *setprotoent()*, *setservent()*, *setsockopt()*, *shutdown()*, *socket()*, *socketatmark()*,  
 128088 *socketpair()*

128089 POSIX\_PIPE: Pipe  
 128090 *pipe()*

128091 POSIX\_ROBUST\_MUTEXES: Robust Mutexes  
 128092 *pthread\_mutex\_consistent()*, *pthread\_mutexattr\_getrobust()*, *pthread\_mutexattr\_setrobust()*

128093 POSIX\_REALTIME\_SIGNALS: Realtime Signals  
 128094 *sigqueue()*, *sigtimedwait()*, *sigwaitinfo()*

128095 POSIX\_REGEX: Regular Expressions  
 128096 *regcomp()*, *regerror()*, *regexec()*, *regfree()*

128097 POSIX\_RW\_LOCKS: Reader Writer Locks  
 128098 *pthread\_rwlock\_destroy()*, *pthread\_rwlock\_init()*, *pthread\_rwlock\_rdlock()*,  
 128099 *pthread\_rwlock\_timedrdlock()*, *pthread\_rwlock\_timedwrlock()*, *pthread\_rwlock\_tryrdlock()*,  
 128100 *pthread\_rwlock\_trywrlock()*, *pthread\_rwlock\_unlock()*, *pthread\_rwlock\_wrlock()*,  
 128101 *pthread\_rwlockattr\_destroy()*, *pthread\_rwlockattr\_init()*, *pthread\_rwlockattr\_getpshared()*,  
 128102 *pthread\_rwlockattr\_setpshared()*

128103 POSIX\_SEMAPHORES: Semaphores  
 128104 *sem\_close()*, *sem\_destroy()*, *sem\_getvalue()*, *sem\_init()*, *sem\_open()*, *sem\_post()*,  
 128105 *sem\_timedwait()*, *sem\_trywait()*, *sem\_unlink()*, *sem\_wait()*

128106 POSIX\_SHELL\_FUNC: Shell and Utilities  
 128107 *pclose()*, *popen()*, *system()*, *wordexp()*, *wordfree()*

128108 POSIX\_SIGNAL\_JUMP: Signal Jump Functions  
 128109 *siglongjmp()*, *sigsetjmp()*

128110 POSIX\_SIGNALS: Signals  
 128111 *abort()*, *alarm()*, *kill()*, *pause()*, *raise()*, *sigaction()*, *sigaddset()*, *sigdelset()*, *sigemptyset()*,  
 128112 *sigfillset()*, *sigismember()*, *signal()*, *sigpending()*, *sigprocmask()*, *sigsuspend()*, *sigwait()*

128113 POSIX\_SIGNALS\_EXT: Extended Signals  
 128114 *psignal()*, *psiginfo()*, *strsignal()*

|        |                                                                                                                                                                            |
|--------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 128115 | POSIX_SINGLE_PROCESS: Single Process                                                                                                                                       |
| 128116 | <i>confstr()</i> , <i>environ</i> , <i>errno</i> , <i>getenv()</i> , <i>setenv()</i> , <i>sysconf()</i> , <i>uname()</i> , <i>unsetenv()</i>                               |
| 128117 | POSIX_SPIN_LOCKS: Spin Locks                                                                                                                                               |
| 128118 | <i>pthread_spin_destroy()</i> , <i>pthread_spin_init()</i> , <i>pthread_spin_lock()</i> , <i>pthread_spin_trylock()</i> ,                                                  |
| 128119 | <i>pthread_spin_unlock()</i>                                                                                                                                               |
| 128120 | POSIX_SYMBOLIC_LINKS: Symbolic Links                                                                                                                                       |
| 128121 | <i>lchown()</i> , <sup>12</sup> <i>lstat()</i> , <i>readlink()</i> , <i>symlink()</i>                                                                                      |
| 128122 | POSIX_SYMBOLIC_LINKS_FD: Symbolic Links File Descriptor Routines                                                                                                           |
| 128123 | <i>readlinkat()</i> , <i>symlinkat()</i>                                                                                                                                   |
| 128124 | POSIX_SYSTEM_DATABASE: System Database                                                                                                                                     |
| 128125 | <i>getgrgid()</i> , <i>getgrnam()</i> , <i>getpwnam()</i> , <i>getpwuid()</i>                                                                                              |
| 128126 | POSIX_SYSTEM_DATABASE_R: Thread-Safe System Database                                                                                                                       |
| 128127 | <i>getgrgid_r()</i> , <i>getgrnam_r()</i> , <i>getpwnam_r()</i> , <i>getpwuid_r()</i>                                                                                      |
| 128128 | POSIX_THREADS_BASE: Base Threads                                                                                                                                           |
| 128129 | <i>pthread_atfork()</i> , <i>pthread_attr_destroy()</i> , <i>pthread_attr_getdetachstate()</i> ,                                                                           |
| 128130 | <i>pthread_attr_getschedparam()</i> , <i>pthread_attr_init()</i> , <i>pthread_attr_setdetachstate()</i> ,                                                                  |
| 128131 | <i>pthread_attr_setschedparam()</i> , <i>pthread_cancel()</i> , <i>pthread_cleanup_pop()</i> , <i>pthread_cleanup_push()</i> ,                                             |
| 128132 | <i>pthread_cond_broadcast()</i> , <i>pthread_cond_destroy()</i> , <i>pthread_cond_init()</i> , <i>pthread_cond_signal()</i> ,                                              |
| 128133 | <i>pthread_cond_timedwait()</i> , <i>pthread_cond_wait()</i> , <i>pthread_condattr_destroy()</i> ,                                                                         |
| 128134 | <i>pthread_condattr_init()</i> , <i>pthread_create()</i> , <i>pthread_detach()</i> , <i>pthread_equal()</i> , <i>pthread_exit()</i> ,                                      |
| 128135 | <i>pthread_getspecific()</i> , <i>pthread_join()</i> , <i>pthread_key_create()</i> , <i>pthread_key_delete()</i> , <i>pthread_kill()</i> ,                                 |
| 128136 | <i>pthread_mutex_destroy()</i> , <i>pthread_mutex_init()</i> , <i>pthread_mutex_lock()</i> ,                                                                               |
| 128137 | <i>pthread_mutex_timedlock()</i> , <i>pthread_mutex_trylock()</i> , <i>pthread_mutex_unlock()</i> ,                                                                        |
| 128138 | <i>pthread_mutexattr_destroy()</i> , <i>pthread_mutexattr_init()</i> , <i>pthread_once()</i> , <i>pthread_self()</i> ,                                                     |
| 128139 | <i>pthread_setcancelstate()</i> , <i>pthread_setcanceltype()</i> , <i>pthread_setspecific()</i> , <i>pthread_sigmask()</i> ,                                               |
| 128140 | <i>pthread_testcancel()</i>                                                                                                                                                |
| 128141 | POSIX_THREADS_EXT: Extended Threads                                                                                                                                        |
| 128142 | <i>pthread_attr_getguardsize()</i> , <i>pthread_attr_setguardsize()</i> , <i>pthread_mutexattr_gettype()</i> ,                                                             |
| 128143 | <i>pthread_mutexattr_settype()</i>                                                                                                                                         |
| 128144 | POSIX_TIMERS: Timers                                                                                                                                                       |
| 128145 | <i>clock_getres()</i> , <i>clock_gettime()</i> , <i>clock_settime()</i> , <i>nanosleep()</i> , <i>timer_create()</i> , <i>timer_delete()</i> ,                             |
| 128146 | <i>timer_getoverrun()</i> , <i>timer_gettime()</i> , <i>timer_settime()</i>                                                                                                |
| 128147 | POSIX_USER_GROUPS: User and Group                                                                                                                                          |
| 128148 | <i>getegid()</i> , <i>geteuid()</i> , <i>getgid()</i> , <i>getgroups()</i> , <i>getlogin()</i> , <i>getuid()</i> , <i>setegid()</i> , <i>seteuid()</i> , <i>setgid()</i> , |
| 128149 | <i>setuid()</i>                                                                                                                                                            |
| 128150 | POSIX_USER_GROUPS_R: Thread-Safe User and Group                                                                                                                            |
| 128151 | <i>getlogin_r()</i>                                                                                                                                                        |
| 128152 | POSIX_WIDE_CHAR_DEVICE_IO: Device Input and Output                                                                                                                         |
| 128153 | <i>fgetwc()</i> , <i>fgetws()</i> , <i>fputwc()</i> , <i>fputws()</i> , <i>fwide()</i> , <i>fwprintf()</i> , <i>fwscanf()</i> , <i>getwc()</i> , <i>getwchar()</i> ,       |
| 128154 | <i>putwc()</i> , <i>putwchar()</i> , <i>ungetwc()</i> , <i>vfwprintf()</i> , <i>vfwscanf()</i> , <i>vwprintf()</i> , <i>vwscanf()</i> , <i>wprintf()</i> ,                 |
| 128155 | <i>wscanf()</i>                                                                                                                                                            |
| 128156 | XSI_C_LANG_SUPPORT: XSI General C Library                                                                                                                                  |
| 128157 | <i>_tolower()</i> , <i>_toupper()</i> , <i>a64l()</i> , <i>daylight()</i> , <i>drand48()</i> , <i>erand48()</i> , <i>ffs()</i> , <i>getdate()</i> , <i>hcreate()</i> ,     |
| 128158 | <i>hdestroy()</i> , <i>hsearch()</i> , <i>initstate()</i> , <i>insque()</i> , <i>isascii()</i> , <i>jrand48()</i> , <i>l64a()</i> , <i>lcong48()</i> , <i>lfind()</i> ,    |
| 128159 | <i>lrand48()</i> , <i>lsearch()</i> , <i>memccpy()</i> , <i>mrand48()</i> , <i>rand48()</i> , <i>random()</i> , <i>remque()</i> , <i>seed48()</i> ,                        |

128160 12. The *lchown()* function also depends on POSIX\_FILE\_ATTRIBUTES.

|        |                                                                                                                                                                                   |
|--------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 128161 | <i>setstate()</i> , <i>signgam</i> , <i>srand48()</i> , <i>srandom()</i> , <i>strptime()</i> , <i>swab()</i> , <i>tdelete()</i> , <i>tfind()</i> , <i>timezone()</i> ,            |
| 128162 | <i>toascii()</i> , <i>tsearch()</i> , <i>twalk()</i>                                                                                                                              |
| 128163 | XSI_DBM: XSI Database Management                                                                                                                                                  |
| 128164 | <i>dbm_clearerr()</i> , <i>dbm_close()</i> , <i>dbm_delete()</i> , <i>dbm_error()</i> , <i>dbm_fetch()</i> , <i>dbm_firstkey()</i> ,                                              |
| 128165 | <i>dbm_nextkey()</i> , <i>dbm_open()</i> , <i>dbm_store()</i>                                                                                                                     |
| 128166 | XSI_DEVICE_IO: XSI Device Input and Output                                                                                                                                        |
| 128167 | <i>fntmsg()</i> , <i>readv()</i> , <i>writv()</i>                                                                                                                                 |
| 128168 | XSI_DEVICE_SPECIFIC: XSI General Terminal                                                                                                                                         |
| 128169 | <i>grantpt()</i> , <i>posix_openpt()</i> , <i>ptsname()</i> , <i>unlockpt()</i>                                                                                                   |
| 128170 | XSI_FILE_SYSTEM: XSI File System                                                                                                                                                  |
| 128171 | <i>basename()</i> , <i>dirname()</i> , <i>ftw()</i> , <i>lockf()</i> , <i>mknod()</i> , <i>nftw()</i> , <i>realpath()</i> , <i>seekdir()</i> , <i>sync()</i> , <i>telldir()</i> , |
| 128172 | <i>tempnam()</i>                                                                                                                                                                  |
| 128173 | XSI_IPC: XSI Interprocess Communication                                                                                                                                           |
| 128174 | <i>flok()</i> , <i>msgctl()</i> , <i>msgget()</i> , <i>msgrcv()</i> , <i>msgsnd()</i> , <i>semctl()</i> , <i>semget()</i> , <i>semop()</i> , <i>shmat()</i> , <i>shmctl()</i> ,   |
| 128175 | <i>shmdt()</i> , <i>shmget()</i>                                                                                                                                                  |
| 128176 | XSI_JUMP: XSI Jump Functions                                                                                                                                                      |
| 128177 | <i>_longjmp()</i> , <i>_setjmp()</i>                                                                                                                                              |
| 128178 | XSI_MATH: XSI Maths Library                                                                                                                                                       |
| 128179 | <i>j0()</i> , <i>j1()</i> , <i>jn()</i> , <i>y0()</i> , <i>y1()</i> , <i>yn()</i>                                                                                                 |
| 128180 | XSI_MULTI_PROCESS: XSI Multiple Process                                                                                                                                           |
| 128181 | <i>getpriority()</i> , <i>getrlimit()</i> , <i>getrusage()</i> , <i>nice()</i> , <i>setpgrp()</i> , <i>setpriority()</i> , <i>setrlimit()</i> , <i>ulimit()</i> ,                 |
| 128182 | XSI_SIGNALS: XSI Signal                                                                                                                                                           |
| 128183 | <i>killpg()</i> , <i>sigaltstack()</i> , <i>sighold()</i> , <i>sigignore()</i> , <i>siginterrupt()</i> , <i>sigpause()</i> , <i>sigrelse()</i> , <i>sigset()</i> ,                |
| 128184 | XSI_SINGLE_PROCESS: XSI Single Process                                                                                                                                            |
| 128185 | <i>gethostid()</i> , <i>gettimeofday()</i> , <i>putenv()</i>                                                                                                                      |
| 128186 | XSI_SYSTEM_DATABASE: XSI System Database                                                                                                                                          |
| 128187 | <i>endpwent()</i> , <i>getpwent()</i> , <i>setpwent()</i>                                                                                                                         |
| 128188 | XSI_SYSTEM_LOGGING: XSI System Logging                                                                                                                                            |
| 128189 | <i>closelog()</i> , <i>openlog()</i> , <i>setlogmask()</i> , <i>syslog()</i>                                                                                                      |
| 128190 | XSI_THREADS_EXT: XSI Threads Extensions                                                                                                                                           |
| 128191 | <i>pthread_attr_getstack()</i> , <i>pthread_attr_setstack()</i> , <i>pthread_getconcurrency()</i> ,                                                                               |
| 128192 | <i>pthread_setconcurrency()</i>                                                                                                                                                   |
| 128193 | XSI_TIMERS: XSI Timers                                                                                                                                                            |
| 128194 | <i>getitimer()</i> , <i>setitimer()</i>                                                                                                                                           |
| 128195 | XSI_USER_GROUPS: XSI User and Group                                                                                                                                               |
| 128196 | <i>endgrent()</i> , <i>endutxent()</i> , <i>getgrent()</i> , <i>getutxent()</i> , <i>getutxid()</i> , <i>getutxline()</i> , <i>pututxline()</i> ,                                 |
| 128197 | <i>setgrent()</i> , <i>setregid()</i> , <i>setreuid()</i> , <i>setutxent()</i>                                                                                                    |
| 128198 | XSI_WIDE_CHAR: XSI Wide-Character Library                                                                                                                                         |
| 128199 | <i>wcswidth()</i> , <i>wcwidth()</i>                                                                                                                                              |



# Index

|                         |           |
|-------------------------|-----------|
| (time) resolution ..... | 85        |
| / .....                 | 197       |
| /dev .....              | 197       |
| /dev/console .....      | 197       |
| /dev/null .....         | 197       |
| /dev/tty .....          | 197, 3459 |
| /etc/passwd .....       | 3474      |
| /tmp .....              | 197       |
| <aio.h> .....           | 220       |
| <alert> .....           | 34        |
| <apostrophe> .....      | 35        |
| <arpa/inet.h> .....     | 222       |
| <assert.h> .....        | 223       |
| <backspace> .....       | 38        |
| <blank> .....           | 44        |
| <carriage-return> ..... | 46        |
| <circumflex> .....      | 48        |
| <complex.h> .....       | 224       |
| <control>-V .....       | 2675      |
| <control>-W .....       | 2675      |
| <cpio.h> .....          | 227       |
| <ctype.h> .....         | 229       |
| <dirent.h> .....        | 231       |
| <dlfcn.h> .....         | 233       |
| <dollar-sign> .....     | 54        |
| <errno.h> .....         | 234       |
| <fcntl.h> .....         | 238       |
| <fenv.h> .....          | 243       |
| <float.h> .....         | 247       |
| <fmtmsg.h> .....        | 252       |
| <fnmatch.h> .....       | 254       |
| <form-feed> .....       | 63        |
| <ftw.h> .....           | 255       |
| <glob.h> .....          | 257       |
| <grp.h> .....           | 259       |
| <iconv.h> .....         | 261       |
| <inttypes.h> .....      | 262       |
| <iso646.h> .....        | 265       |
| <langinfo.h> .....      | 266       |
| <libgen.h> .....        | 269       |
| <limits.h> .....        | 270       |
| <locale.h> .....        | 286       |
| <math.h> .....          | 289       |
| <monetary.h> .....      | 296       |
| <mqueue.h> .....        | 297       |
| <ndbm.h> .....          | 299       |
| <net/if.h> .....        | 301       |

|                        |           |
|------------------------|-----------|
| <netdb.h> .....        | 302       |
| <netinet/in.h> .....   | 306       |
| <netinet/tcp.h> .....  | 310       |
| <newline> .....        | 71        |
| <nl_types.h> .....     | 311       |
| <number-sign> .....    | 72        |
| <period> .....         | 76        |
| <poll.h> .....         | 312       |
| <pthread.h> .....      | 314, 3603 |
| <pwd.h> .....          | 320       |
| <regex.h> .....        | 322       |
| <sched.h> .....        | 324       |
| <search.h> .....       | 326       |
| <semaphore.h> .....    | 328       |
| <setjmp.h> .....       | 330       |
| <signal.h> .....       | 331       |
| <slash> .....          | 89        |
| <space> .....          | 90        |
| <spawn.h> .....        | 340       |
| <stdarg.h> .....       | 342       |
| <stdbool.h> .....      | 344       |
| <stddef.h> .....       | 345       |
| <stdint.h> .....       | 347       |
| <stdio.h> .....        | 354       |
| <stdlib.h> .....       | 358       |
| <string.h> .....       | 362       |
| <strings.h> .....      | 364       |
| <stropts.h> .....      | 365       |
| <sys/dir.h> .....      | 231       |
| <sys/ipc.h> .....      | 370       |
| <sys/mman.h> .....     | 372       |
| <sys/msg.h> .....      | 375       |
| <sys/resource.h> ..... | 377       |
| <sys/select.h> .....   | 379       |
| <sys/sem.h> .....      | 381       |
| <sys/shm.h> .....      | 383       |
| <sys/socket.h> .....   | 385       |
| <sys/stat.h> .....     | 391       |
| <sys/statvfs.h> .....  | 396       |
| <sys/time.h> .....     | 398       |
| <sys/times.h> .....    | 400       |
| <sys/types.h> .....    | 401       |
| <sys/uio.h> .....      | 405       |
| <sys/un.h> .....       | 406       |
| <sys/utsname.h> .....  | 407       |
| <sys/wait.h> .....     | 408       |
| <syslog.h> .....       | 410       |
| <tab> .....            | 97        |
| <tar.h> .....          | 412       |
| <termios.h> .....      | 414       |
| <tgmath.h> .....       | 420       |
| <tilde> .....          | 98        |

|                                          |                       |
|------------------------------------------|-----------------------|
| <time.h> .....                           | 424                   |
| <trace.h> .....                          | 428                   |
| <ulimit.h> .....                         | 432                   |
| <unistd.h> .....                         | 433                   |
| <utime.h> .....                          | 454                   |
| <utmpx.h> .....                          | 455                   |
| <vertical-tab> .....                     | 103                   |
| <wchar.h> .....                          | 457                   |
| <wctype.h> .....                         | 462                   |
| <wordexp.h> .....                        | 464                   |
| ±0 .....                                 | 105                   |
| _asm_builtin_atoi() .....                | 3526                  |
| _BSD .....                               | 3647                  |
| _CFLAGS .....                            | 2516                  |
| _Complex_I .....                         | 224                   |
| _CS_PATH .....                           | 440                   |
| _CS_POSIX_V6_ILP32_OFF32_CFLAGS .....    | 442                   |
| _CS_POSIX_V6_ILP32_OFF32_LDFLAGS .....   | 442                   |
| _CS_POSIX_V6_ILP32_OFF32_LIBS .....      | 442                   |
| _CS_POSIX_V6_ILP32_OFFBIG_CFLAGS .....   | 442                   |
| _CS_POSIX_V6_ILP32_OFFBIG_LDFLAGS .....  | 442                   |
| _CS_POSIX_V6_ILP32_OFFBIG_LIBS .....     | 442                   |
| _CS_POSIX_V6_LP64_OFF64_CFLAGS .....     | 442                   |
| _CS_POSIX_V6_LP64_OFF64_LDFLAGS .....    | 442                   |
| _CS_POSIX_V6_LP64_OFF64_LIBS .....       | 442                   |
| _CS_POSIX_V6_LPBIG_OFFBIG_CFLAGS .....   | 442                   |
| _CS_POSIX_V6_LPBIG_OFFBIG_LDFLAGS .....  | 442                   |
| _CS_POSIX_V6_LPBIG_OFFBIG_LIBS .....     | 442                   |
| _CS_POSIX_V6_WIDTH_RESTRICTED_ENVS ..... | 442                   |
| _CS_POSIX_V7_ILP32_OFF32_CFLAGS .....    | 440                   |
| _CS_POSIX_V7_ILP32_OFF32_LDFLAGS .....   | 441                   |
| _CS_POSIX_V7_ILP32_OFF32_LIBS .....      | 441                   |
| _CS_POSIX_V7_ILP32_OFFBIG_CFLAGS .....   | 441                   |
| _CS_POSIX_V7_ILP32_OFFBIG_LDFLAGS .....  | 441                   |
| _CS_POSIX_V7_ILP32_OFFBIG_LIBS .....     | 441                   |
| _CS_POSIX_V7_LP64_OFF64_CFLAGS .....     | 441                   |
| _CS_POSIX_V7_LP64_OFF64_LDFLAGS .....    | 441                   |
| _CS_POSIX_V7_LP64_OFF64_LIBS .....       | 441                   |
| _CS_POSIX_V7_LPBIG_OFFBIG_CFLAGS .....   | 441                   |
| _CS_POSIX_V7_LPBIG_OFFBIG_LDFLAGS .....  | 441                   |
| _CS_POSIX_V7_LPBIG_OFFBIG_LIBS .....     | 442                   |
| _CS_POSIX_V7_THREADS_CFLAGS .....        | 442                   |
| _CS_POSIX_V7_WIDTH_RESTRICTED_ENVS ..... | 442                   |
| _CS_V6_ENV .....                         | 442                   |
| _CS_V7_ENV .....                         | 442                   |
| _Exit() .....                            | 549                   |
| _exit() .....                            | 549, 2209, 3542, 3561 |
| _Exit() .....                            | 3720                  |
| _exit() .....                            | 3720                  |
| _FILE .....                              | 603                   |
| _Imaginary_I .....                       | 224                   |
| _IOFBF .....                             | 354, 1869, 1911       |

|                                  |                 |
|----------------------------------|-----------------|
| _IOLBF .....                     | 354, 866, 1911  |
| _IONBF .....                     | 354, 1869, 1911 |
| _LDFLAGS .....                   | 2516            |
| _LIBS .....                      | 2516            |
| _LINE_ .....                     | 603             |
| _longjmp() .....                 | 554, 3723       |
| _LVL .....                       | 476             |
| _MAX .....                       | 475             |
| _MIN .....                       | 270, 475        |
| _PC constants                    |                 |
| defined in <unistd.h> .....      | 443             |
| used in pathconf .....           | 893             |
| _PC_2_SYMLINKS .....             | 893             |
| _PC_ALLOC_SIZE_MIN .....         | 893             |
| _PC_ASYNC_IO .....               | 893             |
| _PC_CHOWN_RESTRICTED .....       | 893             |
| _PC_FILESIZEBITS .....           | 893             |
| _PC_LINK_MAX .....               | 893             |
| _PC_MAX_CANON .....              | 893             |
| _PC_MAX_INPUT .....              | 893             |
| _PC_NAME_MAX .....               | 893             |
| _PC_NO_TRUNC .....               | 893             |
| _PC_PATH_MAX .....               | 893             |
| _PC_PIPE_BUF .....               | 893             |
| _PC_PRIO_IO .....                | 893             |
| _PC_REC_INCR_XFER_SIZE .....     | 893             |
| _PC_REC_MAX_XFER_SIZE .....      | 893             |
| _PC_REC_MIN_XFER_SIZE .....      | 893             |
| _PC_REC_XFER_ALIGN .....         | 893             |
| _PC_SYMLINK_MAX .....            | 893             |
| _PC_SYNC_IO .....                | 893             |
| _PC_TIMESTAMP_RESOLUTION .....   | 893             |
| _PC_VDISABLE .....               | 893             |
| _POSIX .....                     | 270             |
| _POSIX maximum values            |                 |
| in <limits.h> .....              | 276             |
| _POSIX minimum values            |                 |
| in <limits.h> .....              | 276             |
| _POSIX2 constants                |                 |
| in sysconf .....                 | 2077            |
| _POSIX2_BC_BASE_MAX .....        | 275, 279        |
| _POSIX2_BC_DIM_MAX .....         | 275, 279        |
| _POSIX2_BC_SCALE_MAX .....       | 275, 279        |
| _POSIX2_BC_STRING_MAX .....      | 275, 279        |
| _POSIX2_CHARCLASS_NAME_MAX ..... | 275, 279        |
| _POSIX2_CHAR_TERM .....          | 438, 2079       |
| _POSIX2_COLL_WEIGHTS_MAX .....   | 275, 279        |
| _POSIX2_C_BIND .....             | 438, 2079       |
| _POSIX2_C_DEV .....              | 438, 2079       |
| _POSIX2_EXPR_NEST_MAX .....      | 275, 279        |
| _POSIX2_FORT_DEV .....           | 438, 2079       |
| _POSIX2_FORT_RUN .....           | 438, 2079       |

|                                 |                                              |
|---------------------------------|----------------------------------------------|
| _POSIX2_LINE_MAX.....           | 275, 279, 282                                |
| _POSIX2_LOCALEDEF.....          | 438, 2079                                    |
| _POSIX2_PBS.....                | 438, 2079                                    |
| _POSIX2_PBS_ACCOUNTING.....     | 438, 2079                                    |
| _POSIX2_PBS_CHECKPOINT.....     | 439, 2079                                    |
| _POSIX2_PBS_LOCATE.....         | 439, 2079                                    |
| _POSIX2_PBS_MESSAGE.....        | 439, 2079                                    |
| _POSIX2_PBS_TRACK.....          | 439, 2079                                    |
| _POSIX2_RE_DUP_MAX.....         | 272, 275, 279                                |
| _POSIX2_SW_DEV.....             | 439, 2079                                    |
| _POSIX2_SYMLINKS.....           | 440                                          |
| _POSIX2_UPE.....                | 439, 2079                                    |
| _POSIX2_VERSION.....            | 433, 2079                                    |
| _POSIX.....                     | 474                                          |
| _POSIX_ADVISORY_INFO.....       | 18, 23, 433, 895, 2077, 3729                 |
| _POSIX_AIO_LISTIO_MAX.....      | 270, 276                                     |
| _POSIX_AIO_MAX.....             | 271, 276                                     |
| _POSIX_ARG_MAX.....             | 271, 276                                     |
| _POSIX_ASYNCHRONOUS_IO.....     | 17, 434, 2077, 3447, 3729                    |
| _POSIX_ASYNC_IO.....            | 440, 893                                     |
| _POSIX_BARRIERS.....            | 17, 434, 2077, 3447, 3730                    |
| _POSIX_CHILD_MAX.....           | 271, 276                                     |
| _POSIX_CHOWN_RESTRICTED.....    | 17, 434, 664, 893, 896, 3441, 3730           |
| _POSIX_CLOCKRES_MIN.....        | 276                                          |
| _POSIX_CLOCK_SELECTION.....     | 17, 434, 2077, 3447, 3730                    |
| _POSIX_CPUTIME.....             | 18, 23, 434, 2077, 3730                      |
| _POSIX_C_SOURCE.....            | 472-473, 3527, 3530                          |
| _POSIX_DELAYTIMER_MAX.....      | 271, 276                                     |
| _POSIX_FSYNC.....               | 18-19, 23, 434, 2077, 3730                   |
| _POSIX_HOST_NAME_MAX.....       | 271, 276                                     |
| _POSIX_IPV6.....                | 18, 434, 2077, 3730                          |
| _POSIX_JOB_CONTROL.....         | 17, 434, 2077, 3441, 3730, 3734              |
| _POSIX_LINK_MAX.....            | 273, 276                                     |
| _POSIX_LOGIN_NAME_MAX.....      | 271, 276                                     |
| _POSIX_MAPPED_FILES.....        | 17, 434, 2078, 3447, 3730                    |
| _POSIX_MAX_CANON.....           | 273, 277                                     |
| _POSIX_MAX_INPUT.....           | 274, 277                                     |
| _POSIX_MEMLOCK.....             | 18, 23, 434, 2078, 3731                      |
| _POSIX_MEMLOCK_RANGE.....       | 18, 23, 434, 2078, 3731                      |
| _POSIX_MEMORY_PROTECTION.....   | 17, 434, 2078, 3447, 3731                    |
| _POSIX_MESSAGE_PASSING.....     | 18, 23, 434, 2078, 3731                      |
| _POSIX_MONOTONIC_CLOCK.....     | 18, 23, 435, 2078, 3731                      |
| _POSIX_MQ_OPEN_MAX.....         | 271, 277                                     |
| _POSIX_MQ_PRIO_MAX.....         | 271, 277                                     |
| _POSIX_NAME_MAX.....            | 274, 277, 1342, 1352, 1530, 1835, 1845, 1918 |
| _POSIX_NGROUPS_MAX.....         | 275, 277                                     |
| _POSIX_NO_TRUNC.....            | 17, 111, 435, 893, 3441                      |
| _POSIX_OPEN_MAX.....            | 271, 277, 1075                               |
| _POSIX_PATH_MAX.....            | 274, 277, 406, 1342, 1352, 1835, 1845, 1918  |
| _POSIX_PIPE_BUF.....            | 274, 277                                     |
| _POSIX_PRIORITIZED_IO.....      | 18, 23, 435, 503, 2078, 3731                 |
| _POSIX_PRIORITY_SCHEDULING..... | 18, 23-24, 435, 503, 2078, 3731              |

|                                          |                              |
|------------------------------------------|------------------------------|
| _POSIX_PRIO_IO.....                      | 440, 893                     |
| _POSIX_RAW_SOCKETS .....                 | 18, 435, 2078                |
| _POSIX_READER_WRITER_LOCKS .....         | 17, 435, 2078, 3447          |
| _POSIX_REALTIME_SIGNALS.....             | 17, 435, 2078, 3447, 3731    |
| _POSIX_REGEX.....                        | 435, 2078, 3731              |
| _POSIX_RE_DUP_MAX.....                   | 277                          |
| _POSIX_RTSIG_MAX.....                    | 272, 277, 3537, 3736         |
| _POSIX_SAVED_IDS .....                   | 17, 435, 2078, 3441, 3732    |
| _POSIX_SEMAPHORES .....                  | 17, 435, 2078, 3447, 3732    |
| _POSIX_SEM_NSEMS_MAX .....               | 272, 277                     |
| _POSIX_SEM_VALUE_MAX .....               | 272, 277                     |
| _POSIX_SHARED_MEMORY_OBJECTS .....       | 18, 23, 435, 2078, 3732      |
| _POSIX_SHELL .....                       | 435, 2078, 3732              |
| _POSIX_SIGQUEUE_MAX.....                 | 272, 278                     |
| _POSIX_SOURCE.....                       | 473, 3527                    |
| _POSIX_SPAWN.....                        | 18, 23, 435, 2078, 3732      |
| _POSIX_SPINLOCKS .....                   | 3732                         |
| _POSIX_SPIN_LOCKS .....                  | 17, 436, 2078, 3447          |
| _POSIX_SPORADIC_SERVER.....              | 18, 23-24, 436, 2078, 3732   |
| _POSIX_SSIZE_MAX.....                    | 278, 281                     |
| _POSIX_SS_REPL_MAX .....                 | 272, 278, 2078, 3576         |
| _POSIX_STREAM_MAX.....                   | 272, 278                     |
| _POSIX_SYMLINK_MAX.....                  | 274, 278                     |
| _POSIX_SYMLOOP_MAX.....                  | 272, 278                     |
| _POSIX_SYNCHRONIZED_IO .....             | 18, 23, 436, 2078, 3732      |
| _POSIX_SYNC_IO .....                     | 440, 893, 3730               |
| _POSIX_THREADS.....                      | 17, 437, 2078, 3447, 3732    |
| _POSIX_THREAD_ATTR_STACKADDR.....        | 18-19, 436, 2078, 3732       |
| _POSIX_THREAD_ATTR_STACKSIZE .....       | 18-19, 436, 2078, 3733       |
| _POSIX_THREAD_CPUTIME .....              | 18, 24-25, 436, 2078         |
| _POSIX_THREAD_DESTRUCTOR_ITERATIONS..... | 272, 278                     |
| _POSIX_THREAD_KEYS_MAX .....             | 272, 278                     |
| _POSIX_THREAD_PRIORITY_SCHEDULING.....   | 18, 24-25, 436, 2078, 3733   |
| _POSIX_THREAD_PRIO_INHERIT .....         | 18, 24, 436, 2078, 3733      |
| _POSIX_THREAD_PRIO_PROTECT.....          | 18, 24, 436, 2078, 3733      |
| _POSIX_THREAD_PROCESS_SHARED .....       | 18-19, 436, 1663, 2078, 3733 |
| _POSIX_THREAD_ROBUST_PRIO_INHERIT .....  | 24, 436, 2078                |
| _POSIX_THREAD_ROBUST_PRIO_PROTECT .....  | 24, 436, 2078                |
| _POSIX_THREAD_SAFE_FUNCTIONS .....       | 17, 437, 2078, 3447, 3733    |
| _POSIX_THREAD_SPORADIC_SERVER.....       | 18, 24-25, 437, 2078, 3733   |
| _POSIX_THREAD_THREADS_MAX .....          | 272, 278                     |
| _POSIX_TIMEOUTS.....                     | 17, 437, 2078, 3447, 3733    |
| _POSIX_TIMERS .....                      | 17, 437, 2078, 3447, 3733    |
| _POSIX_TIMER_MAX.....                    | 273, 278                     |
| _POSIX_TIMESTAMP_RESOLUTION .....        | 440, 893                     |
| _POSIX_TRACE .....                       | 18, 25, 437, 2078, 3733      |
| _POSIX_TRACE_EVENT_FILTER.....           | 18, 25, 437, 2078, 3734      |
| _POSIX_TRACE_EVENT_NAME_MAX.....         | 273, 278, 1500, 1502, 2078   |
| _POSIX_TRACE_INHERIT .....               | 18, 25, 437, 2078, 3734      |
| _POSIX_TRACE_LOG.....                    | 18, 25, 437, 2078, 3734      |
| _POSIX_TRACE_NAME_MAX .....              | 273, 278, 2078               |
| _POSIX_TRACE_SYS_MAX .....               | 273, 278, 1497, 2078         |

|                                   |                          |
|-----------------------------------|--------------------------|
| _POSIX_TRACE_USER_EVENT_MAX ..... | 273, 279, 1502, 2078     |
| _POSIX_TTY_NAME_MAX .....         | 273, 279                 |
| _POSIX_TYPED_MEMORY_OBJECTS ..... | 18, 23, 437, 2078, 3734  |
| _POSIX_TZNAME_MAX .....           | 273, 279, 3499           |
| _POSIX_V6_ILP32_OFF32 .....       | 437, 2079                |
| _POSIX_V6_ILP32_OFFBIG .....      | 437, 2079                |
| _POSIX_V6_LP64_OFF64 .....        | 437, 2079                |
| _POSIX_V6_LPBIG_OFFBIG .....      | 438, 2079                |
| _POSIX_V7_ILP32_OFF32 .....       | 438, 2078                |
| _POSIX_V7_ILP32_OFFBIG .....      | 438, 2078                |
| _POSIX_V7_LP64_OFF64 .....        | 438, 2078                |
| _POSIX_V7_LPBIG_OFFBIG .....      | 438, 2078                |
| _POSIX_VDISABLE .....             | 17, 446, 893, 3233, 3441 |
| _POSIX_VERSION .....              | 17, 433, 2078, 2168      |
| _PROCESS .....                    | 476                      |
| _PTHREAD_THREADS_MAX .....        | 1632                     |
| _SC constants                     |                          |
| defined in <unistd.h> .....       | <b>443</b>               |
| in sysconf .....                  | 2077                     |
| _SC_2_CHAR_TERM .....             | 2079                     |
| _SC_2_C_BIND .....                | 2079                     |
| _SC_2_C_DEV .....                 | 2079                     |
| _SC_2_FORT_DEV .....              | 2079                     |
| _SC_2_FORT_RUN .....              | 2079                     |
| _SC_2_LOCALEDEF .....             | 2079                     |
| _SC_2_PBS_ACCOUNTING .....        | 2079                     |
| _SC_2_PBS_CHECKPOINT .....        | 2079                     |
| _SC_2_PBS_LOCATE .....            | 2079                     |
| _SC_2_PBS_MESSAGE .....           | 2079                     |
| _SC_2_PBS_TRACK .....             | 2079                     |
| _SC_2_SW_DEV .....                | 2079                     |
| _SC_2_UPE .....                   | 2079                     |
| _SC_2_VERSION .....               | 1421, 2079               |
| _SC_ADVISORY_INFO .....           | 2077                     |
| _SC_AIO_LISTIO_MAX .....          | 2077                     |
| _SC_AIO_MAX .....                 | 2077                     |
| _SC_AIO_PRIO_DELTA_MAX .....      | 2077                     |
| _SC_ARG_MAX .....                 | 2077                     |
| _SC_ASYNCHRONOUS_IO .....         | 2077                     |
| _SC_ATEXIT_MAX .....              | 2077                     |
| _SC_BARRIERS .....                | 2077                     |
| _SC_BC_BASE_MAX .....             | 2077                     |
| _SC_BC_DIM_MAX .....              | 2077                     |
| _SC_BC_SCALE_MAX .....            | 2077                     |
| _SC_BC_STRING_MAX .....           | 2077                     |
| _SC_CHILD_MAX .....               | 2077                     |
| _SC_CLK_TCK .....                 | 2077, 2136               |
| _SC_CLOCK_SELECTION .....         | 2077                     |
| _SC_COLL_WEIGHTS_MAX .....        | 2077                     |
| _SC_CPUTIME .....                 | 2077                     |
| _SC_DELAYTIMER_MAX .....          | 2077                     |
| _SC_EXPR_NEST_MAX .....           | 2077                     |

|                                        |                        |
|----------------------------------------|------------------------|
| _SC_FSYNC .....                        | 2077                   |
| _SC_GETGR_R_SIZE_MAX .....             | 1023, 2077             |
| _SC_GETPW_R_SIZE_MAX .....             | 2077                   |
| _SC_IOV_MAX .....                      | 2077                   |
| _SC_IPV6 .....                         | 2077                   |
| _SC_JOB_CONTROL .....                  | 2077                   |
| _SC_LINE_MAX .....                     | 2077                   |
| _SC_LOGIN_NAME_MAX .....               | 2077                   |
| _SC_MEMLOCK .....                      | 2078                   |
| _SC_MEMLOCK_RANGE .....                | 2078                   |
| _SC_MEMORY_PROTECTION .....            | 2078                   |
| _SC_MESSAGE_PASSING .....              | 2078                   |
| _SC_MONOTONIC_CLOCK .....              | 2078                   |
| _SC_MQ_OPEN_MAX .....                  | 2077                   |
| _SC_MQ_PRIO_MAX .....                  | 2077                   |
| _SC_NGROUPS_MAX .....                  | 2077                   |
| _SC_OPEN_MAX .....                     | 2077                   |
| _SC_PAGESIZE .....                     | 1427, 2079, 3560, 3562 |
| _SC_PAGE_SIZE .....                    | 2079                   |
| _SC_PRIORITIZED_IO .....               | 2078                   |
| _SC_PRIORITY_SCHEDULING .....          | 2078                   |
| _SC_RAW_SOCKETS .....                  | 2078                   |
| _SC_READER_WRITER_LOCKS .....          | 2078                   |
| _SC_REALTIME_SIGNALS .....             | 2078                   |
| _SC_REGEX .....                        | 2078                   |
| _SC_RE_DUP_MAX .....                   | 2079                   |
| _SC_RTSIG_MAX .....                    | 2079                   |
| _SC_SAVED_IDS .....                    | 2078                   |
| _SC_SEMAPHORES .....                   | 2078                   |
| _SC_SEM_NSEMS_MAX .....                | 2079                   |
| _SC_SEM_VALUE_MAX .....                | 2079                   |
| _SC_SHARED_MEMORY_OBJECTS .....        | 2078                   |
| _SC_SHELL .....                        | 2078                   |
| _SC_SIGQUEUE_MAX .....                 | 2079                   |
| _SC_SPAWN .....                        | 2078                   |
| _SC_SPIN_LOCKS .....                   | 2078                   |
| _SC_SPORADIC_SERVER .....              | 2078                   |
| _SC_SS_REPL_MAX .....                  | 2078                   |
| _SC_STREAM_MAX .....                   | 2079                   |
| _SC_SYMLOOP_MAX .....                  | 2079                   |
| _SC_SYNCHRONIZED_IO .....              | 2078                   |
| _SC_THREADS .....                      | 2078                   |
| _SC_THREAD_ATTR_STACKADDR .....        | 2078                   |
| _SC_THREAD_ATTR_STACKSIZE .....        | 2078                   |
| _SC_THREAD_CPUTIME .....               | 2078                   |
| _SC_THREAD_DESTRUCTOR_ITERATIONS ..... | 2079                   |
| _SC_THREAD_KEYS_MAX .....              | 2079                   |
| _SC_THREAD_PRIORITY_SCHEDULING .....   | 2078                   |
| _SC_THREAD_PRIO_INHERIT .....          | 2078                   |
| _SC_THREAD_PRIO_PROTECT .....          | 2078                   |
| _SC_THREAD_PROCESS_SHARED .....        | 2078                   |
| _SC_THREAD_ROBUST_PRIO_INHERIT .....   | 2078                   |



|                                     |                                              |
|-------------------------------------|----------------------------------------------|
| _SC_THREAD_ROBUST_PRIO_PROTECT..... | 2078                                         |
| _SC_THREAD_SAFE_FUNCTIONS.....      | 2078                                         |
| _SC_THREAD_SPORADIC_SERVER.....     | 2078                                         |
| _SC_THREAD_STACK_MIN.....           | 2079                                         |
| _SC_THREAD_THREADS_MAX.....         | 2079                                         |
| _SC_TIMEOUTS.....                   | 2078                                         |
| _SC_TIMERS.....                     | 2078                                         |
| _SC_TIMER_MAX.....                  | 2079                                         |
| _SC_TRACE.....                      | 2078                                         |
| _SC_TRACE_EVENT_FILTER.....         | 2078                                         |
| _SC_TRACE_EVENT_NAME_MAX.....       | 2078                                         |
| _SC_TRACE_INHERIT.....              | 2078                                         |
| _SC_TRACE_LOG.....                  | 2078                                         |
| _SC_TRACE_NAME_MAX.....             | 2078                                         |
| _SC_TRACE_SYS_MAX.....              | 2078                                         |
| _SC_TRACE_USER_EVENT_MAX.....       | 2078                                         |
| _SC_TTY_NAME_MAX.....               | 2079                                         |
| _SC_TYPED_MEMORY_OBJECTS.....       | 2078                                         |
| _SC_TZNAME_MAX.....                 | 2079                                         |
| _SC_V6_ILP32_OFF32.....             | 2079                                         |
| _SC_V6_ILP32_OFFBIG.....            | 2079                                         |
| _SC_V6_LP64_OFF64.....              | 2079                                         |
| _SC_V6_LPBIG_OFFBIG.....            | 2079                                         |
| _SC_V7_ILP32_OFF32.....             | 2078                                         |
| _SC_V7_ILP32_OFFBIG.....            | 2078                                         |
| _SC_V7_LP64_OFF64.....              | 2078                                         |
| _SC_V7_LPBIG_OFFBIG.....            | 2078                                         |
| _SC_VERSION.....                    | 2078                                         |
| _SC_XOPEN_CRYPT.....                | 2079                                         |
| _SC_XOPEN_ENH_I18N.....             | 2079                                         |
| _SC_XOPEN_REALTIME.....             | 2079                                         |
| _SC_XOPEN_REALTIME_THREADS.....     | 2079                                         |
| _SC_XOPEN_SHM.....                  | 2079                                         |
| _SC_XOPEN_STREAMS.....              | 2079                                         |
| _SC_XOPEN_UNIX.....                 | 2079                                         |
| _SC_XOPEN_UUCP.....                 | 2079                                         |
| _SC_XOPEN_VERSION.....              | 2079                                         |
| _setjmp().....                      | 554, 3723                                    |
| _t.....                             | 476                                          |
| _TIME.....                          | 476                                          |
| _tolower().....                     | 556                                          |
| _toupper().....                     | 557                                          |
| _XOPEN_CRYPT.....                   | 18, 22, 439, 2079                            |
| _XOPEN_ENH_I18N.....                | 439, 2079                                    |
| _XOPEN_IOV_MAX.....                 | 271, 280                                     |
| _XOPEN_NAME_MAX.....                | 274, 280, 1342, 1352, 1530, 1835, 1845, 1918 |
| _XOPEN_PATH_MAX.....                | 274, 280, 1342, 1352, 1529, 1835, 1845, 1918 |
| _XOPEN_REALTIME.....                | 18, 22-23, 439, 821, 2079                    |
| _XOPEN_REALTIME_THREADS.....        | 18, 24, 439, 2079                            |
| _XOPEN_SHM.....                     | 439, 2079                                    |
| _XOPEN_SOURCE.....                  | 473, 3527                                    |
| _XOPEN_STREAMS.....                 | 18, 25-26, 439, 2079                         |

|                                               |                                                                      |
|-----------------------------------------------|----------------------------------------------------------------------|
| _XOPEN_UNIX.....                              | 18-19, 439, 2079                                                     |
| _XOPEN_UUCP .....                             | 439, 2079                                                            |
| _XOPEN_VERSION.....                           | 19, 433, 2079                                                        |
| __errno().....                                | 3534                                                                 |
| a64l().....                                   | 558                                                                  |
| ABDAY_.....                                   | 267                                                                  |
| ABDAY_1.....                                  | 1387                                                                 |
| ABMON_.....                                   | 267                                                                  |
| abort().....                                  | 560, 3720                                                            |
| abortive release.....                         | 33                                                                   |
| abs().....                                    | 562                                                                  |
| absolute pathname.....                        | 33, 111                                                              |
| accept().....                                 | 563                                                                  |
| access mode.....                              | 33                                                                   |
| access().....                                 | 565, 3473, 3720                                                      |
| Account_Name.....                             | 2405                                                                 |
| acos().....                                   | 569                                                                  |
| acosf().....                                  | 569                                                                  |
| acosh().....                                  | 571                                                                  |
| acoshf().....                                 | 571                                                                  |
| acoshl().....                                 | 571                                                                  |
| acosl().....                                  | 569, 573                                                             |
| ACTION.....                                   | 1103                                                                 |
| actions equivalent to functions.....          | 2305                                                                 |
| active trace stream.....                      | 3646                                                                 |
| adb                                           |                                                                      |
| rationale for omission.....                   | 3709                                                                 |
| additional file access control mechanism..... | 33                                                                   |
| address families.....                         | 3620                                                                 |
| address information.....                      | 924                                                                  |
| address space.....                            | 33                                                                   |
| address string.....                           | 924                                                                  |
| addressing.....                               | 3620                                                                 |
| addrinfo.....                                 | 303, 924                                                             |
| admin.....                                    | 2424                                                                 |
| ADV.....                                      | 7                                                                    |
| advanced realtime.....                        | 23                                                                   |
| ADVANCED REALTIME.....                        | 340, 671, 1423, 1425, 1427, 1429, 1431, 1435, 1443, 1446, 1449, 1451 |
| .....                                         | 1453, 1455, 1457, 1459, 1461, 1463, 1526, 1528                       |
| advanced realtime threads.....                | 24                                                                   |
| ADVANCED REALTIME THREADS.....                | 1626                                                                 |
| advisory information.....                     | 34, 3547                                                             |
| affirmative response.....                     | 34                                                                   |
| AF.....                                       | 476                                                                  |
| AF_INET.....                                  | 387                                                                  |
| AF_INET6.....                                 | 387                                                                  |
| AF_UNIX.....                                  | 387                                                                  |
| AF_UNSPEC.....                                | 388, 695                                                             |
| AIO_.....                                     | 475                                                                  |
| aio.....                                      | 475                                                                  |
| AIO_ALLDONE.....                              | 220, 574                                                             |
| aio_cancel().....                             | 574, 3555-3556                                                       |

|                                               |                               |
|-----------------------------------------------|-------------------------------|
| AIO_CANCELED .....                            | 220, 574                      |
| aio_error() .....                             | <b>576</b>                    |
| aio_fsync() .....                             | <b>578</b> , 3539, 3554       |
| AIO_LISTIO_MAX .....                          | 270, 1234, 2077, 3734         |
| AIO_MAX .....                                 | 271, 1234, 2077, 3734         |
| AIO_NOTCANCELED .....                         | 220, 574                      |
| AIO_PRIO_DELTA_MAX .....                      | 271, 503, 2077, 3734          |
| aio_read() .....                              | <b>580</b> , 3556             |
| aio_return() .....                            | <b>583</b>                    |
| aio_suspend() .....                           | <b>585</b> , 3555, 3581       |
| aio_write() .....                             | <b>587</b> , 3556             |
| ai_ .....                                     | 475                           |
| AI_ADDRCONFIG .....                           | 303, 925                      |
| AI_ALL .....                                  | 303, 925                      |
| AI_CANONNAME .....                            | 303, 925                      |
| AI_INET6 .....                                | 925                           |
| AI_NUMERICHOST .....                          | 303, 925                      |
| AI_NUMERICSERV .....                          | 303, 925                      |
| AI_PASSIVE .....                              | 303, 925                      |
| AI_V4MAPPED .....                             | 303, 925                      |
| alarm() .....                                 | <b>590</b> , 3544, 3580, 3720 |
| alert .....                                   | 34                            |
| alert character .....                         | 34                            |
| alias .....                                   | 2318, 2339, 2429, 3675, 3724  |
| alias name .....                              | 34                            |
| alias substitution .....                      | 2322, 3678                    |
| alignment .....                               | 35                            |
| alphasort() .....                             | <b>592</b>                    |
| alternate file access control mechanism ..... | 35                            |
| alternate signal stack .....                  | 35                            |
| ALT_DIGITS .....                              | 267                           |
| AM_STR .....                                  | 267                           |
| anchoring .....                               | 187                           |
| ancillary data .....                          | 35                            |
| AND list .....                                | 2343, 3696                    |
| AND-OR list .....                             | 2341                          |
| angle brackets .....                          | 35                            |
| anycast .....                                 | 531                           |
| ANYMARK .....                                 | 368, 1139                     |
| API .....                                     | 36                            |
| apostrophe character .....                    | 35                            |
| appending redirected output .....             | 2335                          |
| application .....                             | 35                            |
| application address .....                     | 36                            |
| application conformance .....                 | 29                            |
| application instrumentation .....             | 3634                          |
| application program interface .....           | 36                            |
| application-managed thread stack .....        | 521, 3620                     |
| appropriate privileges .....                  | 36, 567, 894, 3450            |
| ar. ....                                      | <b>2432</b> , 3726-3727       |
| arbitrary file size .....                     | 3674                          |

|                                           |                                                       |
|-------------------------------------------|-------------------------------------------------------|
| archives                                  |                                                       |
| ar command .....                          | 2432                                                  |
| AREGTYPE .....                            | 412                                                   |
| argc .....                                | 785                                                   |
| argument .....                            | 36                                                    |
| ARG_MAX .....                             | 271, 481, 778, 782, 787, 2077, 3413, 3460, 3668, 3734 |
| arithmetic expansion .....                | 2332, 3686                                            |
| arithmetic language                       |                                                       |
| bc .....                                  | 2493                                                  |
| arithmetic precision and operations ..... | 2305                                                  |
| arm (a timer) .....                       | 36                                                    |
| array identifiers .....                   | 2498                                                  |
| as                                        |                                                       |
| rationale for omission .....              | 3709                                                  |
| asa .....                                 | <b>2440</b> , 3724, 3726, 3728                        |
| ASCII .....                               | 3462                                                  |
| asctime() .....                           | <b>595</b>                                            |
| asctime_r() .....                         | 595                                                   |
| asin() .....                              | <b>598</b>                                            |
| asinf() .....                             | 598                                                   |
| asinh() .....                             | <b>600</b>                                            |
| asinhf() .....                            | 600                                                   |
| asinhf() .....                            | 600                                                   |
| asinhf() .....                            | 600                                                   |
| asinhf() .....                            | 598, 602                                              |
| assert() .....                            | <b>603</b>                                            |
| asterisk .....                            | 36                                                    |
| async-cancel safety .....                 | 3618                                                  |
| async-cancel-safe function .....          | 36                                                    |
| async-signal-safe .....                   | 1543, 3542                                            |
| async-signal-safe function .....          | 37                                                    |
| asynchronous error .....                  | 3622                                                  |
| asynchronous events .....                 | 37                                                    |
| asynchronous I/O .....                    | 3554, 3721                                            |
| completion .....                          | 37                                                    |
| operation .....                           | 37                                                    |
| asynchronous input and output .....       | 37                                                    |
| asynchronous lists .....                  | 2342, 3695                                            |
| asynchronously-generated signal .....     | 37                                                    |
| at .....                                  | <b>2443</b> , 3724                                    |
| at-job .....                              | 2443                                                  |
| atan() .....                              | <b>604</b>                                            |
| atan2() .....                             | <b>606</b>                                            |
| atan2f() .....                            | 606                                                   |
| atan2l() .....                            | 606                                                   |
| atanf() .....                             | 604, 609                                              |
| atanh() .....                             | <b>610</b>                                            |
| atanhf() .....                            | 610                                                   |
| atanhl() .....                            | 610                                                   |
| atanl() .....                             | 604, 612                                              |
| atexit() .....                            | <b>613</b> , 3617                                     |
| ATEXIT_MAX .....                          | 271, 613, 2077                                        |
| atof() .....                              | <b>615</b>                                            |

|                                    |                                                                              |
|------------------------------------|------------------------------------------------------------------------------|
| atoi()                             | 616, 3525-3526                                                               |
| atol()                             | 618                                                                          |
| atoll()                            | 618                                                                          |
| attributes                         |                                                                              |
| clock-resolution                   | 540, 1475                                                                    |
| creation-time                      | 540, 1475                                                                    |
| generation-version                 | 540, 1475                                                                    |
| inheritance                        | 540, 1477                                                                    |
| log-full-policy                    | 538, 540, 1477, 1480                                                         |
| log-max-size                       | 540, 1478, 1480                                                              |
| max-data-size                      | 540, 1480                                                                    |
| stream-full-policy                 | 537-538, 540, 1478                                                           |
| stream-min-size                    | 540, 1481                                                                    |
| trace-name                         | 540, 1475                                                                    |
| truncation-status                  | 1500                                                                         |
| AT_EACCESS                         | 239                                                                          |
| AT_FDCWD                           | 239, 565, 660, 664, 955, 976, 1228, 1301, 1307, 1312, 1394, 1764, 1798, 2073 |
| AT_REMOVEDIR                       | 240, 2174                                                                    |
| AT_SYMLINK_FOLLOW                  | 240, 1228                                                                    |
| AT_SYMLINK_NOFOLLOW                | 240, 660, 664, 955, 977                                                      |
| authentication                     | 37                                                                           |
| authorization                      | 38                                                                           |
| automatic storage class            | 2502                                                                         |
| awk                                | 2452, 3724, 3726                                                             |
| actions                            | 2464                                                                         |
| arithmetic functions               | 2466                                                                         |
| escape sequences                   | 2462                                                                         |
| expression patterns                | 2464                                                                         |
| expressions                        | 2455                                                                         |
| functions                          | 2466                                                                         |
| grammar                            | 2470                                                                         |
| input/output and general functions | 2468                                                                         |
| lexical conventions                | 2476                                                                         |
| output statements                  | 2465                                                                         |
| overall program structure          | 2455                                                                         |
| pattern ranges                     | 2464                                                                         |
| patterns                           | 2463                                                                         |
| regular expressions                | 2461                                                                         |
| special patterns                   | 2463                                                                         |
| string functions                   | 2467                                                                         |
| user-defined functions             | 2469                                                                         |
| variables and special variables    | 2459                                                                         |
| background                         | 1889, 3456-3459, 3511                                                        |
| background job                     | 38                                                                           |
| background process                 | 38, 2110                                                                     |
| background process group           | 38                                                                           |
| background work                    |                                                                              |
| at                                 | 2443                                                                         |
| batch                              | 2490                                                                         |
| bg                                 | 2508                                                                         |
| crontab                            | 2579                                                                         |

|                                       |                      |
|---------------------------------------|----------------------|
| fg .....                              | 2751                 |
| jobs .....                            | 2835                 |
| nice .....                            | 2991                 |
| nohup .....                           | 3004                 |
| renice .....                          | 3157                 |
| backquote .....                       | 38                   |
| BACKREF .....                         | 191                  |
| backslash .....                       | 38, 3676             |
| backspace character .....             | 38                   |
| bandinfo .....                        | <b>365</b>           |
| banner                                |                      |
| rationale for omission .....          | 3709                 |
| barrier .....                         | 39, 3596             |
| basename .....                        | 39, 2487, 3723, 3725 |
| basename() .....                      | <b>619</b>           |
| basic regular expression .....        | 39, 183, 3502        |
| batch .....                           | <b>2490</b> , 3724   |
| general concepts .....                | 3705                 |
| batch access list .....               | 39                   |
| batch administration .....            | 2400                 |
| batch administrator .....             | 39                   |
| batch authorization .....             | 2400                 |
| batch client .....                    | 39                   |
| batch client-server interaction ..... | 2397                 |
| batch destination .....               | 40                   |
| batch destination identifier .....    | 40                   |
| batch directive .....                 | 40                   |
| batch environment .....               | 3702                 |
| option definitions .....              | 3703                 |
| services .....                        | 2397                 |
| utilities .....                       | 2397                 |
| utilities, common behavior .....      | 3708                 |
| batch job .....                       | 40                   |
| batch job abort .....                 | 2400, 2411           |
| batch job attribute .....             | 40                   |
| batch job creation .....              | 2398                 |
| batch job execution .....             | 2399, 2403           |
| batch job exit .....                  | 2400, 2410           |
| batch job identifier .....            | 40, 2419             |
| batch job message request .....       | 2413                 |
| batch job name .....                  | 41                   |
| batch job owner .....                 | 41                   |
| batch job priority .....              | 41                   |
| batch job routing .....               | 2399, 2410           |
| batch job state .....                 | 41                   |
| batch job states .....                | 2402                 |
| batch job status request .....        | 2414                 |
| batch job tracking .....              | 2398                 |
| batch name service .....              | 41                   |
| batch name space .....                | 41                   |
| batch node .....                      | 42                   |
| batch notification .....              | 2401                 |

|                                      |                              |
|--------------------------------------|------------------------------|
| batch operator .....                 | 42                           |
| batch queue .....                    | 42, 2398                     |
| batch queue attribute .....          | 42                           |
| batch queue position .....           | 42                           |
| batch queue priority .....           | 42                           |
| batch queue status request .....     | 2416                         |
| batch rerunability .....             | 43                           |
| batch restart .....                  | 43                           |
| batch server .....                   | 43                           |
| batch server name .....              | 43                           |
| batch server restart .....           | 2411                         |
| batch service .....                  | 43                           |
| batch service request .....          | 43                           |
| batch services .....                 | 2401, 3707                   |
| batch submission .....               | 43                           |
| batch system .....                   | 44                           |
| historical implementations .....     | 3703                         |
| history .....                        | 3702                         |
| batch target user .....              | 44                           |
| batch user .....                     | 44                           |
| baud rate functions .....            | 653                          |
| bc .....                             | 2493, 3723-3724, 3729        |
| grammar .....                        | 2494                         |
| lexical conventions .....            | 2496                         |
| operations .....                     | 2498                         |
| operators .....                      | 2498                         |
| bcc (mailer blind carbon copy) ..... | 2931                         |
| bcmp() .....                         | 3649                         |
| bcopy() .....                        | 3649                         |
| BC_ constants .....                  |                              |
| in sysconf .....                     | 2077                         |
| BC_BASE_MAX .....                    | 275, 2077, 2308, 3668        |
| BC_DIM_MAX .....                     | 275, 2077, 2308, 3668        |
| BC_SCALE_MAX .....                   | 275, 2077, 2308, 3668        |
| BC_STRING_MAX .....                  | 275, 2077, 2308, 2496        |
| BE .....                             | 7                            |
| bg .....                             | 2318, 2339, 2508, 3675, 3724 |
| binary primaries .....               | 3252                         |
| bind .....                           | 44                           |
| bind() .....                         | 621                          |
| bi_ .....                            | 475                          |
| blank character .....                | 44                           |
| blank line .....                     | 44                           |
| blkcnt_t .....                       | 401                          |
| blksize_t .....                      | 401                          |
| BLKTYPE .....                        | 412                          |
| block special file .....             | 45                           |
| block-mode terminal .....            | 45                           |
| blocked process (or thread) .....    | 44                           |
| blocking .....                       | 44                           |
| BOOT_TIME .....                      | 455, 765-766                 |
| bounded response .....               | 3722                         |

|                                              |                    |
|----------------------------------------------|--------------------|
| braces .....                                 | 45                 |
| bracket expression                           |                    |
| grammar .....                                | 3507               |
| brackets .....                               | 45                 |
| BRE                                          |                    |
| expression anchoring .....                   | 3504               |
| grammar lexical conventions .....            | 3507               |
| matching a collating element .....           | 3502               |
| matching a single character .....            | 3502               |
| matching multiple characters .....           | 3504               |
| ordinary character .....                     | 3502               |
| periods .....                                | 3502               |
| precedence .....                             | 3504               |
| special character .....                      | 3502               |
| BRE (ERE) matching a single character .....  | 182                |
| BRE (ERE) matching multiple characters ..... | 182                |
| break .....                                  | <b>2358</b>        |
| BRKINT .....                                 | 415                |
| broadcast .....                              | 45                 |
| BSD .....                                    | 3453, 3510, 3535   |
| BSDLY .....                                  | 416                |
| bsd_signal() .....                           | 3649               |
| bsearch() .....                              | <b>624</b>         |
| BSn .....                                    | 416                |
| btowc() .....                                | <b>627</b>         |
| buffer cache .....                           | 963                |
| BUFSIZ .....                                 | 354, 1869          |
| built-in .....                               | 46                 |
| built-in utilities .....                     | 46, 2318, 3674     |
| builtin .....                                | 2566               |
| BUS_ .....                                   | 475                |
| BUS_ADRALN .....                             | 336                |
| BUS_ADRERR .....                             | 336                |
| BUS_OBJERR .....                             | 336                |
| byte .....                                   | 46                 |
| byte input/output functions .....            | 46                 |
| byte-oriented stream .....                   | 498                |
| byte-stream mode .....                       | 1753               |
| bzero() .....                                | 3650               |
| C Shell .....                                | 3456, 3458         |
| C-language extensions .....                  | 3717, 3723         |
| c99 .....                                    | <b>2511</b> , 3726 |
| external symbols .....                       | 2515               |
| standard libraries .....                     | 2514               |
| cabs() .....                                 | <b>628</b>         |
| cabsf() .....                                | 628                |
| cabsl() .....                                | 628                |
| cacos() .....                                | <b>629</b>         |
| cacosf() .....                               | 629                |
| cacosh() .....                               | <b>630</b>         |
| cacoshf() .....                              | 630                |
| cacoshl() .....                              | 630                |



|                                 |                                   |
|---------------------------------|-----------------------------------|
| cacosl()                        | 629, 631                          |
| cal                             | <b>2522</b>                       |
| calendar                        |                                   |
| rationale for omission          | 3709                              |
| calloc()                        | <b>632</b>                        |
| can                             | 5                                 |
| cancel                          |                                   |
| rationale for omission          | 3709                              |
| cancel-safe                     | 1710                              |
| cancelability state             | 516, 1633, 1710                   |
| cancelability type              | 1633, 1710                        |
| canceled execution of a thread  | 1586                              |
| cancellation cleanup handler    | 1591, 1603, 1622, 1636, 3616-3617 |
| cancellation cleanup stack      | 3616                              |
| cancellation points             | 517                               |
| canonical mode input processing | 202, 3511                         |
| canonical name                  | 925                               |
| carg()                          | <b>634</b>                        |
| cargf()                         | 634                               |
| cargl()                         | 634                               |
| carriage-control characters     | 2440                              |
| carriage-return character       | 46                                |
| case                            | 3697                              |
| case conditional construct      | 2345                              |
| case folding                    | 3474-3475                         |
| casin()                         | <b>635</b>                        |
| casinf()                        | 635                               |
| casinh()                        | <b>636</b>                        |
| casinhf()                       | 636                               |
| casinhl()                       | 636                               |
| casinl()                        | 635, 637                          |
| cat                             | <b>2525</b> , 3674                |
| catan()                         | <b>638</b>                        |
| catanf()                        | 638                               |
| catanh()                        | <b>639</b>                        |
| catanhf()                       | 639                               |
| catanhl()                       | 639                               |
| catanl()                        | 638, 640                          |
| catclose()                      | <b>641</b> , 3725                 |
| catgets()                       | <b>642</b> , 3725                 |
| catopen()                       | <b>644</b> , 3725                 |
| CBAUD                           | 477                               |
| cbrt()                          | <b>646</b>                        |
| cbrtf()                         | 646                               |
| cbrtl()                         | 646                               |
| cc (mailer carbon copy)         | 2931                              |
| ccos()                          | <b>647</b>                        |
| ccosf()                         | 647                               |
| ccosh()                         | <b>648</b>                        |
| ccoshf()                        | 648                               |
| ccoshl()                        | 648                               |
| ccosl()                         | 647, 649                          |

|                                        |                                              |
|----------------------------------------|----------------------------------------------|
| CD.....                                | 7                                            |
| cd .....                               | 2318, 2339, 2529, 3675, 3725                 |
| ceil() .....                           | 650                                          |
| ceilf() .....                          | 650                                          |
| ceill() .....                          | 650                                          |
| CEO .....                              | 3503                                         |
| cexp() .....                           | 652                                          |
| cexpf() .....                          | 652                                          |
| cexpl() .....                          | 652                                          |
| cfgetispeed() .....                    | 653                                          |
| cfgetospeed() .....                    | 655                                          |
| cflow .....                            | 2534                                         |
| cfsetispeed() .....                    | 656                                          |
| cfsetospeed() .....                    | 657                                          |
| change current working directory ..... | 659, 2305                                    |
| change file modes .....                | 662                                          |
| change history .....                   | 3442, 3521, 3665                             |
| change owner and group of file .....   | 666                                          |
| char .....                             | 546, 3649                                    |
| character .....                        | 47, 3451                                     |
| rationale .....                        | 3451                                         |
| character array .....                  | 47                                           |
| character class .....                  | 47                                           |
| character counting .....               | 3395                                         |
| character encoding .....               | 128, 3483                                    |
| state-dependent .....                  | 132                                          |
| character set .....                    | 47, 3482                                     |
| description file .....                 | 3483                                         |
| portable filename .....                | 3462                                         |
| character special file .....           | 47                                           |
| character string .....                 | 47                                           |
| CHARCLASS_NAME_MAX .....               | 275, 3488                                    |
| charmap .....                          |                                              |
| description .....                      | 129                                          |
| with localedef .....                   | 2874                                         |
| writing names with locale .....        | 2868                                         |
| charmap file .....                     | 2872, 3233                                   |
| CHAR_BIT .....                         | 280                                          |
| CHAR_MAX .....                         | 280, 1245, 1247, 3490                        |
| CHAR_MIN .....                         | 280                                          |
| chdir() .....                          | 658                                          |
| Checkpoint .....                       | 2405                                         |
| chgrp .....                            | 2537, 3674, 3725-3726                        |
| child process .....                    | 48, 3451                                     |
| CHILD_MAX .....                        | 271, 890, 2077, 3441, 3648, 3668, 3695, 3734 |
| chmod .....                            | 2540, 3674, 3725-3726                        |
| grammar .....                          | 2543                                         |
| chmod() .....                          | 660, 3720                                    |
| chown .....                            | 2547, 3674, 3725-3726                        |
| chown() .....                          | 664, 3720                                    |
| chroot .....                           |                                              |
| rationale for omission .....           | 3709                                         |

|                            |                                                      |
|----------------------------|------------------------------------------------------|
| chroot()                   | 3462                                                 |
| CHRTYPE                    | 412                                                  |
| cimag()                    | <b>668</b>                                           |
| cimagf()                   | 668                                                  |
| cimagl()                   | 668                                                  |
| circumflex                 | 48                                                   |
| cksum                      | 2551, 3674, 3725                                     |
| CLD_                       | 475                                                  |
| CLD_CONTINUED              | 336                                                  |
| CLD_DUMPED                 | 336                                                  |
| CLD_EXITED                 | 336                                                  |
| CLD_KILLED                 | 336                                                  |
| CLD_STOPPED                | 336                                                  |
| CLD_TRAPPED                | 336                                                  |
| clearerr()                 | <b>669</b>                                           |
| CLOCAL                     | 417                                                  |
| clock                      | 48, 3576                                             |
| clock jump                 | 48                                                   |
| clock tick                 | 48, 590, 2080, 2136, 3451                            |
| per second                 | 2077                                                 |
| rationale                  | 3451                                                 |
| clock()                    | <b>670</b>                                           |
| clock-resolution attribute | 540, 1475                                            |
| clockid_t                  | 401                                                  |
| CLOCKRES_MIN               | 3734                                                 |
| clocks                     | 3576                                                 |
| CLOCKS_PER_SEC             | 401, 424, 670                                        |
| CLOCK_                     | 476                                                  |
| clock_                     | 476                                                  |
| clock_getcpuclockid()      | <b>671</b> , 3583-3584                               |
| clock_getres()             | <b>672</b>                                           |
| clock_gettime()            | 672                                                  |
| CLOCK_MONOTONIC            | 425, 511, 677                                        |
| clock_nanosleep()          | <b>676</b> , 3581                                    |
| CLOCK_PROCESS_CPUTIME_ID   | 425, 511, 3583-3584                                  |
| CLOCK_REALTIME             | 276, 425, 511, 672, 677, 1373, 1658, 2128, 3576-3581 |
| clock_settime()            | 672, 679                                             |
| clock_t                    | 401                                                  |
| CLOCK_THREAD_CPUTIME_ID    | 425, 512, 3583-3584                                  |
| clog()                     | <b>680</b>                                           |
| clogf()                    | 680                                                  |
| clogl()                    | 680                                                  |
| close a file               | 683                                                  |
| close()                    | <b>681</b> , 3561, 3720                              |
| closedir()                 | <b>685</b> , 3721                                    |
| closelog()                 | <b>687</b> , 3726                                    |
| cmp                        | <b>2556</b> , 3674, 3724                             |
| cmmsg_                     | 476                                                  |
| CMSG_                      | 477                                                  |
| CMSG_DATA                  | 386                                                  |
| CMSG_FIRSTHDR              | 386                                                  |
| CMSG_NXTHDR                | 386                                                  |

|                                   |                                        |
|-----------------------------------|----------------------------------------|
| coded character set.....          | 48                                     |
| codes.....                        | 3445                                   |
| codeset .....                     | 49                                     |
| CODESET .....                     | 267                                    |
| codeset conversion.....           | 2818                                   |
| tr.....                           | 3272                                   |
| col                               |                                        |
| rationale for omission.....       | 3709                                   |
| collating element .....           | 49                                     |
| collating element order.....      | 3503                                   |
| collation .....                   | 49                                     |
| collation sequence .....          | 49                                     |
| COLL_ELEM_MULTI.....              | 191                                    |
| COLL_ELEM_SINGLE.....             | 191                                    |
| COLL_WEIGHTS_MAX .....            | 275, 2077, 2308, 3668                  |
| colon.....                        | <b>2360</b>                            |
| column position.....              | 50, 3452                               |
| COLUMNS.....                      | 177, 3498                              |
| comm.....                         | <b>2559</b> , 3724                     |
| command.....                      | 50, 2318, 2339, 2562, 3451, 3675, 3723 |
| command execution.....            | 3693                                   |
| command interpreter               |                                        |
| portable.....                     | 2209                                   |
| command language.....             | 3717, 3723                             |
| command language interpreter..... | 50                                     |
| command mode.....                 | 2642                                   |
| command search.....               | 2339, 3693                             |
| command substitution.....         | 2331, 3684                             |
| communications commands           |                                        |
| mailx.....                        | 2908                                   |
| talk.....                         | 3243                                   |
| uucp.....                         | 3314                                   |
| uudecode.....                     | 3318                                   |
| uuencode .....                    | 3321                                   |
| uustat .....                      | 3326                                   |
| uux.....                          | 3329                                   |
| write .....                       | 3405                                   |
| compare thread IDs.....           | 1621                                   |
| compilation environment.....      | <b>472</b> , 3526                      |
| compilers                         |                                        |
| c99.....                          | 2511                                   |
| fort77 .....                      | 2775                                   |
| yacc.....                         | 3415                                   |
| complex.....                      | 224                                    |
| complex data manipulation.....    | 3718, 3724                             |
| composite graphic symbol .....    | 50                                     |
| compound commands.....            | 2343, 3696                             |
| compound-list.....                | 2341                                   |
| compress.....                     | <b>2568</b>                            |
| compression                       |                                        |
| compress.....                     | 2568                                   |
| uncompress .....                  | 3299                                   |

|                                         |                                                |
|-----------------------------------------|------------------------------------------------|
| zcat .....                              | 3432                                           |
| concepts .....                          | 3444                                           |
| concurrent execution .....              | 107, 3472                                      |
| of processes .....                      | 2301                                           |
| condition variable .....                | 50                                             |
| conditional construct                   |                                                |
| case .....                              | 3697                                           |
| if .....                                | 3697                                           |
| configurable limits .....               | 3729, 3734                                     |
| configuration interrogation .....       | 3716, 3719                                     |
| configuration options .....             | 3727                                           |
| shell and utilities .....               | 3727                                           |
| system interfaces .....                 | 3729                                           |
| configuration values .....              | 2796                                           |
| conformance .....                       | 15, 29, 3442, 3445-3446, 3449-3450, 3475, 3647 |
| POSIX .....                             | 15                                             |
| POSIX system interfaces .....           | 17                                             |
| XSI .....                               | 15                                             |
| XSI system interfaces .....             | 19                                             |
| conformance document .....              | 16, 3442                                       |
| rationale .....                         | 3442                                           |
| conforming application .....            | 16, 1979, 2437, 3449, 3536, 3671, 3673         |
| conforming application, strictly .....  | 590, 785, 3446, 3449, 3542                     |
| conforming implementation options ..... | 20                                             |
| confstr() .....                         | <b>691</b> , 3721                              |
| conj() .....                            | <b>694</b>                                     |
| confj() .....                           | 694                                            |
| conjl() .....                           | 694                                            |
| connect() .....                         | <b>695</b>                                     |
| connected socket .....                  | 51                                             |
| connection .....                        | 51                                             |
| connection indication queue .....       | 3621                                           |
| connection mode .....                   | 51                                             |
| connectionless mode .....               | 51                                             |
| consequences of shell errors .....      | 2337                                           |
| continue .....                          | <b>2362</b>                                    |
| control character .....                 | 51, 3230                                       |
| control data .....                      | 499                                            |
| control mode .....                      | 3514                                           |
| control operator .....                  | 51                                             |
| control-normal .....                    | <b>1753</b>                                    |
| controlling process .....               | 51                                             |
| controlling terminal .....              | 52, 200, 2301, 3452, 3510, 3720                |
| CONTTYPE .....                          | 412                                            |
| conversion descriptor .....             | 52, 779, 784, 1109-1110, 1112-1113             |
| conversion specification .....          | 900, 938, 982, 992, 2017                       |
| modified .....                          | 2025                                           |
| conversion specifier                    |                                                |
| modified .....                          | 2043                                           |
| Coordinated Universal Time (UTC) .....  | 2604                                           |
| copy .....                              | 154                                            |

|                                               |                          |
|-----------------------------------------------|--------------------------|
| copy files commands                           |                          |
| cp .....                                      | 2571                     |
| dd .....                                      | 2606                     |
| ln .....                                      | 2863                     |
| mv .....                                      | 2981                     |
| pax .....                                     | 3032                     |
| copysign() .....                              | <b>698</b>               |
| copysignf() .....                             | 698                      |
| copysignl() .....                             | 698                      |
| core .....                                    | 2210, 3474               |
| core file .....                               | 52, 551                  |
| cos() .....                                   | <b>699</b>               |
| cosf() .....                                  | 699                      |
| cosh() .....                                  | <b>701</b>               |
| coshf() .....                                 | 701                      |
| coshl() .....                                 | 701                      |
| cosl() .....                                  | 699, 703                 |
| covert channel .....                          | 1212, 3475               |
| cp .....                                      | <b>2571</b> , 3674, 3725 |
| cpio                                          |                          |
| rationale for omission .....                  | 3709                     |
| cpio format .....                             | 3053                     |
| cpow() .....                                  | <b>704</b>               |
| cpowf() .....                                 | 704                      |
| cpowl() .....                                 | 704                      |
| cpp                                           |                          |
| rationale for omission .....                  | 3709                     |
| cproj() .....                                 | <b>705</b>               |
| cprojf() .....                                | 705                      |
| cprojl() .....                                | 705                      |
| CPT .....                                     | <b>7</b>                 |
| CPU .....                                     | 400                      |
| CPU time .....                                | 52, 3261                 |
| clock .....                                   | 52                       |
| timer .....                                   | 52                       |
| CRDLY .....                                   | 415                      |
| CREAD .....                                   | 417                      |
| creal() .....                                 | <b>706</b>               |
| crealf() .....                                | 706                      |
| creall() .....                                | 706                      |
| creat() .....                                 | <b>707</b> , 3561, 3674  |
| create a per-process timer .....              | 2129                     |
| create an interprocess channel .....          | 1414                     |
| create session and set process group ID ..... | 1902                     |
| creation-time attribute .....                 | 540, 1475                |
| CRn .....                                     | 415                      |
| CRNCYSTR .....                                | 267                      |
| cron daemon .....                             | 2582                     |
| crontab .....                                 | <b>2579</b> , 3724       |
| CRYPT .....                                   | 709, 750, 1882           |
| crypt() .....                                 | <b>709</b>               |
| csin() .....                                  | <b>711</b>               |

|                             |                    |
|-----------------------------|--------------------|
| csinf()                     | 711                |
| csinh()                     | <b>712</b>         |
| csinhf()                    | 712                |
| csinhl()                    | 712                |
| csinl()                     | 711, 713           |
| CSIZE                       | 417, 3514          |
| CSn                         | 417                |
| csplit                      | <b>2583</b> , 3724 |
| csqrt()                     | <b>714</b>         |
| csqrtf()                    | 714                |
| csqrtl()                    | 714                |
| CSTOPB                      | 417                |
| CS_POSIX_V7_THREADS_LDFLAGS | 442                |
| ctags                       | <b>2587</b> , 3726 |
| ctan()                      | <b>715</b>         |
| ctanf()                     | 715                |
| ctanh()                     | <b>716</b>         |
| ctanhf()                    | 716                |
| ctanhl()                    | 716                |
| ctanl()                     | 715, 717           |
| ctermid()                   | <b>718</b>         |
| ctime()                     | <b>720</b> , 3723  |
| ctime_r()                   | 720                |
| cu                          |                    |
| rationale for omission      | 3709               |
| currency_symbol             | 154                |
| current job                 | 52                 |
| current working directory   | 53, 104, 2301      |
| cursor position             | 53                 |
| cut                         | <b>2592</b> , 3724 |
| CX                          | <b>7</b>           |
| cxref                       | <b>2596</b>        |
| c_                          | 476                |
| C_ constants in <cpio.h>    | <b>227</b>         |
| C_IRGRP                     | 227                |
| C_IROTH                     | 227                |
| C_IRUSR                     | 227                |
| C_ISBLK                     | 227                |
| C_ISCHR                     | 227                |
| C_ISCTG                     | 227                |
| C_ISDIR                     | 227                |
| C_ISFIFO                    | 227                |
| C_ISGID                     | 227                |
| C_ISLNK                     | 227                |
| C_ISREG                     | 227                |
| C_ISSOCK                    | 227                |
| C_ISUID                     | 227                |
| C_ISVTX                     | 227                |
| C_IWGRP                     | 227                |
| C_IWOTH                     | 227                |
| C_IWUSR                     | 227                |
| C_IXGRP                     | 227                |

|                                          |            |
|------------------------------------------|------------|
| C_IXOTH.....                             | 227        |
| C_IXUSR.....                             | 227        |
| data access.....                         | 3716, 3720 |
| data key creation .....                  | 1637       |
| data keywords .....                      | 3082       |
| data messages .....                      | 499        |
| data segment.....                        | 53         |
| data structure                           |            |
| dirent.....                              | 231        |
| entry .....                              | 326        |
| group.....                               | 259        |
| lconv .....                              | 286        |
| msqid_ds .....                           | 375        |
| stat .....                               | 391        |
| data type.....                           | 545, 3647  |
| ACTION .....                             | 326        |
| cc_t.....                                | 414        |
| DIR .....                                | 231        |
| div_t .....                              | 358        |
| ENTRY .....                              | 326        |
| FILE .....                               | 354        |
| fpos_t.....                              | 354        |
| glob_t .....                             | 257        |
| ldiv_t .....                             | 358        |
| lldiv_t.....                             | 358        |
| mbstate_t .....                          | 457        |
| msglen_t .....                           | 375        |
| msgqnum_t .....                          | 375        |
| nl_catd.....                             | 311        |
| nl_item .....                            | 311        |
| pid_t .....                              | 331        |
| ptrdiff_t.....                           | 345        |
| regex_t.....                             | 322        |
| regmatch_t.....                          | 322        |
| regoff_t.....                            | 322        |
| shmatt_t .....                           | 383        |
| sigset_t .....                           | 331        |
| sig_atomic_t .....                       | 331        |
| size_t .....                             | 345        |
| speed_t.....                             | 414        |
| tcflag_t.....                            | 414        |
| VISIT .....                              | 326        |
| wchar_t .....                            | 345        |
| wctrans_t .....                          | 462        |
| wctype_t .....                           | 457        |
| wint_t .....                             | 457        |
| data types                               |            |
| defined in <fenv.h> .....                | 243        |
| defined in <sys/types.h> .....           | 401        |
| date.....                                | 2599, 3725 |
| conversion specifications .....          | 2599       |
| modified conversion specifications ..... | 2600       |



|                                |                                                            |
|--------------------------------|------------------------------------------------------------|
| DATEMSK .....                  | 177, 1009                                                  |
| datum .....                    | 299                                                        |
| daylight .....                 | 722, 2162                                                  |
| DAY_ .....                     | 267                                                        |
| DBL_ constants                 |                                                            |
| defined in <float.h> .....     | 248                                                        |
| DBL_DIG .....                  | 249                                                        |
| DBL_EPSILON .....              | 250                                                        |
| DBL_MANT_DIG .....             | 248                                                        |
| DBL_MAX .....                  | 250                                                        |
| DBL_MAX_10_EXP .....           | 250                                                        |
| DBL_MAX_EXP .....              | 249                                                        |
| DBL_MIN .....                  | 250, 598, 600, 604, 606, 610, 770, 791, 793, 795, 824, 876 |
| DBL_MIN_10_EXP .....           | 249                                                        |
| DBL_MIN_EXP .....              | 249                                                        |
| DBM .....                      | 299, 723-724                                               |
| DBM_ .....                     | 475                                                        |
| dbm_ .....                     | 475                                                        |
| dbm_clearerr() .....           | 723                                                        |
| dbm_close() .....              | 723                                                        |
| dbm_delete() .....             | 723                                                        |
| dbm_error() .....              | 723                                                        |
| dbm_fetch() .....              | 723                                                        |
| dbm_firstkey() .....           | 723                                                        |
| DBM_INSERT .....               | 299, 724                                                   |
| dbm_nextkey() .....            | 723                                                        |
| dbm_open() .....               | 723                                                        |
| DBM_REPLACE .....              | 299, 724                                                   |
| dbm_store() .....              | 723                                                        |
| dc                             |                                                            |
| rationale for omission .....   | 3709                                                       |
| dd .....                       | 2606, 3674, 3724-3725                                      |
| DEAD_PROCESS .....             | 455, 765-766                                               |
| DECIMAL_DIG .....              | 248                                                        |
| default queue .....            | 2415                                                       |
| DEFECHO .....                  | 477                                                        |
| deferred batch service .....   | 53, 2403                                                   |
| deferred cancelability .....   | 1633                                                       |
| defined types .....            | 545, 3647                                                  |
| definitions .....              | 3444                                                       |
| delay process execution .....  | 1978                                                       |
| DELAYTIMER_MAX .....           | 271, 2077, 2133, 3734                                      |
| delete batch job request ..... | 2412                                                       |
| delta .....                    | 2615                                                       |
| dependency order .....         | 738                                                        |
| descriptive name .....         | 924                                                        |
| destination .....              | 2420                                                       |
| destroying a mutex .....       | 1644                                                       |
| destructor functions .....     | 1636                                                       |
| detaching a thread .....       | 1619                                                       |
| determinism .....              | 3716                                                       |
| device .....                   | 53                                                         |

|                                  |                                             |
|----------------------------------|---------------------------------------------|
| output.....                      | 198                                         |
| device ID.....                   | 53                                          |
| device number .....              | 3453                                        |
| device, logical .....            | 3459                                        |
| DEV_BSIZE .....                  | 394                                         |
| dev_t.....                       | 401                                         |
| df.....                          | 2619, 3674, 3725-3726                       |
| diff.....                        | 2623, 3724-3725                             |
| binary output format .....       | 2625                                        |
| default output format .....      | 2625                                        |
| directory comparison format..... | 2624                                        |
| -c or -C output format.....      | 2626                                        |
| -e output format.....            | 2626                                        |
| -f output format .....           | 2626                                        |
| -u or -U output format.....      | 2627                                        |
| difftime() .....                 | 727                                         |
| DIR.....                         | 231, 545, 685, 1759, 1761, 1803, 1824, 2118 |
| dircmp                           |                                             |
| rationale for omission.....      | 3709                                        |
| direct I/O.....                  | 3453                                        |
| directive .....                  | 900, 938, 982, 992                          |
| directory .....                  | 53, 3453                                    |
| device .....                     | 3507                                        |
| entry .....                      | 53, 3453                                    |
| files .....                      | 3507                                        |
| link.....                        | 53                                          |
| list .....                       | 2888                                        |
| operations.....                  | 831                                         |
| protection.....                  | 107, 3473                                   |
| root .....                       | 3462                                        |
| stream.....                      | 54                                          |
| structure.....                   | 3507                                        |
| directory commands               |                                             |
| cd .....                         | 2529                                        |
| pwd .....                        | 3093                                        |
| dirent.....                      | 231, 831                                    |
| dirfd() .....                    | 728                                         |
| dirname.....                     | 2632, 3723, 3725                            |
| dirname() .....                  | 730                                         |
| DIRTYPE.....                     | 412                                         |
| dis                              |                                             |
| rationale for omission.....      | 3709                                        |
| disarm (a timer) .....           | 54                                          |
| disk space commands              |                                             |
| df.....                          | 2619                                        |
| du.....                          | 2635                                        |
| ulimit.....                      | 3288                                        |
| display .....                    | 54, 3453                                    |
| display line .....               | 54                                          |
| div() .....                      | 732                                         |
| dlclose().....                   | 733                                         |
| dlerror().....                   | 735                                         |

|                                       |                                 |
|---------------------------------------|---------------------------------|
| dlopen()                              | 737                             |
| dlsym()                               | 740                             |
| documentation                         | 16, 2956                        |
| dollar-sign                           | 54                              |
| domain error                          | 116                             |
| dot                                   | 54, 831, 1800, 2364, 3453       |
| dot-dot                               | 55, 831, 1800, 3453, 3461, 3478 |
| double-quote                          | 55, 2320, 3676                  |
| downshifting                          | 55                              |
| dprintf()                             | 742, 900                        |
| drand48()                             | 743                             |
| driver                                | 55                              |
| du                                    | 2635, 3674, 3725-3726           |
| dup()                                 | 746, 3561, 3720                 |
| dup2()                                | 746, 3561, 3720                 |
| duplicating an input file descriptor  | 2336                            |
| duplicating an output file descriptor | 2336                            |
| duplocale()                           | 748                             |
| DUP_COUNT                             | 191                             |
| dynamic package initialization        | 1684                            |
| d_                                    | 475                             |
| D_FMT                                 | 267                             |
| D_T_FMT                               | 267                             |
| E2BIG                                 | 234, 481                        |
| EACCES                                | 234, 482                        |
| EADDRINUSE                            | 234, 482                        |
| EADDRNOTAVAIL                         | 234, 482                        |
| EAFNOSUPPORT                          | 234, 482                        |
| EAGAIN                                | 234, 482, 487                   |
| EAI_AGAIN                             | 304, 999                        |
| EAI_BADFLAGS                          | 304, 999                        |
| EAI_FAIL                              | 304, 999                        |
| EAI_FAMILY                            | 304, 999                        |
| EAI_MEMORY                            | 304, 999                        |
| EAI_NONAME                            | 304, 999                        |
| EAI_OVERFLOW                          | 304, 999                        |
| EAI_SERVICE                           | 304, 999                        |
| EAI_SOCKTYPE                          | 304, 999                        |
| EAI_SYSTEM                            | 304, 999                        |
| EALREADY                              | 234, 482                        |
| EBADF                                 | 234, 482                        |
| EBADMSG                               | 234, 482                        |
| EBUSY                                 | 234, 482, 3534, 3604            |
| ECANCELED                             | 234, 482, 3532                  |
| ECHILD                                | 234, 482                        |
| ECHO                                  | 417                             |
| echo                                  | 2639, 3723                      |
| ECHOCTL                               | 477                             |
| ECHOE                                 | 417, 3514                       |
| ECHOK                                 | 417, 3514                       |
| ECHOKE                                | 477                             |
| ECHONL                                | 417, 3514                       |

|                                              |                 |
|----------------------------------------------|-----------------|
| ECHOPRT .....                                | 477             |
| ECONNABORTED .....                           | 234, 483        |
| ECONNREFUSED .....                           | 234, 483        |
| ECONNRESET .....                             | 234, 483        |
| ecvt() .....                                 | 3650            |
| ed .....                                     | 2642, 3724-3725 |
| addresses .....                              | 2644            |
| append command .....                         | 2646            |
| change command .....                         | 2647            |
| commands .....                               | 2645            |
| copy command .....                           | 2652            |
| delete command .....                         | 2647            |
| edit command .....                           | 2647            |
| edit without checking command .....          | 2647            |
| filename command .....                       | 2648            |
| global command .....                         | 2648            |
| global non-matched command .....             | 2652            |
| help command .....                           | 2649            |
| help-mode command .....                      | 2649            |
| insert command .....                         | 2649            |
| interactive global command .....             | 2648            |
| interactive global not-matched command ..... | 2652            |
| join command .....                           | 2649            |
| line number command .....                    | 2653            |
| list command .....                           | 2649            |
| mark command .....                           | 2649            |
| move command .....                           | 2650            |
| null command .....                           | 2653            |
| number command .....                         | 2650            |
| print command .....                          | 2650            |
| prompt command .....                         | 2650            |
| quit command .....                           | 2650            |
| quit without checking command .....          | 2650            |
| read command .....                           | 2651            |
| regular expressions .....                    | 2644            |
| shell escape command .....                   | 2653            |
| substitute command .....                     | 2651            |
| undo command .....                           | 2652            |
| write command .....                          | 2652            |
| EDEADLK .....                                | 234, 483        |
| EDESTADDRREQ .....                           | 234, 483        |
| edit buffer .....                            | 2662, 3336      |
| edit line .....                              | 3194            |
| editors .....                                |                 |
| ed .....                                     | 2642            |
| ex .....                                     | 2662            |
| sed .....                                    | 3179            |
| vi .....                                     | 3336            |
| EDOM .....                                   | 234, 483, 3534  |
| EDQUOT .....                                 | 234, 483        |
| ED_FILE_MAX .....                            | 2654            |
| ED_LINE_MAX .....                            | 2655            |

|                                    |                                      |
|------------------------------------|--------------------------------------|
| EEXIST .....                       | 234, 483                             |
| EFAULT .....                       | 235, 483, 3532                       |
| EFBIG .....                        | 235, 483                             |
| effective group ID.....            | 55, 666, 786, 1030, 2301             |
| effective user ID.....             | 55, 567, 786, 1212, 2301, 3473       |
| EFTYPE.....                        | 3532                                 |
| EHOSTUNREACH .....                 | 235, 483                             |
| EIDRM .....                        | 235, 483                             |
| eight-bit transparency.....        | 55                                   |
| Eighth Edition UNIX .....          | 2291, 2566                           |
| EILSEQ.....                        | 235, 483, 499, 3534                  |
| EINPROGRESS.....                   | 235, 483, 504, 3555                  |
| EINTR.....                         | 235, 484, 519, 3532, 3535, 3544-3545 |
| EINVAL .....                       | 235, 484, 3532                       |
| EIO.....                           | 235, 484                             |
| EISCONN .....                      | 235, 484                             |
| EISDIR.....                        | 235, 484                             |
| ELOOP .....                        | 235, 484, 3532                       |
| ELSIZE .....                       | 1275                                 |
| emacs                              |                                      |
| rationale for omission.....        | 3710                                 |
| EMFILE.....                        | 235, 484                             |
| EMLINK .....                       | 235, 484                             |
| EMPTY.....                         | 455, 766                             |
| empty directory .....              | 56                                   |
| empty line.....                    | 56                                   |
| empty string (or null string)..... | 56                                   |
| empty wide-character string .....  | 56                                   |
| EMSGSIZE.....                      | 235, 484                             |
| EMULTIHOP .....                    | 235, 484                             |
| ENAMETOOLONG.....                  | 235, 484, 3533                       |
| encoding                           |                                      |
| character .....                    | 128                                  |
| encoding rule .....                | 56                                   |
| encrypt().....                     | <b>750</b>                           |
| encryption .....                   | 22                                   |
| endgrent().....                    | <b>752</b> , 3471                    |
| endhostent().....                  | <b>754</b>                           |
| endnetent() .....                  | <b>756</b>                           |
| endprotoent() .....                | <b>758</b>                           |
| endpwent() .....                   | <b>760</b> , 3471                    |
| endservent().....                  | <b>763</b>                           |
| endutxent() .....                  | <b>765</b>                           |
| ENETDOWN .....                     | 235, 484                             |
| ENETRESET .....                    | 235, 485                             |
| ENETUNREACH.....                   | 235, 485                             |
| ENFILE .....                       | 235, 485                             |
| ENOBUFS.....                       | 235, 485                             |
| ENODATA .....                      | 235, 485                             |
| ENODEV .....                       | 235, 485                             |
| ENOENT .....                       | 235, 485                             |
| ENOEXEC .....                      | 235, 485                             |

|                                   |                           |
|-----------------------------------|---------------------------|
| ENOLCK .....                      | 235, 485                  |
| ENOLINK.....                      | 235, 485                  |
| ENOMEM.....                       | 235, 485, 3533            |
| ENOMSG.....                       | 235, 485                  |
| ENOPROTOOPT.....                  | 235, 485                  |
| ENOSPC .....                      | 235, 485                  |
| ENOSR.....                        | 235, 485                  |
| ENOSTR .....                      | 236, 486                  |
| ENOSYS.....                       | 236, 486, 3533, 3558      |
| ENOTCONN.....                     | 236, 486                  |
| ENOTDIR.....                      | 236, 486                  |
| ENOTEMPTY .....                   | 236, 486                  |
| ENOTRECOVERABLE.....              | 236, 486                  |
| ENOTSOCK .....                    | 236, 486                  |
| ENOTSUP.....                      | 236, 486, 3533            |
| ENOTTY.....                       | 236, 486, 3508, 3532-3533 |
| entire regular expression.....    | 56, 181                   |
| ENTRY .....                       | 1103                      |
| env .....                         | <b>2658</b> , 3723, 3726  |
| environ.....                      | <b>768</b> , 786          |
| environment access .....          | 3716, 3721                |
| environment variable.....         | 3497                      |
| definition .....                  | 3497                      |
| internationalization.....         | 174                       |
| envp.....                         | 786                       |
| ENXIO.....                        | 236, 486                  |
| EOF.....                          | 355                       |
| EOPNOTSUPP.....                   | 236, 486                  |
| EOVERFLOW .....                   | 236, 486, 3533            |
| EOWNERDEAD.....                   | 236, 486                  |
| EPERM.....                        | 236, 486, 2576, 3613      |
| EPIPE.....                        | 236, 486, 3534            |
| Epoch .....                       | 57, 3454, 3479, 3577      |
| EPROTO .....                      | 236, 487                  |
| EPROTONOSUPPORT.....              | 236, 487                  |
| EPROTOTYPE .....                  | 236, 487                  |
| equivalence class .....           | 57                        |
| era .....                         | 57                        |
| ERA .....                         | 267                       |
| erand48().....                    | 743, 769                  |
| ERANGE.....                       | 236, 487, 3534            |
| ERASE.....                        | 3511                      |
| ERA_D_FMT.....                    | 267                       |
| ERA_D_T_FMT .....                 | 267                       |
| ERA_T_FMT .....                   | 267                       |
| ERE.....                          | 3505                      |
| alternation .....                 | 3506                      |
| bracket expression.....           | 3506                      |
| expression anchoring.....         | 3506                      |
| grammar .....                     | 3507                      |
| grammar lexical conventions ..... | 3507                      |
| matching a collating element..... | 3505                      |

|                                       |                         |
|---------------------------------------|-------------------------|
| matching a single character .....     | 3505                    |
| matching multiple characters .....    | 3506                    |
| ordinary character .....              | 3505                    |
| periods .....                         | 3506                    |
| precedence .....                      | 3506                    |
| special character .....               | 3506                    |
| erf() .....                           | <b>770</b>              |
| erfc() .....                          | <b>772</b>              |
| erfcf() .....                         | 772                     |
| erfcl() .....                         | 772                     |
| erff() .....                          | 770, 774                |
| erfl() .....                          | 770, 774                |
| EROFS .....                           | 236, 487, 3534          |
| errno .....                           | <b>775</b> , 3531       |
| per-thread .....                      | 3534                    |
| error conditions .....                | 3481, 3691              |
| mathematical functions .....          | 116                     |
| error descriptions .....              | 999                     |
| error handling .....                  | 3700                    |
| error numbers .....                   | 481, 3531, 3535         |
| additional .....                      | 488                     |
| Error_Path .....                      | 2406                    |
| escape character .....                | 3676                    |
| escape character (backslash) .....    | 2320                    |
| escape sequences                      |                         |
| awk .....                             | 2462                    |
| gencat .....                          | 2785                    |
| lex .....                             | 2854                    |
| ESPIPE .....                          | 236, 487                |
| ESRCH .....                           | 236, 487                |
| EST5EDT .....                         | 2162                    |
| establish cancellation handlers ..... | 1591                    |
| establish the locale .....            | 2305                    |
| ESTALE .....                          | 236, 487                |
| ETIME .....                           | 236, 487                |
| ETIMEDOUT .....                       | 236, 487                |
| ETXTBSY .....                         | 236, 487                |
| eval .....                            | <b>2366</b>             |
| event management .....                | 57                      |
| EWOULDBLOCK .....                     | 236, 487                |
| ex .....                              | <b>2662</b> , 3724-3725 |
| <backslash> .....                     | 2675                    |
| <control>-D command .....             | 2698                    |
| <newline> .....                       | 2675                    |
| abbreviate command .....              | 2678                    |
| addressing .....                      | 2668                    |
| adjust window command .....           | 2696                    |
| append command .....                  | 2678                    |
| args command .....                    | 2679                    |
| autoindent option .....               | 2700                    |
| autoprint option .....                | 2701                    |
| autowrite option .....                | 2701                    |

|                            |            |
|----------------------------|------------|
| beautify option .....      | 2701       |
| change command .....       | 2679       |
| chdir command .....        | 2679       |
| command descriptions ..... | 2675       |
| copy command .....         | 2679       |
| delete command .....       | 2680       |
| directory option .....     | 2702       |
| edcompatible option .....  | 2702       |
| edit command .....         | 2680       |
| edit options .....         | 2700       |
| errorbells option .....    | 2702       |
| escape command .....       | 2697       |
| execute command .....      | 2699       |
| exrc command .....         | 2702       |
| file command .....         | 2681       |
| global command .....       | 2681       |
| ignorecase option .....    | 2702       |
| initialization .....       | 2666       |
| input editing .....        | 2673       |
| insert command .....       | 2682       |
| join command .....         | 2682       |
| list command .....         | 2683, 2702 |
| magic command .....        | 2703       |
| map command .....          | 2683       |
| mark command .....         | 2685       |
| mesg command .....         | 2703       |
| move command .....         | 2686       |
| next command .....         | 2686       |
| number command .....       | 2687       |
| number option .....        | 2703       |
| open command .....         | 2687       |
| paragraphs option .....    | 2703       |
| preserve command .....     | 2662, 2687 |
| print command .....        | 2687       |
| prompt command .....       | 2703       |
| put command .....          | 2688       |
| quit command .....         | 2688       |
| read command .....         | 2688       |
| readonly command .....     | 2704       |
| recover command .....      | 2689       |
| redraw command .....       | 2704       |
| regular expressions .....  | 2699       |
| remap command .....        | 2704       |
| replacement strings .....  | 2700       |
| report command .....       | 2704       |
| rewind command .....       | 2689       |
| scroll command .....       | 2674, 2705 |
| sections command .....     | 2705       |
| set command .....          | 2690       |
| shell command .....        | 2690       |
| shell option .....         | 2705       |
| shift left command .....   | 2698       |



|                                          |                                                                         |
|------------------------------------------|-------------------------------------------------------------------------|
| shift right command .....                | 2698                                                                    |
| shiftwidth option.....                   | 2705                                                                    |
| showmatch option.....                    | 2705                                                                    |
| showmode command .....                   | 2705                                                                    |
| slowopen command.....                    | 2706                                                                    |
| source command .....                     | 2690                                                                    |
| substitute command .....                 | 2690                                                                    |
| suspend command.....                     | 2692                                                                    |
| tabstop option.....                      | 2706                                                                    |
| tag command .....                        | 2692                                                                    |
| taglength option .....                   | 2706                                                                    |
| tags command.....                        | 2706                                                                    |
| term command .....                       | 2706                                                                    |
| terse command .....                      | 2706                                                                    |
| unabbrev command.....                    | 2693                                                                    |
| undo command .....                       | 2693                                                                    |
| unmap command .....                      | 2693                                                                    |
| version command.....                     | 2694                                                                    |
| visual command.....                      | 2694                                                                    |
| warn command.....                        | 2707                                                                    |
| window command .....                     | 2707                                                                    |
| wrapmargin option.....                   | 2707                                                                    |
| wrapscan option.....                     | 2708                                                                    |
| write command .....                      | 2694                                                                    |
| write line number command.....           | 2699                                                                    |
| writeln option .....                     | 2708                                                                    |
| xit command .....                        | 2695                                                                    |
| yank command.....                        | 2696                                                                    |
| examine and change blocked signals ..... | 1718                                                                    |
| examine and change signal action .....   | 1934                                                                    |
| EXDEV .....                              | 236, 488                                                                |
| exec.....                                | 777, 2368, 3005                                                         |
| of shell scripts .....                   | 785                                                                     |
| exec family .....                        | 567, 683, 818, 864, 891, 1543, 1890, 2209, 2318, 2566, 2989, 3413, 3456 |
| .....                                    | 3559, 3669, 3698, 3720                                                  |
| exec().....                              | 3640                                                                    |
| execl() .....                            | 777                                                                     |
| execle() .....                           | 777                                                                     |
| execlp().....                            | 777                                                                     |
| executable file .....                    | 57                                                                      |
| execute .....                            | 58                                                                      |
| execute a file.....                      | 785                                                                     |
| execution time.....                      | 52, 58                                                                  |
| measurement.....                         | 110, 3476                                                               |
| monitoring.....                          | 58, 511, 3582                                                           |
| Execution_Time .....                     | 2407                                                                    |
| execv() .....                            | 777                                                                     |
| execve() .....                           | 777                                                                     |
| execvp().....                            | 777                                                                     |
| EXINIT .....                             | 2662                                                                    |
| exit .....                               | 2370                                                                    |
| exit status.....                         | 3691                                                                    |

|                                   |                                                         |
|-----------------------------------|---------------------------------------------------------|
| and errors .....                  | 2337                                                    |
| for commands .....                | 2337                                                    |
| exit() .....                      | <b>790</b> , 3542, 3720                                 |
| EXIT_FAILURE .....                | 358, 549, 790, 3722                                     |
| EXIT_SUCCESS .....                | 358, 549, 790, 3722                                     |
| exp() .....                       | <b>791</b>                                              |
| exp2() .....                      | <b>793</b>                                              |
| exp2f() .....                     | 793                                                     |
| exp2l() .....                     | 793                                                     |
| expand .....                      | 58, 2735, 3724                                          |
| expf() .....                      | 791                                                     |
| expl() .....                      | 791                                                     |
| expm1() .....                     | <b>795</b>                                              |
| expm1f() .....                    | 795                                                     |
| expm1l() .....                    | 795                                                     |
| export .....                      | <b>2372</b>                                             |
| expr .....                        | <b>2738</b> , 3723-3724                                 |
| matching expression .....         | 2740                                                    |
| string operand .....              | 2740                                                    |
| expression argument .....         | 2465                                                    |
| expression list .....             | 2465                                                    |
| EXPR_NEST_MAX .....               | 275, 2077, 2308, 3668                                   |
| EXTA .....                        | 477                                                     |
| EXTB .....                        | 477                                                     |
| extended regular expression ..... | 58, 188, 2461, 2575, 2765, 2807, 2853, 2983, 3042, 3162 |
| .....                             | 3410, 3505                                              |
| extended security controls .....  | 58, 107, 3473                                           |
| extension                         |                                                         |
| CX .....                          | 7                                                       |
| OH .....                          | 9                                                       |
| XSI .....                         | 12                                                      |
| F-LOCK .....                      | <b>443</b>                                              |
| fabs() .....                      | <b>797</b>                                              |
| fabsf() .....                     | 797                                                     |
| fabsl() .....                     | 797                                                     |
| faccessat() .....                 | 565, 799                                                |
| false .....                       | 2318, 2339, 2743, 3675, 3723                            |
| fattach() .....                   | <b>800</b>                                              |
| fc .....                          | 2318, 2339, 2745, 3675, 3724                            |
| fchdir() .....                    | <b>803</b>                                              |
| fchmod() .....                    | <b>804</b> , 3720                                       |
| fchmodat() .....                  | 660, 806                                                |
| fchown() .....                    | <b>807</b>                                              |
| fchownat() .....                  | 664, 809                                                |
| fclose() .....                    | <b>810</b> , 3720                                       |
| fcntl() .....                     | <b>813</b> , 3508, 3532, 3561, 3720                     |
| fcntl() locks .....               | 3619                                                    |
| fcvt() .....                      | 3650                                                    |
| FD .....                          | 7                                                       |
| fdatasync() .....                 | <b>821</b>                                              |
| fdetach() .....                   | <b>822</b>                                              |
| fdim() .....                      | <b>824</b>                                              |

|                     |                                                                 |
|---------------------|-----------------------------------------------------------------|
| fdimf()             | 824                                                             |
| fdiml()             | 824                                                             |
| fdopen()            | 826, 3561                                                       |
| fdopendir()         | 829                                                             |
| fds_                | 475                                                             |
| FD_                 | 475                                                             |
| fd_                 | 475                                                             |
| FD_CLOEXEC          | 238, 496, 644, 747, 778, 813, 831, 1113, 1391, 1436, 1443, 1913 |
| FD_CLR              | 1536                                                            |
| FD_CLR()            | 548                                                             |
| FD_ISSET            | 548, 1536                                                       |
| fd_set              | 379, 398                                                        |
| FD_SET              | 548, 1536                                                       |
| FD_SETSIZE          | 379                                                             |
| FD_ZERO             | 548, 1536                                                       |
| feature test macro  | 59, 472, 1003, 3527, 3647                                       |
| _POSIX_C_SOURCE     | 472                                                             |
| _XOPEN_SOURCE       | 473                                                             |
| feclearexcept()     | 833                                                             |
| fegetenv()          | 834                                                             |
| fegetexceptflag()   | 835                                                             |
| fegetround()        | 836                                                             |
| fehldexcept()       | 838                                                             |
| fenv_t              | 243                                                             |
| feof()              | 839                                                             |
| feraiseexcept()     | 840                                                             |
| error()             | 841                                                             |
| fesetenv()          | 834, 842                                                        |
| fesetexceptflag()   | 835, 843                                                        |
| fesetround()        | 836, 844                                                        |
| fetestexcept()      | 845                                                             |
| feupdateenv()       | 847                                                             |
| fexcept_t           | 243                                                             |
| fexecve             | 849                                                             |
| fexecve()           | 777                                                             |
| FE_                 | 477                                                             |
| FE_constants        |                                                                 |
| defined in <fenv.h> | 243                                                             |
| FE_ALL_EXCEPT       | 243                                                             |
| FE_DFL_ENV          | 244                                                             |
| FE_DIVBYZERO        | 243                                                             |
| FE_DOWNWARD         | 243                                                             |
| FE_INEXACT          | 243                                                             |
| FE_INVALID          | 243                                                             |
| FE_OVERFLOW         | 243                                                             |
| FE_TONEAREST        | 243                                                             |
| FE_TOWARDZERO       | 243                                                             |
| FE_UNDERFLOW        | 243                                                             |
| FE_UPWARD           | 243                                                             |
| FFDLY               | 416                                                             |
| fflush()            | 850                                                             |
| FFn                 | 416                                                             |

|                         |                                             |
|-------------------------|---------------------------------------------|
| ffs()                   | 853                                         |
| fg                      | 2318, 2339, 2751, 3675, 3724                |
| fgetc()                 | 854, 3721                                   |
| fgetpos()               | 856                                         |
| fgets()                 | 858                                         |
| fgetwc()                | 860                                         |
| fgetws()                | 862                                         |
| field                   | 59                                          |
| field splitting         | 2333, 3688                                  |
| FIFO                    | 59, 1307-1308, 1397, 2289, 3454, 3462, 3570 |
| FIFO special file       | 59, 2966, 3454                              |
| FIFOTYPE                | 412                                         |
| file                    | 59                                          |
| FILE                    | 355, 457, 545                               |
| file                    | 2753, 3454                                  |
| locking                 | 818                                         |
| file access permissions | 108, 2302, 3473                             |
| file accessibility      | 566                                         |
| file characteristics    |                                             |
| data structure          | 394                                         |
| header                  | 394                                         |
| file classes            | 3454                                        |
| file comparisons        |                                             |
| cmp                     | 2556                                        |
| comm                    | 2559                                        |
| diff                    | 2623                                        |
| uniq                    | 3308                                        |
| file contents           | 2304                                        |
| file control            | 818                                         |
| file conversion         |                                             |
| cut                     | 2592                                        |
| dd                      | 2606                                        |
| expand                  | 2735                                        |
| fold                    | 2771                                        |
| head                    | 2815                                        |
| join                    | 2839                                        |
| od                      | 3008                                        |
| paste                   | 3016                                        |
| patch                   | 3020                                        |
| sort                    | 3210                                        |
| strings                 | 3221                                        |
| tail                    | 3239                                        |
| tr                      | 3272                                        |
| tsort                   | 3281                                        |
| unexpand                | 3302                                        |
| uniq                    | 3308                                        |
| uudecode                | 3318                                        |
| uuencode                | 3321                                        |
| file creation           | 2302, 3666                                  |
| file description        | 59                                          |
| file descriptor         | 60, 2301, 2334, 2340, 3545, 3690            |
| file format notation    | 3482                                        |

|                                   |                           |
|-----------------------------------|---------------------------|
| file group class .....            | 60                        |
| file hierarchy .....              | 108, 3473                 |
| file hierarchy manipulation ..... | 3718, 3725                |
| file mode .....                   | 60                        |
| file mode bits .....              | 60                        |
| file mode creation mask .....     | 2301                      |
| FILE object .....                 | 495                       |
| file offset .....                 | 61                        |
| file other class .....            | 61                        |
| file owner class .....            | 61                        |
| file permission bits .....        | 61, 567                   |
| file permission commands          |                           |
| chgrp .....                       | 2537                      |
| chmod .....                       | 2540                      |
| chown .....                       | 2547                      |
| umask .....                       | 3290                      |
| file permissions .....            | 567, 896, 957, 3473, 3511 |
| file position indicator .....     | 495                       |
| file read .....                   | 2302, 3666                |
| file removal .....                | 2304, 3666                |
| file searching                    |                           |
| grep .....                        | 2807                      |
| file serial number .....          | 61                        |
| file size, arbitrary .....        | 3674                      |
| file system .....                 | 61, 3454                  |
| file system, mounted .....        | 3460                      |
| file system, root .....           | 3462-3463                 |
| file time values .....            | 2304                      |
| file times update .....           | 109, 3475                 |
| file tree commands                |                           |
| diff .....                        | 2623                      |
| find .....                        | 2761                      |
| ls .....                          | 2888                      |
| mkdir .....                       | 2963                      |
| rmdir .....                       | 3168                      |
| file type .....                   | 62                        |
| file write .....                  | 2302, 3666                |
| file, passwd .....                | 3461                      |
| filename .....                    | 60, 109, 3454, 3474       |
| filename portability .....        | 109, 3475                 |
| filename string .....             | 61                        |
| FILENAME_MAX .....                | 354                       |
| fileno() .....                    | 864, 3460                 |
| FILESIZEBITS .....                | 273, 893                  |
| filter .....                      | 62                        |
| filtering trace event types ..... | 3643                      |
| filters                           |                           |
| asa .....                         | 2440                      |
| awk .....                         | 2452                      |
| compress .....                    | 2568                      |
| dd .....                          | 2606                      |
| expand .....                      | 2735                      |

|                              |                                                            |
|------------------------------|------------------------------------------------------------|
| fold .....                   | 2771                                                       |
| head .....                   | 2815                                                       |
| iconv .....                  | 2818                                                       |
| more .....                   | 2969                                                       |
| nl .....                     | 2995                                                       |
| paste .....                  | 3016                                                       |
| pax .....                    | 3032                                                       |
| pr .....                     | 3070                                                       |
| read .....                   | 3154                                                       |
| sed .....                    | 3179                                                       |
| tail .....                   | 3239                                                       |
| tee .....                    | 3247                                                       |
| tr .....                     | 3272                                                       |
| uncompress .....             | 3299                                                       |
| unexpand .....               | 3302                                                       |
| zcat .....                   | 3432                                                       |
| FIND .....                   | 1103                                                       |
| find .....                   | <b>2761</b> , 3724-3725                                    |
| find string token .....      | 2058                                                       |
| FIPS .....                   | 17                                                         |
| FIPS requirements .....      | 3441                                                       |
| first open (of a file) ..... | 62                                                         |
| flockfile() .....            | <b>865</b> , 3535                                          |
| floor() .....                | <b>867</b>                                                 |
| floorf() .....               | 867                                                        |
| floorl() .....               | 867                                                        |
| flow control .....           | 62                                                         |
| FLT_ constants               |                                                            |
| defined in <float.h> .....   | <b>248</b>                                                 |
| FLT_DIG .....                | 249                                                        |
| FLT_EPSILON .....            | 250                                                        |
| FLT_EVAL_METHOD .....        | 247                                                        |
| FLT_MANT_DIG .....           | 248                                                        |
| FLT_MAX .....                | 250                                                        |
| FLT_MAX_10_EXP .....         | 250                                                        |
| FLT_MAX_EXP .....            | 249                                                        |
| FLT_MIN .....                | 250, 598, 600, 604, 606, 610, 770, 791, 793, 795, 824, 876 |
| FLT_MIN_10_EXP .....         | 249                                                        |
| FLT_MIN_EXP .....            | 249                                                        |
| FLT_RADIX .....              | 248, 1265                                                  |
| FLT_ROUNDS .....             | 247, 869                                                   |
| FLUSH .....                  | 475                                                        |
| FLUSHO .....                 | 477                                                        |
| FLUSHR .....                 | 367, 1133                                                  |
| FLUSHRW .....                | 367, 1133                                                  |
| FLUSHW .....                 | 367, 1133                                                  |
| fma() .....                  | <b>869</b>                                                 |
| fmaf() .....                 | 869                                                        |
| fmal() .....                 | 869                                                        |
| fmax() .....                 | <b>871</b>                                                 |
| fmaxf() .....                | 871                                                        |
| fmaxl() .....                | 871                                                        |

|                     |                                                           |
|---------------------|-----------------------------------------------------------|
| fmemopen()          | 872                                                       |
| fmin()              | 875                                                       |
| fminf()             | 875                                                       |
| fminl()             | 875                                                       |
| FMNAMESZ            | 366-367, 1132                                             |
| fmod()              | 876                                                       |
| fmodf()             | 876                                                       |
| fmodl()             | 876                                                       |
| fmtmsg()            | 878                                                       |
| fnmatch()           | 881, 3726                                                 |
| FNM_                | 475                                                       |
| FNM_ constants      |                                                           |
| in <fnmatch.h>      | 254                                                       |
| FNM_NOESCAPE        | 254, 881                                                  |
| FNM_NOMATCH         | 254, 881                                                  |
| FNM_PATHNAME        | 254, 881                                                  |
| FNM_PERIOD          | 254, 881                                                  |
| fold                | 2771, 3724                                                |
| fopen()             | 883, 3455, 3674, 3720                                     |
| FOPEN_MAX           | 272, 354, 826, 873, 885, 2139                             |
| for loop            | 2344, 3696                                                |
| foreground          | 1889, 3456-3459, 3509-3511                                |
| foreground job      | 62                                                        |
| foreground process  | 62                                                        |
| group               | 62                                                        |
| group ID            | 62                                                        |
| fork()              | 888, 3456, 3510, 3550, 3559, 3561, 3640, 3647, 3652, 3720 |
| forkall             | 891                                                       |
| form-feed character | 63                                                        |
| format of entries   | 219, 469                                                  |
| fort77              | 2775, 3726                                                |
| external symbols    | 2778                                                      |
| standard libraries  | 2777                                                      |
| fpathconf()         | 893, 3719, 3721                                           |
| fpclassify()        | 899                                                       |
| FPE_                | 475                                                       |
| FPE_FLTDIV          | 336                                                       |
| FPE_FLTINV          | 336                                                       |
| FPE_FLTOVF          | 336                                                       |
| FPE_FLTRES          | 336                                                       |
| FPE_FLTSUB          | 336                                                       |
| FPE_FLTUND          | 336                                                       |
| FPE_INTDIV          | 336                                                       |
| FPE_INTOVF          | 336                                                       |
| fprintf()           | 900                                                       |
| fputc()             | 913, 3721                                                 |
| fputs()             | 915                                                       |
| fputwc()            | 917                                                       |
| fputws()            | 919                                                       |
| FP_ILOGB0           | 1119                                                      |
| FP_ILOGBNAN         | 1119                                                      |
| FQDN                | 1045                                                      |

|                                  |                            |
|----------------------------------|----------------------------|
| FR.....                          | 7                          |
| frac_digits.....                 | 155                        |
| fread().....                     | 921, 3721                  |
| free().....                      | 923, 3543, 3616            |
| freeaddrinfo().....              | 924                        |
| freelocale().....                | 929                        |
| freopen().....                   | 931                        |
| frexp().....                     | 936                        |
| frexpf().....                    | 936                        |
| frexpl().....                    | 936                        |
| fsblkcnt_t.....                  | 401                        |
| FSC.....                         | 8                          |
| fscanf().....                    | 938                        |
| fseek().....                     | 946, 3721                  |
| fseeko().....                    | 946                        |
| fsetpos().....                   | 949, 3721                  |
| fsfilcnt_t.....                  | 401                        |
| fstat().....                     | 951, 3720                  |
| fstatat().....                   | 954                        |
| fstatvfs().....                  | 960                        |
| fsync().....                     | 963, 3554                  |
| ftell().....                     | 965                        |
| ftello(0.....                    | 965                        |
| ftime().....                     | 3650                       |
| ftok().....                      | 967                        |
| ftruncate().....                 | 969, 3560-3561, 3563, 3720 |
| ftrylockfile().....              | 865, 971                   |
| FTW.....                         | 255, 475, 1381-1382        |
| ftw().....                       | 972, 3674                  |
| FTW_ constants                   |                            |
| in <ftw.h>.....                  | 255                        |
| FTW_CHDIR.....                   | 255, 1381                  |
| FTW_D.....                       | 255, 972, 1381             |
| FTW_DEPTH.....                   | 255, 1381                  |
| FTW_DNR.....                     | 255, 972, 1381-1382        |
| FTW_DP.....                      | 255, 1381                  |
| FTW_F.....                       | 255, 972, 1381             |
| FTW_MOUNT.....                   | 255, 1381                  |
| FTW_NS.....                      | 255, 972, 1381-1382        |
| FTW_PHYS.....                    | 255, 1381                  |
| FTW_SL.....                      | 255, 972, 1381             |
| FTW_SLN.....                     | 255, 1382                  |
| fully-qualified domain name..... | 1045                       |
| function definition command..... | 2346, 3697                 |
| function identifiers.....        | 2498                       |
| functions.....                   | 471                        |
| implementation.....              | 471, 3525                  |
| use.....                         | 471, 3525                  |
| funlockfile().....               | 865, 975                   |
| fuser.....                       | 2781                       |
| futimens().....                  | 976                        |
| fwide().....                     | 980                        |



|                                     |                        |
|-------------------------------------|------------------------|
| fwprintf()                          | 982                    |
| fwrite()                            | 990, 3721              |
| fwscanf()                           | 992                    |
| f_                                  | 476                    |
| F_                                  | 477                    |
| F_DUPFD                             | 238, 813, 815          |
| F_DUPFD_CLOEXEC                     | 238, 813, 816          |
| F_GETFD                             | 238, 813, 816          |
| F_GETFL                             | 238, 813, 816          |
| F_GETLK                             | 238, 814, 816          |
| F_GETOWN                            | 238, 814, 816          |
| F_LOCK                              | 517, 1254              |
| F_OK                                | 440                    |
| F_RDLCK                             | 238, 816               |
| F_SETFD                             | 238, 813, 816          |
| F_SETFL                             | 238, 813, 816          |
| F_SETLK                             | 238, 814, 816          |
| F_SETLKW                            | 238, 517, 814, 816     |
| F_SETOWN                            | 238, 814, 816          |
| F_TEST                              | 443, 1254              |
| F_TLOCK                             | 443, 1254              |
| F_ULOCK                             | 443, 1254              |
| F_UNLCK                             | 238, 814-815           |
| F_WRLCK                             | 238                    |
| g-file                              | 2615                   |
| gai_strerror()                      | 999                    |
| gcvt()                              | 3650                   |
| gencat                              | 2784, 3725             |
| escape sequences                    | 2785                   |
| general terminal interface          | 3508                   |
| generated file                      | 2615                   |
| generation-version attribute        | 540, 1475              |
| get                                 | 2788                   |
| get configurable pathname variables | 895                    |
| get configurable system variables   | 2080                   |
| get file status                     | 957                    |
| get process times                   | 2136                   |
| get supplementary group IDs         | 1030                   |
| get system time                     | 2126                   |
| get thread ID                       | 1708                   |
| get user name                       | 1039                   |
| getaddrinfo()                       | 924, 1000              |
| GETALL                              | 381, 1848              |
| getc()                              | 1001, 3607, 3721       |
| getch()                             | 3721                   |
| getchar()                           | 1004                   |
| getchar_unlocked()                  | 1002, 1005             |
| getconf                             | 2796, 3667, 3719, 3726 |
| getcontext()                        | 3650                   |
| getcwd()                            | 1006                   |
| getc_unlocked()                     | 1002                   |
| getdate()                           | 1009                   |

|                          |                                      |
|--------------------------|--------------------------------------|
| getdate_err .....        | 1009                                 |
| getdelim() .....         | <b>1014</b>                          |
| getegid() .....          | <b>1016</b> , 3720                   |
| getenv() .....           | 786, 1017                            |
| geteuid() .....          | <b>1020</b> , 3720                   |
| getgid() .....           | <b>1021</b> , 3720                   |
| getgrent() .....         | 752, 1022, 3471                      |
| getgrgid() .....         | <b>1023</b> , 3471, 3608, 3720       |
| getgrgid_r() .....       | 1023                                 |
| getgrnam() .....         | <b>1027</b> , 3471, 3476, 3608, 3720 |
| getgrnam_r() .....       | 1027                                 |
| getgroups() .....        | <b>1030</b> , 3463                   |
| gethostbyaddr() .....    | 3650                                 |
| gethostbyname() .....    | 3650                                 |
| gethostent() .....       | 754, 1032                            |
| gethostid() .....        | <b>1033</b>                          |
| gethostname() .....      | <b>1034</b>                          |
| getitimer() .....        | <b>1035</b>                          |
| getline() .....          | 1014, 1037                           |
| getlogin() .....         | <b>1038</b> , 3720                   |
| getlogin_r() .....       | 1038                                 |
| getmsg() .....           | <b>1041</b>                          |
| getnameinfo() .....      | <b>1045</b>                          |
| GETNCNT .....            | 381, 1848-1849                       |
| getnetbyaddr() .....     | 756, 1048                            |
| getnetbyname() .....     | 756, 1048                            |
| getnetent() .....        | 756, 1048                            |
| getopt() .....           | <b>1049</b> , 3516, 3725-3726        |
| getopts .....            | 2318, 2339, 2802, 3675, 3726         |
| getpeername() .....      | <b>1053</b>                          |
| getpgid() .....          | <b>1055</b>                          |
| getpgrp() .....          | <b>1056</b> , 3458                   |
| GETPID .....             | 381, 1848-1849                       |
| getpid() .....           | <b>1057</b> , 3545, 3720             |
| getpmsg() .....          | 1041, 1058                           |
| getppid() .....          | <b>1059</b> , 3720                   |
| getpriority() .....      | <b>1060</b> , 3571                   |
| getprotent() .....       | 1063                                 |
| getprotobyname() .....   | 758, 1063                            |
| getprotobynumber() ..... | 758, 1063                            |
| getprotoent() .....      | 758                                  |
| getpwent() .....         | 760, 1064, 3471                      |
| getpwnam() .....         | <b>1065</b> , 3471, 3476, 3608, 3720 |
| getpwnam_r() .....       | 1065                                 |
| getpwuid() .....         | <b>1069</b> , 3471, 3608, 3720       |
| getpwuid_r() .....       | 1069                                 |
| getrlimit() .....        | <b>1073</b> , 3674                   |
| getrusage() .....        | <b>1076</b> , 3584                   |
| gets() .....             | <b>1078</b>                          |
| getservbyname() .....    | 763, 1080                            |
| getservbyport() .....    | 763, 1080                            |
| getservent() .....       | 763, 1080                            |

|                       |                        |
|-----------------------|------------------------|
| getsid()              | 1081                   |
| getsockname()         | 1082                   |
| getsockopt()          | 1084                   |
| getsubopt()           | 1086                   |
| gettimeofday()        | 1090                   |
| getty                 | 3510                   |
| getuid()              | 1091, 3545, 3648, 3720 |
| getutxent()           | 765, 1092              |
| getutxid()            | 765, 1092              |
| getutxline()          | 765, 1092              |
| GETVAL                | 381, 1848-1849         |
| getwc()               | 1093                   |
| getwchar()            | 1094                   |
| getwd()               | 3650                   |
| GETZCNT               | 381, 1848-1849         |
| gid_t                 | 401, 3471              |
| glob()                | 1095, 3726             |
| global storage class  | 2502                   |
| globfree()            | 1095, 3726             |
| GLOB_                 | 475                    |
| GLOB_ constants       |                        |
| defined in <glob.h>   | 257                    |
| error returns of glob | 1097                   |
| used in glob          | 1095                   |
| GLOB_ABORTED          | 257, 1097              |
| GLOB_APPEND           | 257, 1095-1096         |
| GLOB_DOOFFS           | 257, 1095-1096         |
| GLOB_ERR              | 257, 1095, 1097        |
| GLOB_MARK             | 257, 1095              |
| GLOB_NOCHECK          | 257, 1096-1097         |
| GLOB_NOESCAPE         | 257, 1096              |
| GLOB_NOMATCH          | 257, 1097              |
| GLOB_NOSORT           | 257, 1096              |
| GLOB_NOSPACE          | 257, 1097              |
| gl                    | 475                    |
| GMT0                  | 2162                   |
| gmtime()              | 1099, 3480             |
| gmtime_r()            | 1099                   |
| GNU make              | 2951                   |
| grammar               |                        |
| conventions           | 3670                   |
| locale                | 165                    |
| regular expression    | 191                    |
| grantpt()             | 1101                   |
| graphic character     | 63                     |
| grep                  | 2807, 3724-3725        |
| group database        | 63, 3455               |
| group database access | 3472                   |
| group file            | 3455                   |
| group ID              | 63                     |
| group name            | 63                     |
| grouping commands     | 2344, 3696             |

|                                  |                         |
|----------------------------------|-------------------------|
| HALT.....                        | 879                     |
| hard limit.....                  | 63                      |
| hard link .....                  | 64                      |
| hash .....                       | <b>2812</b>             |
| hcreate().....                   | <b>1103</b>             |
| hdestroy().....                  | 1103                    |
| head.....                        | <b>2815</b> , 3724      |
| headers.....                     | <b>219</b> , 3518       |
| here-document.....               | 2335, 3691              |
| high resolution sleep.....       | 1373                    |
| historical implementations ..... | 3455                    |
| history command                  |                         |
| fc .....                         | 2745                    |
| hold batch job request.....      | 2413                    |
| Hold_Types .....                 | 2407                    |
| HOME.....                        | <b>177</b> , 2679, 3441 |
| home directory.....              | 64                      |
| host byte order.....             | 64, 110, 3476           |
| host name .....                  | 924                     |
| hosted implementation .....      | 3455                    |
| hostent.....                     | <b>302</b>              |
| HOST_NAME_MAX .....              | 271                     |
| hsearch().....                   | 1103                    |
| htonl().....                     | <b>1106</b>             |
| htons().....                     | 1106                    |
| HUGE_VAL.....                    | 290, 2246               |
| HUGE_VALF .....                  | 290                     |
| HUGE_VALL .....                  | 290                     |
| hunk .....                       | 3022                    |
| HUPCL .....                      | 417                     |
| hypot().....                     | <b>1107</b>             |
| hypotf().....                    | 1107                    |
| hypotl().....                    | 1107                    |
| h_.....                          | 475                     |
| h_errno.....                     | 3650                    |
| I.....                           | 224                     |
| ICANON .....                     | 417, 3512, 3514         |
| iconv .....                      | <b>2818</b>             |
| iconv().....                     | <b>1109</b> , 3725      |
| iconv_close() .....              | <b>1112</b> , 3725      |
| iconv_open() .....               | <b>1113</b> , 3725      |
| ICRNL .....                      | 415                     |
| ic_.....                         | 475                     |
| id.....                          | <b>2822</b> , 3726      |
| idtype_t.....                    | <b>408</b>              |
| id_t.....                        | 401                     |
| IEEE Std 754-1985.....           | 469                     |
| IEEE Std 854-1987.....           | 469                     |
| IEXTEN.....                      | 417                     |
| if.....                          | 3697                    |
| if conditional construct.....    | 2345                    |
| ifc_.....                        | 476                     |

|                               |              |
|-------------------------------|--------------|
| ifra_.....                    | 476          |
| ifru_.....                    | 476          |
| IF_.....                      | 475          |
| if_.....                      | 475-476      |
| if_freenameindex().....       | <b>1115</b>  |
| if_indextoname().....         | <b>1116</b>  |
| if_nameindex.....             | <b>301</b>   |
| if_nameindex().....           | <b>1117</b>  |
| IF_NAMESIZE.....              | 301          |
| if_nametoindex().....         | <b>1118</b>  |
| IGNBRK.....                   | 415          |
| IGNCR.....                    | 415          |
| IGNPAR.....                   | 415          |
| ILL_.....                     | 475          |
| ILL_BADSTK.....               | 336          |
| ILL_COPROC.....               | 336          |
| ILL_ILLADR.....               | 336          |
| ILL_ILLOPC.....               | 336          |
| ILL_ILLOPN.....               | 336          |
| ILL_ILLTRP.....               | 336          |
| ILL_PRVOPC.....               | 336          |
| ILL_PRVREG.....               | 336          |
| ilogb().....                  | <b>1119</b>  |
| ilogbf().....                 | 1119         |
| ilogbl().....                 | 1119         |
| imaginary.....                | 224          |
| imaxabs().....                | <b>1121</b>  |
| imaxdiv().....                | <b>1122</b>  |
| implementation.....           | 3455         |
| historical.....               | 3455         |
| hosted.....                   | 3455         |
| native.....                   | 3460         |
| specific.....                 | 3455         |
| implementation-defined.....   | 5, 3442-3443 |
| rationale.....                | 3442         |
| IMPLINK_.....                 | 477          |
| in6_.....                     | 475          |
| IN6_.....                     | 477          |
| IN6_IS_ADDR_LINKLOCAL.....    | 308          |
| IN6_IS_ADDR_LOOPBACK.....     | 308          |
| IN6_IS_ADDR_MC_GLOBAL.....    | 308          |
| IN6_IS_ADDR_MC_LINKLOCAL..... | 308          |
| IN6_IS_ADDR_MC_NODELOCAL..... | 308          |
| IN6_IS_ADDR_MC_ORGLOCAL.....  | 308          |
| IN6_IS_ADDR_MC_SITELOCAL..... | 308          |
| IN6_IS_ADDR_MULTICAST.....    | 308          |
| IN6_IS_ADDR_SITELOCAL.....    | 308          |
| IN6_IS_ADDR_UNSPECIFIED.....  | 308          |
| IN6_IS_ADDR_V4COMPAT.....     | 308          |
| IN6_IS_ADDR_V4MAPPED.....     | 308          |
| INADDR_.....                  | 475          |
| INADDR_ANY.....               | 307          |

|                                     |                        |
|-------------------------------------|------------------------|
| INADDR_BROADCAST .....              | 307                    |
| include line .....                  | 2938                   |
| incomplete line .....               | 64                     |
| incomplete pathname .....           | 3455                   |
| index() .....                       | 3650                   |
| INET6_ADDRSTRLEN .....              | 308                    |
| inet_ .....                         | 475                    |
| inet_addr() .....                   | <b>1123</b>            |
| INET_ADDRSTRLEN .....               | 307                    |
| inet_ntoa() .....                   | 1123                   |
| inet_ntop() .....                   | <b>1125</b>            |
| inet_pton() .....                   | 1125                   |
| Inf .....                           | 64, 598, 600, 604, 610 |
| INF .....                           | 903, 985               |
| inference rule .....                | 2934                   |
| INFINITY .....                      | 290, 903, 985          |
| INFO .....                          | 879                    |
| infu_ .....                         | 476                    |
| inheritance attribute .....         | 540, 1477              |
| init .....                          | 552, 1212              |
| initialize a named semaphore .....  | 1836                   |
| initializing a mutex .....          | 1644                   |
| initstate() .....                   | <b>1127</b>            |
| INIT_PROCESS .....                  | 455, 765-766           |
| INLCR .....                         | 415                    |
| ino_t .....                         | 401                    |
| INPCK .....                         | 415                    |
| input and output rationale .....    | 1755                   |
| input file descriptor               |                        |
| duplication .....                   | 3691                   |
| input mode .....                    | 2642, 3513             |
| input processing .....              | 3511                   |
| canonical mode .....                | 3511                   |
| non-canonical mode .....            | 3512                   |
| insque() .....                      | <b>1129</b>            |
| instrumented application .....      | 64                     |
| INT .....                           | 477                    |
| inter-user communication .....      | 3718, 3725             |
| interactive facilities .....        | 3717, 3724             |
| interactive shell .....             | 64                     |
| interface .....                     | 3620                   |
| characteristics .....               | 3509                   |
| international environment .....     | 1884                   |
| internationalization .....          | 65                     |
| internationalization variable ..... | 3497                   |
| Internet Protocols .....            | 530                    |
| interprocess communication .....    | 65, 3546               |
| INTMAX_MAX .....                    | 351                    |
| INTMAX_MIN .....                    | 351                    |
| INTN_MAX .....                      | 350                    |
| INTN_MIN .....                      | 350                    |
| INTPTR_MAX .....                    | 351                    |

|                              |                                                                |
|------------------------------|----------------------------------------------------------------|
| INTPTR_MIN .....             | 350                                                            |
| int_curr_symbol .....        | 154                                                            |
| INT_FASTN_MAX .....          | 350                                                            |
| INT_FASTN_MIN .....          | 350                                                            |
| int_frac_digits .....        | 155                                                            |
| INT_LEASTN_MAX .....         | 350                                                            |
| INT_LEASTN_MIN .....         | 350                                                            |
| INT_MAX .....                | 280, 1119                                                      |
| INT_MIN .....                | 280, 562                                                       |
| int_n_cs_precedes .....      | 156                                                            |
| int_n_sep_by_space .....     | 156                                                            |
| int_n_sign_posn .....        | 156                                                            |
| int_p_cs_precedes .....      | 156                                                            |
| int_p_sep_by_space .....     | 156                                                            |
| int_p_sign_posn .....        | 156                                                            |
| invalid .....                | 182                                                            |
| use in RE .....              | 3501                                                           |
| invariant values .....       | 282                                                            |
| invoke .....                 | 65                                                             |
| in_ .....                    | 475                                                            |
| IN_ .....                    | 477                                                            |
| in_addr .....                | 306                                                            |
| ioctl() .....                | 1132, 3508, 3533                                               |
| iovec .....                  | 405                                                            |
| iov_ .....                   | 476                                                            |
| IOV_ .....                   | 477                                                            |
| IOV_MAX .....                | 271, 405, 1767, 2077, 2294                                     |
| IP6 .....                    | 8                                                              |
| IPC .....                    | 370, 501, 1356, 1358, 1361, 1363, 1852, 1857, 1924, 1927, 3546 |
| ipcrm .....                  | 2826                                                           |
| ipcs .....                   | 2828                                                           |
| IPC_ .....                   | 475                                                            |
| ipc_ .....                   | 475                                                            |
| IPC_ constants               |                                                                |
| defined in <sys/ipc.h> ..... | 370                                                            |
| used in semctl .....         | 1849                                                           |
| used in shmctl .....         | 1922                                                           |
| IPC_CREAT .....              | 370, 1357, 1851, 1926                                          |
| IPC_EXCL .....               | 370, 1357, 1851                                                |
| IPC_NOWAIT .....             | 370, 1359-1360, 1362-1363, 1853                                |
| IPC_PRIVATE .....            | 370, 1357, 1851, 1926                                          |
| IPC_RMID .....               | 370, 1355, 1849, 1922                                          |
| IPC_SET .....                | 370, 1355, 1849, 1922                                          |
| IPC_STAT .....               | 370, 1355, 1849, 1922                                          |
| IPPORT_ .....                | 477                                                            |
| IPPROTO_ .....               | 475                                                            |
| IPPROTO_ICMP .....           | 307                                                            |
| IPPROTO_IP .....             | 307                                                            |
| IPPROTO_IPV6 .....           | 307                                                            |
| IPPROTO_RAW .....            | 307                                                            |
| IPPROTO_TCP .....            | 307                                                            |
| IPPROTO_UDP .....            | 307                                                            |

|                                 |          |
|---------------------------------|----------|
| IPv4.....                       | 531      |
| IPv4-compatible address.....    | 532      |
| IPv4-mapped address.....        | 532      |
| IPv6.....                       | 531      |
| compatibility with IPv4.....    | 532      |
| interface identification.....   | 533      |
| options .....                   | 533      |
| IPv6 address                    |          |
| anycast .....                   | 531      |
| loopback .....                  | 532      |
| multicast .....                 | 532      |
| unicast.....                    | 531      |
| unspecified .....               | 532      |
| IPV6_.....                      | 475      |
| IPV6_JOIN_GROUP .....           | 308, 533 |
| IPV6_LEAVE_GROUP.....           | 308, 533 |
| ipv6_mreq.....                  | 307      |
| IPV6_MULTICAST_HOPS.....        | 308, 533 |
| IPV6_MULTICAST_IF.....          | 308, 533 |
| IPV6_MULTICAST_LOOP .....       | 308, 533 |
| IPV6_UNICAST_HOPS.....          | 308, 533 |
| IPV6_V6ONLY.....                | 308, 534 |
| ip_.....                        | 475      |
| IP_.....                        | 477      |
| isalnum() .....                 | 1144     |
| isalnum_l() .....               | 1144     |
| isalpha().....                  | 1146     |
| isalpha_l().....                | 1146     |
| isascii().....                  | 1148     |
| isastream().....                | 1149     |
| isatty().....                   | 1150     |
| isblank().....                  | 1151     |
| isblank_l().....                | 1151     |
| iscntrl() .....                 | 1153     |
| iscntrl_l() .....               | 1153     |
| isdigit() .....                 | 1155     |
| isdigit_l() .....               | 1155     |
| isfinite() .....                | 1157     |
| isgraph() .....                 | 1158     |
| isgraph_l().....                | 1158     |
| isgreater() .....               | 1160     |
| isgreaterequal() .....          | 1161     |
| ISIG.....                       | 417      |
| isinf().....                    | 1162     |
| isless() .....                  | 1163     |
| islessequal() .....             | 1164     |
| islessgreater().....            | 1165     |
| islower() .....                 | 1166     |
| islower_l() .....               | 1166     |
| isnan().....                    | 1169     |
| isnormal().....                 | 1170     |
| ISO/IEC 646: 1991 standard..... | 3462     |



|                      |                                                                         |
|----------------------|-------------------------------------------------------------------------|
| ISO C standard ..... | 224, 469, 590, 785, 818, 1003, 1277, 1749, 1800, 1884, 1935, 1951, 1962 |
| .....                | 2126, 3449, 3451, 3480, 3508, 3525                                      |
| isprint().....       | <b>1171</b>                                                             |
| isprint_l().....     | 1171                                                                    |
| ispunct() .....      | <b>1173</b>                                                             |
| ispunct_l() .....    | 1173                                                                    |
| isspace().....       | <b>1175</b>                                                             |
| isspace_l().....     | 1175                                                                    |
| ISTRIP .....         | 415, 3513                                                               |
| isunordered().....   | <b>1177</b>                                                             |
| isupper().....       | <b>1178</b>                                                             |
| isupper_l().....     | 1178                                                                    |
| iswalnum().....      | <b>1180</b>                                                             |
| iswalnum_l().....    | 1180                                                                    |
| iswalpha() .....     | <b>1182</b>                                                             |
| iswalpha_l() .....   | 1182                                                                    |
| iswblank() .....     | <b>1184</b>                                                             |
| iswblank_l() .....   | 1184                                                                    |
| iswcntrl().....      | <b>1186</b>                                                             |
| iswcntrl_l().....    | 1186                                                                    |
| iswctype().....      | <b>1188</b>                                                             |
| iswctype_l() .....   | 1188                                                                    |
| iswdigit().....      | <b>1190</b>                                                             |
| iswdigit_l().....    | 1190                                                                    |
| iswgraph().....      | <b>1192</b>                                                             |
| iswgraph_l().....    | 1192                                                                    |
| iswlower().....      | <b>1194</b>                                                             |
| iswlower_l().....    | 1194                                                                    |
| iswprint() .....     | <b>1196</b>                                                             |
| iswprint_l() .....   | 1196                                                                    |
| iswpunct().....      | <b>1198</b>                                                             |
| iswpunct_l().....    | 1198                                                                    |
| iswspace() .....     | <b>1200</b>                                                             |
| iswspace_l() .....   | 1200                                                                    |
| iswupper() .....     | <b>1202</b>                                                             |
| iswupper_l() .....   | 1202                                                                    |
| iswxdigit().....     | <b>1204</b>                                                             |
| iswxdigit_l().....   | 1204                                                                    |
| isxdigit() .....     | <b>1206</b>                                                             |
| isxdigit_l().....    | 1206                                                                    |
| itimerspec .....     | <b>424</b>                                                              |
| itimerval .....      | <b>398</b>                                                              |
| ITIMER_.....         | 476                                                                     |
| ITIMER_PROF.....     | 398, 1035                                                               |
| ITIMER_REAL.....     | 398, 1035                                                               |
| ITIMER_VIRTUAL.....  | 398, 1035                                                               |
| it_.....             | 476                                                                     |
| IXANY .....          | 415                                                                     |
| IXOFF .....          | 415                                                                     |
| IXON .....           | 415                                                                     |
| I_.....              | 475                                                                     |
| I_ATMARK.....        | 366, 1139                                                               |

|                                 |                                                                          |
|---------------------------------|--------------------------------------------------------------------------|
| I_CANPUT .....                  | 366, 1139                                                                |
| I_CKBAND .....                  | 366, 1139                                                                |
| I_FDINSERT .....                | 366, 1135                                                                |
| I_FIND .....                    | 366, 1134                                                                |
| I_FLUSH .....                   | 366, 1132                                                                |
| I_FLUSHBAND .....               | 366, 1133                                                                |
| I_GETBAND .....                 | 366, 1139                                                                |
| I_GETCLTIME .....               | 366, 1140                                                                |
| I_GETSIG .....                  | 366, 1134                                                                |
| I_GRDOPT .....                  | 366, 1135, 1753                                                          |
| I_GWROPT .....                  | 366, 1137                                                                |
| I_ISVTX .....                   | 2542                                                                     |
| I_LINK .....                    | 366, 1140                                                                |
| I_LIST .....                    | 366, 1138                                                                |
| I_LOOK .....                    | 366, 1132                                                                |
| I_NREAD .....                   | 366, 1135                                                                |
| I_PEEK .....                    | 366, 1134                                                                |
| I_PLINK .....                   | 366, 1141                                                                |
| I_POP .....                     | 366, 1132                                                                |
| I_PUNLINK .....                 | 366, 1141                                                                |
| I_PUSH .....                    | 366, 1132                                                                |
| I_RECVFD .....                  | 366, 482, 1138                                                           |
| I_SENDFD .....                  | 366, 1137-1138                                                           |
| I_SETCLTIME .....               | 366, 681, 1139                                                           |
| I_SETSIG .....                  | 366, 1133-1134                                                           |
| I_SRDOPT .....                  | 366, 1135, 1753                                                          |
| I_STR .....                     | 366, 1136                                                                |
| I_SWROPT .....                  | 367, 1137, 2287                                                          |
| I_UNLINK .....                  | 367, 1140                                                                |
| j0() .....                      | <b>1208</b>                                                              |
| j1() .....                      | 1208                                                                     |
| jn() .....                      | 1208                                                                     |
| job .....                       | 65                                                                       |
| job control.....                | 65, 552, 1056, 1212, 1889, 1902, 2080, 2209, 3456-3459, 3461, 3509, 3511 |
| .....                           | 3536, 3542, 3720                                                         |
| implementing applications ..... | 3458                                                                     |
| implementing shells.....        | 3456                                                                     |
| implementing systems.....       | 3458                                                                     |
| job control job ID .....        | 65                                                                       |
| jobs.....                       | 2318, 2339, 2835, 3675, 3724                                             |
| Job_Owner.....                  | 2407                                                                     |
| join .....                      | <b>2839</b> , 3724                                                       |
| Join_Path.....                  | 2407                                                                     |
| jr48() .....                    | 743, 1210                                                                |
| JST-9.....                      | 2162                                                                     |
| Keep_Files .....                | 2407                                                                     |
| kernel .....                    | 3459                                                                     |
| kernel entity .....             | 3610                                                                     |
| keyword-value pairs.....        | 2421                                                                     |
| key_t.....                      | 401                                                                      |
| kill .....                      | 2318, 2339, 2844, 3675, 3724                                             |
| kill() .....                    | <b>1211</b> , 3535-3537, 3540, 3542, 3647                                |

|                        |                                                                          |
|------------------------|--------------------------------------------------------------------------|
| killpg()               | 1214                                                                     |
| l64a()                 | 558, 1216                                                                |
| labs()                 | 1217                                                                     |
| LANG                   | 174, 644                                                                 |
| last close             | 1918, 3561                                                               |
| last close (of a file) | 66                                                                       |
| LASTMARK               | 368, 1139                                                                |
| lchown()               | 1218                                                                     |
| lcong48()              | 743, 1221                                                                |
| LC_ALL                 | 175, 286, 779, 1247, 1387, 1883, 1885                                    |
| LC_COLLATE             | 175, 275, 286, 1095-1096, 1883, 1885, 2006, 2068, 2224, 2265, 3488       |
| description            | 147                                                                      |
| LC_CTYPE               | 175, 267, 286, 462, 1188, 1282, 1291, 1293, 1883, 1885, 2144, 2150, 3487 |
| description            | 139                                                                      |
| LC_GLOBAL_LOCALE       | 748, 929                                                                 |
| LC_MESSAGES            | 175, 267, 286, 311, 644, 1883-1885, 2014, 3493                           |
| description            | 164                                                                      |
| LC_MONETARY            | 175, 267, 286, 1247, 1883, 1885, 2019, 3490                              |
| description            | 154                                                                      |
| LC_NUMERIC             | 175, 267, 286, 901, 938, 982, 992, 1247, 1883, 1885, 2019, 2051, 2246    |
| description            | 3491, 3516                                                               |
| LC_TIME                | 175, 267, 286, 1010, 1388, 1883, 1885, 3491                              |
| description            | 159                                                                      |
| ld                     |                                                                          |
| rationale for omission | 3710                                                                     |
| LDBL_constants         |                                                                          |
| defined in <float.h>   | 248                                                                      |
| LDBL_DIG               | 249                                                                      |
| LDBL_EPSILON           | 250                                                                      |
| LDBL_MANT_DIG          | 248                                                                      |
| LDBL_MAX               | 250                                                                      |
| LDBL_MAX_10_EXP        | 250                                                                      |
| LDBL_MAX_EXP           | 249                                                                      |
| LDBL_MIN               | 250, 598, 600, 604, 606, 610, 770, 791, 793, 795, 824, 876               |
| LDBL_MIN_10_EXP        | 249                                                                      |
| LDBL_MIN_EXP           | 249                                                                      |
| ldexp()                | 1222                                                                     |
| ldexpf()               | 1222                                                                     |
| ldexpl()               | 1222                                                                     |
| ldiv()                 | 1224                                                                     |
| legacy                 | 5, 3443                                                                  |
| rationale              | 3443                                                                     |
| lex                    | 2849, 3726, 3728                                                         |
| actions                | 2855                                                                     |
| definitions            | 2852                                                                     |
| escape sequences       | 2854                                                                     |
| regular expressions    | 2853                                                                     |
| rules                  | 2853                                                                     |
| table sizes            | 2852                                                                     |
| translation table      | 2860                                                                     |
| user subroutines       | 2853                                                                     |

|                        |                                                                       |
|------------------------|-----------------------------------------------------------------------|
| lfind()                | 1225, 1275                                                            |
| lgamma()               | 1226                                                                  |
| lgammaf()              | 1226                                                                  |
| lgammal()              | 1226                                                                  |
| libraries              |                                                                       |
| ar command             | 2432                                                                  |
| library routine        | 3459                                                                  |
| LIMIT                  | 2307                                                                  |
| limit                  |                                                                       |
| numerical              | 280                                                                   |
| limits                 | 3667                                                                  |
| line                   | 66                                                                    |
| rationale for omission | 3710                                                                  |
| line counting          | 3395                                                                  |
| LINES                  | 177, 3498                                                             |
| LINE_MAX               | 275, 2077, 2308, 2453, 2655, 2663, 2930, 3187, 3311, 3460, 3470, 3668 |
| linger                 | 66                                                                    |
| link                   | 66, 2861                                                              |
| link count             | 66                                                                    |
| link to a file         | 1230                                                                  |
| link()                 | 1228, 3453, 3721                                                      |
| linkat()               | 1228                                                                  |
| LINK_MAX               | 273, 484, 893, 1229, 1798, 3668, 3735                                 |
| lint                   |                                                                       |
| rationale for omission | 3710                                                                  |
| LIO_                   | 475                                                                   |
| lio_                   | 475                                                                   |
| lio_listio()           | 1233, 3539, 3555                                                      |
| LIO_NOP                | 220, 1233                                                             |
| LIO_NOWAIT             | 220, 1233                                                             |
| LIO_READ               | 220, 1233                                                             |
| LIO_WAIT               | 220, 1233                                                             |
| LIO_WRITE              | 220, 1233                                                             |
| list directed I/O      | 1235                                                                  |
| listen()               | 1237                                                                  |
| lists                  | 2341, 3695                                                            |
| AND-OR                 | 2341                                                                  |
| compound-list          | 2341                                                                  |
| llabs()                | 1217, 1239                                                            |
| lldiv()                | 1224, 1240                                                            |
| LLONG_MAX              | 280, 2060, 2253                                                       |
| LLONG_MIN              | 280, 2060, 2253                                                       |
| llrint()               | 1241                                                                  |
| llrintf()              | 1241                                                                  |
| llrintl()              | 1241                                                                  |
| llround()              | 1243                                                                  |
| llroundf()             | 1243                                                                  |
| llroundl()             | 1243                                                                  |
| ln                     | 2863, 3674, 3725                                                      |
| LNKTYPE                | 412                                                                   |
| load order             | 738                                                                   |
| LOBLK                  | 477                                                                   |

|                                    |                            |
|------------------------------------|----------------------------|
| local customs.....                 | 66                         |
| local IPC.....                     | 66                         |
| local mode.....                    | 3514                       |
| locale.....                        | 67, 135, 2868, 3486, 3725  |
| configuration.....                 | 3718, 3725                 |
| definition.....                    | 136, 3486                  |
| definition example.....            | 3494                       |
| definition grammar.....            | 3493                       |
| grammar.....                       | 165, 3493                  |
| lexical conventions.....           | 3493                       |
| POSIX.....                         | 136                        |
| localeconv().....                  | <b>1245</b>                |
| localedef.....                     | <b>2874</b> , 3725-3726    |
| localization.....                  | 67                         |
| localtime().....                   | <b>1250</b> , 3480, 3606   |
| localtime_r().....                 | 1250                       |
| locate batch job request.....      | 2414                       |
| lockf().....                       | <b>1254</b>                |
| locking.....                       | 818                        |
| advisory.....                      | 819                        |
| mandatory.....                     | 819                        |
| locking and unlocking a mutex..... | 1655                       |
| locking file.....                  | 2544                       |
| log().....                         | <b>1257</b>                |
| log-full-policy attribute.....     | 538, 540, 1477, 1480, 1497 |
| log-max-size attribute.....        | 540, 1478, 1480            |
| log10().....                       | <b>1259</b>                |
| log10f().....                      | 1259                       |
| log10l().....                      | 1259                       |
| log1p().....                       | <b>1261</b>                |
| log1pf().....                      | 1261                       |
| log1pl().....                      | 1261                       |
| log2().....                        | <b>1263</b>                |
| log2f().....                       | 1263                       |
| log2l().....                       | 1263                       |
| logb().....                        | <b>1265</b>                |
| logbf().....                       | 1265                       |
| logbl().....                       | 1265                       |
| logf().....                        | 1257, 1267                 |
| logger.....                        | <b>2878</b> , 3726         |
| logical device.....                | 3459                       |
| login.....                         | 67                         |
| rationale for omission.....        | 3710                       |
| login name.....                    | 67                         |
| login shell.....                   | 785                        |
| LOGIN_NAME_MAX.....                | 271, 1038, 2077, 3735      |
| LOGIN_PROCESS.....                 | 455, 765-766               |
| logl().....                        | 1257, 1267                 |
| LOGNAME.....                       | <b>178</b>                 |
| logname.....                       | <b>2881</b>                |
| LOGNAME.....                       | 3441, 3499                 |
| logname.....                       | 3726                       |

|                              |                                            |
|------------------------------|--------------------------------------------|
| LOG_.....                    | 476                                        |
| LOG_constants in syslog..... | <b>687</b>                                 |
| LOG_ALERT .....              | 411, 687                                   |
| LOG_AUTH.....                | 410                                        |
| LOG_CONS.....                | 410, 688                                   |
| LOG_CRIT .....               | 411, 687                                   |
| LOG_CRON.....                | 410                                        |
| LOG_DAEMON .....             | 410                                        |
| LOG_DEBUG .....              | 411, 687                                   |
| LOG_EMERG.....               | 411, 687                                   |
| LOG_ERR .....                | 411, 687                                   |
| LOG_INFO .....               | 411, 687                                   |
| LOG_KERN.....                | 410                                        |
| LOG_LOCAL .....              | 410, 687                                   |
| LOG_LPR .....                | 410                                        |
| LOG_MAIL .....               | 410                                        |
| LOG_MASK.....                | 410                                        |
| LOG_NDELAY .....             | 410, 688                                   |
| LOG_NEWS.....                | 410                                        |
| LOG_NOTICE.....              | 411, 687                                   |
| LOG_NOWAIT .....             | 410, 688                                   |
| LOG_ODELAY .....             | 410, 688                                   |
| LOG_PID.....                 | 410, 688                                   |
| LOG_USER .....               | 410, 687-688                               |
| LOG_UUCP .....               | 410                                        |
| LOG_WARNING.....             | 411, 687                                   |
| longjmp() .....              | <b>1268</b> , 3532, 3543, 3614, 3616, 3723 |
| LONG_BIT .....               | 280                                        |
| LONG_MAX .....               | 281, 2060, 2253, 3515                      |
| LONG_MIN .....               | 281, 2060, 2253, 3515                      |
| lorder                       |                                            |
| rationale for omission.....  | 3710                                       |
| lower multiplexing.....      | 92                                         |
| lp.....                      | <b>2883</b> , 3726                         |
| lpstat                       |                                            |
| rationale for omission.....  | 3710                                       |
| LR(1) grammars.....          | 3429                                       |
| lrand48().....               | 743, 1270                                  |
| lrint().....                 | <b>1271</b>                                |
| lrintf() .....               | 1271                                       |
| lrintl().....                | 1271                                       |
| lround() .....               | <b>1273</b>                                |
| lroundf().....               | 1273                                       |
| lroundl() .....              | 1273                                       |
| ls.....                      | <b>2888</b> , 3674, 3725                   |
| lsearch() .....              | <b>1275</b>                                |
| lseek().....                 | <b>1277</b> , 3554-3555, 3561, 3605, 3721  |
| lstat().....                 | 954, 1279, 3674, 3720                      |
| l_.....                      | 475-476                                    |
| L_ANCHOR.....                | 192                                        |
| L_ctermid .....              | 354, 718                                   |
| l_sysid .....                | 819                                        |

|                                       |                        |
|---------------------------------------|------------------------|
| L_tmpnam .....                        | 354                    |
| m4 .....                              | <b>2898</b>            |
| macro .....                           | 3444                   |
| macro processor .....                 | 2898                   |
| MAGIC .....                           | 227                    |
| magic file .....                      | 2758                   |
| mail                                  |                        |
| rationale for omission .....          | 3710                   |
| mailx .....                           | <b>2908, 3724-3725</b> |
| change current directory .....        | 2919                   |
| change folder .....                   | 2921                   |
| command escapes .....                 | 2927                   |
| commands .....                        | 2918                   |
| copy messages .....                   | 2919                   |
| declare aliases .....                 | 2919                   |
| declare alternatives .....            | 2919                   |
| delete aliases .....                  | 2925                   |
| delete messages .....                 | 2920                   |
| delete messages and display .....     | 2920                   |
| direct messages to mbox .....         | 2923                   |
| discard header fields .....           | 2920                   |
| display beginning of messages .....   | 2925                   |
| display current message number .....  | 2927                   |
| display header summary .....          | 2922                   |
| display list of folders .....         | 2921                   |
| display message .....                 | 2923                   |
| display message size .....            | 2925                   |
| echo a string .....                   | 2920                   |
| edit message .....                    | 2920, 2926             |
| execute commands conditionally .....  | 2922                   |
| exit .....                            | 2921                   |
| follow up specified messages .....    | 2921                   |
| help .....                            | 2922                   |
| hold messages .....                   | 2922                   |
| internal variables .....              | 2915                   |
| invoke a shell .....                  | 2925                   |
| invoke shell command .....            | 2926                   |
| list available commands .....         | 2922                   |
| mail a message .....                  | 2923                   |
| null command .....                    | 2927                   |
| pipe message .....                    | 2923                   |
| process next specified message .....  | 2923                   |
| quit .....                            | 2924                   |
| read mailx commands from a file ..... | 2925                   |
| receive mode .....                    | 2908                   |
| reply to a message .....              | 2924                   |
| reply to a message list .....         | 2924                   |
| retain header fields .....            | 2924                   |
| save messages .....                   | 2924                   |
| scroll header display .....           | 2926                   |
| send mode .....                       | 2908                   |
| set variables .....                   | 2925                   |

|                               |                                                             |
|-------------------------------|-------------------------------------------------------------|
| start-up.....                 | 2915                                                        |
| touch messages.....           | 2925                                                        |
| undelete messages.....        | 2926                                                        |
| unset variables.....          | 2926                                                        |
| write messages to a file..... | 2926                                                        |
| Mail_Points .....             | 2408                                                        |
| Mail_Users .....              | 2408                                                        |
| make.....                     | <b>2934</b> , 3726-3727                                     |
| default rules .....           | 2945                                                        |
| inference rules.....          | 2942                                                        |
| internal macros .....         | 2943                                                        |
| libraries .....               | 2943                                                        |
| macros.....                   | 2940                                                        |
| makefile execution .....      | 2938                                                        |
| makefile syntax.....          | 2937                                                        |
| target rules.....             | 2939                                                        |
| make, GNU version .....       | 2951                                                        |
| makecontext() .....           | 3650-3651                                                   |
| malloc() .....                | <b>1280</b> , 3543, 3564, 3566, 3594, 3606-3607, 3616, 3618 |
| man.....                      | <b>2956</b> , 3724                                          |
| manipulate signal sets .....  | 1941                                                        |
| map.....                      | 67, 3459                                                    |
| mapped.....                   | 3459                                                        |
| mappings.....                 | 1327                                                        |
| MAP_ .....                    | 475                                                         |
| MAP_FAILED .....              | 1328                                                        |
| MAP_FIXED .....               | 372, 1323, 1326                                             |
| MAP_PRIVATE.....              | 372, 888, 1323, 1327, 1332, 1365                            |
| MAP_SHARED .....              | 372, 891, 1323-1324                                         |
| margin code.....              | 3445                                                        |
| notation.....                 | 13, 3445                                                    |
| marked message.....           | 67                                                          |
| matched .....                 | 68, 181                                                     |
| mathematical functions .....  | 2307                                                        |
| domain error .....            | 116                                                         |
| error conditions .....        | 116, 3481                                                   |
| NaN arguments .....           | 118, 3481                                                   |
| pole error .....              | 117                                                         |
| range error.....              | 117                                                         |
| max-data-size attribute.....  | 540, 1480                                                   |
| MAXARGS .....                 | 343                                                         |
| MAXFLOAT .....                | 290                                                         |
| maximum values.....           | 276                                                         |
| MAX_CANON .....               | 273, 893, 3512, 3668, 3735                                  |
| MAX_INPUT .....               | 274, 893, 3668, 3735                                        |
| may.....                      | 5, 3442                                                     |
| rationale.....                | 3442                                                        |
| mblen() .....                 | <b>1282</b>                                                 |
| mbrlen().....                 | <b>1284</b>                                                 |
| mbrtowc() .....               | <b>1286</b>                                                 |
| mbsinit().....                | <b>1288</b>                                                 |
| mbsnrtowcs().....             | 1289                                                        |



|                         |                                         |
|-------------------------|-----------------------------------------|
| mbsrtowcs()             | 1289                                    |
| mbstowcs()              | 1291                                    |
| mbtowc()                | 1293                                    |
| MB_CUR_MAX              | 358, 1282, 1284, 1286, 1293, 2217, 2267 |
| MB_LEN_MAX              | 280-281                                 |
| MC1                     | 8                                       |
| MCL_                    | 475                                     |
| MCL_CURRENT             | 372, 1320                               |
| MCL_FUTURE              | 372, 1320, 3558                         |
| MCL_INHERIT             | 3559                                    |
| mcontext_t              | 334                                     |
| memccpy()               | 1295                                    |
| memchr()                | 1296                                    |
| memcmp()                | 1297                                    |
| memcpy()                | 1298                                    |
| MEMLOCK_FUTURE          | 1328                                    |
| memmove()               | 1299                                    |
| memory locking          | 3557                                    |
| memory management       | 504, 3556, 3720                         |
| memory management unit  | 3557                                    |
| memory mapped files     | 68                                      |
| memory object           | 68, 3459                                |
| memory synchronization  | 110, 3476                               |
| memory-resident         | 68, 3459                                |
| memset()                | 1300                                    |
| mesg                    | 2960, 3725-3726                         |
| message                 | 68                                      |
| message catalog         | 68                                      |
| descriptor              | 69, 549, 779, 784                       |
| generation              | 2784                                    |
| message parts           | 500                                     |
| message passing         | 3548, 3720, 3722                        |
| message priority        | 500                                     |
| high-priority           | 500                                     |
| normal                  | 500                                     |
| priority                | 500                                     |
| message queue           | 69, 3548                                |
| message-discard mode    | 1753                                    |
| message-nondiscard mode | 1753                                    |
| MET-1MEST               | 2162                                    |
| META_CHAR               | 191                                     |
| MIL-STD-1753            | 2779                                    |
| minimum values          | 276                                     |
| Minimum_Cpu_Interval    | 2405                                    |
| MINSIGSTKSZ             | 334, 1938                               |
| mkdir                   | 2963, 3725                              |
| mkdir()                 | 1301, 3721                              |
| mkdirat()               | 1301                                    |
| mkdtemp()               | 1304                                    |
| mkfifo                  | 2966, 3725                              |
| mkfifo()                | 1307, 3455                              |
| mkfifoat()              | 1307                                    |

|                                |                               |
|--------------------------------|-------------------------------|
| mknod                          |                               |
| rationale for omission .....   | 3710                          |
| mknod() .....                  | <b>1311</b> , 3455            |
| mknodat() .....                | 1311                          |
| mkstemp() .....                | 1304, 1315                    |
| mktemp() .....                 | 3651                          |
| mktime() .....                 | <b>1316</b> , 3480            |
| ML .....                       | <b>8</b>                      |
| mlock() .....                  | <b>1318</b>                   |
| mlockall() .....               | <b>1320</b> , 3558            |
| MLR .....                      | <b>8</b>                      |
| mmap() .....                   | <b>1322</b> , 3560-3562, 3564 |
| MMU .....                      | 3557                          |
| MM_ .....                      | 475                           |
| MM_ macros .....               | <b>252</b>                    |
| MM_APPL .....                  | 252, 878                      |
| MM_CONSOLE .....               | 252, 878                      |
| MM_ERROR .....                 | 252, 879-880                  |
| MM_FIRM .....                  | 252                           |
| mm_FIRM .....                  | 878                           |
| MM_HALT .....                  | 252, 879                      |
| MM_HARD .....                  | 252, 878                      |
| MM_INFO .....                  | 252, 879                      |
| MM_NOCON .....                 | 253, 879                      |
| MM_NOMSG .....                 | 252, 879                      |
| MM_NOSEV .....                 | 252, 879                      |
| MM_NOTOK .....                 | 252, 879                      |
| MM_NRECOV .....                | 252, 878                      |
| MM_NULLACT .....               | 252                           |
| MM_NULLBL .....                | 252                           |
| MM_NULLMC .....                | 252, 878                      |
| MM_NULLSEV .....               | 252                           |
| MM_NULLTAG .....               | 252                           |
| MM_NULLTXT .....               | 252                           |
| MM_OK .....                    | 252, 879                      |
| MM_OPSYS .....                 | 252, 878                      |
| MM_PRINT .....                 | 252, 878, 880                 |
| MM_RECOVER .....               | 252, 878                      |
| MM_SOFT .....                  | 252, 878                      |
| MM_UTIL .....                  | 252, 878                      |
| MM_WARNING .....               | 252, 879                      |
| mode .....                     | 69                            |
| modem disconnect .....         | 3512                          |
| mode_t .....                   | 401                           |
| modf() .....                   | <b>1330</b>                   |
| modff() .....                  | 1330                          |
| modfl() .....                  | 1330                          |
| modify batch job request ..... | 2414                          |
| MON .....                      | <b>8</b>                      |
| monotonic clock .....          | 69, 3580                      |
| MON_ .....                     | 267                           |
| mon_decimal_point .....        | 155                           |

|                                          |                            |
|------------------------------------------|----------------------------|
| mon_grouping .....                       | 155                        |
| mon_thousands_sep .....                  | 155                        |
| more .....                               | <b>2969</b> , 3724-3725    |
| discard and refresh .....                | 2975                       |
| display position .....                   | 2978                       |
| examine new file .....                   | 2977                       |
| examine next file .....                  | 2977                       |
| examine previous file .....              | 2977                       |
| go to beginning of file .....            | 2975                       |
| go to end-of-file .....                  | 2975                       |
| go to tag .....                          | 2977                       |
| help .....                               | 2974                       |
| invoke editor .....                      | 2977                       |
| mark position .....                      | 2975                       |
| quit .....                               | 2978                       |
| refresh the screen .....                 | 2975                       |
| repeat search .....                      | 2976                       |
| repeat search in reverse .....           | 2976                       |
| return to mark .....                     | 2976                       |
| return to previous position .....        | 2976                       |
| scroll backward one half screenful ..... | 2975                       |
| scroll backward one line .....           | 2974                       |
| scroll backward one screenful .....      | 2974                       |
| scroll forward one half screenful .....  | 2974                       |
| scroll forward one line .....            | 2974                       |
| scroll forward one screenful .....       | 2974                       |
| search backward for pattern .....        | 2976                       |
| search forward for pattern .....         | 2976                       |
| skip forward one line .....              | 2975                       |
| MORECTL .....                            | 368, 1042                  |
| MOREDATA .....                           | 368, 1042                  |
| motion command .....                     | 3195                       |
| mount point .....                        | 69, 3460                   |
| mount() .....                            | 3460                       |
| mounted file system .....                | 3460                       |
| move batch job request .....             | 2415                       |
| mprotect() .....                         | <b>1332</b> , 3561         |
| MQ .....                                 | 475                        |
| mq .....                                 | 475                        |
| mq_close() .....                         | <b>1334</b>                |
| mq_getattr() .....                       | <b>1335</b>                |
| mq_notify() .....                        | <b>1337</b>                |
| mq_open() .....                          | <b>1340</b> , 3549         |
| MQ_OPEN_MAX .....                        | 271, 2077, 3735            |
| MQ_PRIO_MAX .....                        | 271, 1346-1347, 2077, 3735 |
| mq_receive() .....                       | <b>1343</b> , 3550         |
| mq_send() .....                          | <b>1346</b> , 3550         |
| mq_setattr() .....                       | <b>1348</b>                |
| mq_timedreceive() .....                  | 1343, 1350, 3581           |
| mq_timedsend() .....                     | 1346, 1351, 3581           |
| mq_unlink() .....                        | <b>1352</b>                |
| mrnd48() .....                           | 743, 1354                  |

|                                        |                                   |
|----------------------------------------|-----------------------------------|
| MSG.....                               | 8                                 |
| msg.....                               | 475                               |
| msg*().....                            | 3546                              |
| msgctl().....                          | 1355, 3546                        |
| msgget().....                          | 1357, 3546                        |
| msgrcv().....                          | 1359, 3546                        |
| msgsnd().....                          | 1362, 3546                        |
| MSGVERB.....                           | 178, 879-880                      |
| MSG_.....                              | 475-476                           |
| msg_.....                              | 476                               |
| MSG_ANY.....                           | 368, 1041                         |
| MSG_BAND.....                          | 368, 1041, 1734                   |
| MSG_CTRUNC.....                        | 387                               |
| MSG_DONTROUTE.....                     | 387                               |
| MSG_EOR.....                           | 387, 1858, 1861, 1865, 1983, 1985 |
| MSG_HIPRI.....                         | 368, 1041, 1734                   |
| MSG_NOERROR.....                       | 375, 1359-1360                    |
| MSG_NOSIGNAL.....                      | 387, 1858, 1861, 1865             |
| MSG_OOB.....                           | 387, 1858, 1861, 1865             |
| MSG_PEEK.....                          | 387                               |
| msg_perm.....                          | 501                               |
| MSG_TRUNC.....                         | 387                               |
| MSG_WAITALL.....                       | 387                               |
| msqid.....                             | 501                               |
| MST7MDT.....                           | 2162                              |
| msync().....                           | 1365, 3561                        |
| MS.....                                | 475                               |
| MS_ASYNC.....                          | 372, 1324, 1365                   |
| MS_INVALIDATE.....                     | 372, 1365-1366                    |
| MS_SYNC.....                           | 372, 1324, 1365                   |
| multi-byte character.....              | 3511, 3513                        |
| multi-character collating element..... | 69                                |
| multicast.....                         | 532                               |
| multiple tasks.....                    | 3717, 3724                        |
| munlock().....                         | 1318, 1368                        |
| munlockall().....                      | 1320, 1369                        |
| munmap().....                          | 1370, 3560-3562, 3564, 3567       |
| mutex.....                             | 69, 3598                          |
| attributes.....                        | 1663                              |
| extended attributes.....               | 3601                              |
| initialization.....                    | 3613                              |
| initialization attributes.....         | 1662                              |
| performance.....                       | 1663                              |
| MUXID_ALL.....                         | 368, 1140-1141                    |
| MUXID_R.....                           | 477                               |
| mv.....                                | 2981, 3674, 3725                  |
| MX.....                                | 9                                 |
| MXX.....                               | 9                                 |
| M_.....                                | 289, 477                          |
| M_E.....                               | 289                               |
| M_LN.....                              | 289                               |
| M_LOG10E.....                          | 289                               |

|                             |                                                     |
|-----------------------------|-----------------------------------------------------|
| M_LOG2E.....                | 289                                                 |
| M_PI.....                   | 289                                                 |
| M_SQRT1_2.....              | 290                                                 |
| M_SQRT2.....                | 290                                                 |
| name.....                   | 70                                                  |
| name information.....       | 1045                                                |
| name space.....             | 473, 3527                                           |
| name space pollution.....   | 3527-3528                                           |
| named STREAM.....           | 70                                                  |
| NAME_MAX.....               | 111, 231, 274, 484, 893, 2437, 3060, 3668, 3735     |
| NaN.....                    | 70, 247                                             |
| NAN.....                    | 290                                                 |
| NaN.....                    | 598, 600, 604, 610                                  |
| NAN.....                    | 903                                                 |
| NaN.....                    | 903                                                 |
| NAN.....                    | 985                                                 |
| NaN.....                    | 985                                                 |
| NaN arguments.....          | 3481                                                |
| mathematical functions..... | 118                                                 |
| nan().....                  | 1372                                                |
| nanf().....                 | 1372                                                |
| nanl().....                 | 1372                                                |
| nanosleep().....            | 1373, 3578, 3580-3581, 3722                         |
| native implementation.....  | 3460                                                |
| native language.....        | 70                                                  |
| NCCS.....                   | 414                                                 |
| NDEBUG.....                 | 223, 480, 603                                       |
| nearbyint().....            | 1375                                                |
| nearbyintf().....           | 1375                                                |
| nearbyintl().....           | 1375                                                |
| negative response.....      | 70                                                  |
| negative_sign.....          | 155                                                 |
| netent.....                 | 302                                                 |
| network.....                | 70                                                  |
| network address.....        | 70                                                  |
| network byte order.....     | 71, 110, 3476                                       |
| network interfaces.....     | 523                                                 |
| newgrp.....                 | 2318, 2339, 2987, 3726                              |
| newline character.....      | 71                                                  |
| newlocale().....            | 1376                                                |
| news.....                   |                                                     |
| rationale for omission..... | 3710                                                |
| NEW_TIME.....               | 455, 765-766                                        |
| nextafter().....            | 1379                                                |
| nextafterf().....           | 1379                                                |
| nextafterl().....           | 1379                                                |
| nexttoward().....           | 1379                                                |
| nexttowardf().....          | 1379                                                |
| nexttowardl().....          | 1379                                                |
| nftw().....                 | 1381                                                |
| NGROUPS_MAX.....            | 275, 1031, 2077, 2990, 3441, 3463, 3668, 3729, 3735 |
| nice.....                   | 2991, 3724                                          |

|                                           |                                     |
|-------------------------------------------|-------------------------------------|
| nice value.....                           | 71, 3460                            |
| nice() .....                              | <b>1385</b> , 3572                  |
| Ninth Edition UNIX.....                   | 2505, 2641, 3079                    |
| NI_DGRAM .....                            | 303                                 |
| NI_NAMEREQD .....                         | 303                                 |
| NI_NOFQDN.....                            | 303                                 |
| NI_NUMERICHOST .....                      | 303                                 |
| NI_NUMERICSCOPE .....                     | 303                                 |
| NI_NUMERICSERV .....                      | 303                                 |
| nl.....                                   | <b>2995</b>                         |
| NLDLY .....                               | 415                                 |
| nlink_t.....                              | 401                                 |
| NLn .....                                 | 415                                 |
| NLSPATH .....                             | <b>175</b> , 644                    |
| NL_.....                                  | 475                                 |
| NL_ARGMAX .....                           | 282, 900, 938, 982, 992             |
| NL_CAT_LOCALE.....                        | 311, 644                            |
| nl_langinfo() .....                       | <b>1387</b> , 3723                  |
| nl_langinfo_l().....                      | 1387                                |
| NL_LANGMAX .....                          | 282                                 |
| NL_MSGMAX .....                           | 282                                 |
| NL_SETD.....                              | 311                                 |
| NL_SETMAX .....                           | 282                                 |
| NL_TEXTMAX .....                          | 282                                 |
| nm.....                                   | <b>2999</b> , 3726                  |
| noclobber option .....                    | 3030, 3689                          |
| NOEXPR.....                               | 267                                 |
| NOFLSH .....                              | 417                                 |
| nohup .....                               | 786, 3004, 3724                     |
| non-blocking .....                        | 71                                  |
| non-canonical mode input processing ..... | 202, 3512                           |
| non-local jumps .....                     | 1962                                |
| non-printable .....                       | 2655, 3186, 3246, 3470              |
| non-spacing characters.....               | 71                                  |
| non-volatile storage .....                | 963                                 |
| normative references .....                | 3442                                |
| NOSTR.....                                | 267                                 |
| NQS.....                                  | 3703                                |
| rand48().....                             | 743, 1389                           |
| ntohl() .....                             | 1106, 1390                          |
| ntohs() .....                             | 1106, 1390                          |
| NUL.....                                  | 72                                  |
| NULL .....                                | 345, 424, 692, 725, 735, 1327, 1761 |
| null byte.....                            | 72                                  |
| null pointer.....                         | 72                                  |
| null string .....                         | 72                                  |
| null wide-character code.....             | 72                                  |
| number-sign .....                         | 72                                  |
| numerical limits.....                     | 280                                 |
| NUM_EMPL .....                            | 1104                                |
| NZERO .....                               | 282, 1060, 1385                     |
| n_ .....                                  | 475                                 |

|                                   |                                                                        |
|-----------------------------------|------------------------------------------------------------------------|
| n_cs_precedes .....               | 155                                                                    |
| n_sep_by_space .....              | 155                                                                    |
| n_sign_posn .....                 | 156                                                                    |
| OB .....                          | 9                                                                      |
| object file .....                 | 72, 2999                                                               |
| obsolescent .....                 | 3443                                                                   |
| rationale .....                   | 3443                                                                   |
| OCRNL .....                       | 415                                                                    |
| octet .....                       | 72                                                                     |
| od .....                          | 3008, 3724, 3726                                                       |
| OF .....                          | 9                                                                      |
| OFDEL .....                       | 415                                                                    |
| offset maximum .....              | 73                                                                     |
| off_t .....                       | 401                                                                    |
| OFILL .....                       | 415                                                                    |
| OH .....                          | 9                                                                      |
| OLD_TIME .....                    | 455, 765-766                                                           |
| ONLCR .....                       | 415                                                                    |
| ONLRET .....                      | 415                                                                    |
| ONOCR .....                       | 415                                                                    |
| opaque address .....              | 73                                                                     |
| open a file .....                 | 1397                                                                   |
| open a named semaphore .....      | 1836                                                                   |
| open a shared memory object ..... | 1915                                                                   |
| open file .....                   | 73                                                                     |
| open file description .....       | 73, 3460                                                               |
| open file descriptors .....       | 3691                                                                   |
| for reading and writing .....     | 2337                                                                   |
| open mode .....                   | 2662                                                                   |
| open() .....                      | 1391, 3455, 3510, 3561-3564, 3674, 3720                                |
| openat() .....                    | 1391, 1403                                                             |
| opendir() .....                   | 829, 1404, 3721                                                        |
| openlog() .....                   | 687, 1405, 3726                                                        |
| OPEN_MAX .....                    | 271, 328, 760, 816, 829, 972, 1340, 1443, 1446, 2077, 2140, 3441, 3668 |
| .....                             | 3735-3736                                                              |
| open_memstream() .....            | 1401                                                                   |
| open_wmemstream() .....           | 1401                                                                   |
| operand .....                     | 73                                                                     |
| operator .....                    | 73                                                                     |
| OPOST .....                       | 415                                                                    |
| optarg .....                      | 1049, 1406                                                             |
| opterr .....                      | 1049, 1406                                                             |
| optind .....                      | 1049, 1406                                                             |
| option .....                      | 74                                                                     |
| ADV .....                         | 7                                                                      |
| BE .....                          | 7                                                                      |
| CD .....                          | 7                                                                      |
| CPT .....                         | 7                                                                      |
| FD .....                          | 7                                                                      |
| FR .....                          | 7                                                                      |
| FSC .....                         | 8                                                                      |
| IP6 .....                         | 8                                                                      |

|                              |                     |
|------------------------------|---------------------|
| MC1 .....                    | 8                   |
| ML .....                     | 8                   |
| MLR .....                    | 8                   |
| MON .....                    | 8                   |
| MSG .....                    | 8                   |
| MX .....                     | 9                   |
| MXX .....                    | 9                   |
| PIO .....                    | 9                   |
| PS .....                     | 9                   |
| RPI .....                    | 9                   |
| RPP .....                    | 10                  |
| RS .....                     | 10                  |
| SD .....                     | 10                  |
| SHM .....                    | 10                  |
| SIO .....                    | 10                  |
| SPN .....                    | 10                  |
| SS .....                     | 10                  |
| TCT .....                    | 10                  |
| TEF .....                    | 11                  |
| TPI .....                    | 11                  |
| TPP .....                    | 11                  |
| TPS .....                    | 11                  |
| TRC .....                    | 11                  |
| TRI .....                    | 11                  |
| TRL .....                    | 11                  |
| TSA .....                    | 12                  |
| TSH .....                    | 12                  |
| TSP .....                    | 12                  |
| TSS .....                    | 12                  |
| TYM .....                    | 12                  |
| UP .....                     | 12                  |
| UU .....                     | 12                  |
| XSR .....                    | 13                  |
| option definitions .....     | 3703                |
| option-argument .....        | 74                  |
| optional behavior .....      | 3737                |
| options .....                | 3622                |
| shell and utilities .....    | 27                  |
| system interfaces .....      | 27                  |
| optopt .....                 | 1049, 1052, 1406    |
| optstring .....              | 1052                |
| OR lists .....               | 2343, 3696          |
| ordinary identifiers .....   | 2498                |
| ORD_CHAR .....               | 192                 |
| orientation .....            | 74                  |
| orphaned process group ..... | 74, 552, 3461, 3542 |
| output device .....          | 198, 3508           |
| output file descriptor       |                     |
| duplication .....            | 3691                |
| output mode .....            | 3513                |
| output processing .....      | 3512                |
| Output_Path .....            | 2408                |



|                                        |                                                        |
|----------------------------------------|--------------------------------------------------------|
| overrun conditions.....                | 3646                                                   |
| overrun in dumping trace streams ..... | 3646                                                   |
| overrun in trace streams.....          | 3646                                                   |
| O_ .....                               | 477                                                    |
| O_ constants                           |                                                        |
| defined in <fcntl.h> .....             | 238-239                                                |
| used in open() .....                   | 1391                                                   |
| used in posix_openpt() .....           | 1433                                                   |
| O_ACCMODE .....                        | 239, 813                                               |
| O_APPEND.....                          | 239, 503, 587, 723, 827, 1391, 2285                    |
| O_CLOEXEC.....                         | 238, 1391, 3564                                        |
| O_CREAT.....                           | 238, 707, 1340-1341, 1391, 1827, 1834, 1913-1914, 1916 |
| O_DIRECTORY .....                      | 238, 1392                                              |
| O_DSYNC .....                          | 239, 578, 1392, 1753, 2286                             |
| O_EXCL.....                            | 239, 1341, 1392, 1834, 1913-1914                       |
| O_EXEC .....                           | 239, 1391                                              |
| O_NDELAY.....                          | 2291                                                   |
| O_NOCTTY.....                          | 239, 1392, 1433                                        |
| O_NOFOLLOW.....                        | 239, 1392                                              |
| O_NONBLOCK.....                        | 239, 483, 813, 1341, 1392                              |
| O_RDONLY .....                         | 239, 730, 1340, 1391, 1913, 1916                       |
| O_RDWR.....                            | 239, 803, 1254, 1340, 1391, 1433, 1913, 1916           |
| O_RSYNC.....                           | 239, 1392, 1753                                        |
| O_SEARCH.....                          | 239, 1391                                              |
| O_SYNC.....                            | 239, 578, 1393, 1753, 2286                             |
| O_TRUNC.....                           | 239, 707, 1393, 1914, 1916                             |
| O_TTY_INIT .....                       | 199, 239, 1393                                         |
| O_WRONLY.....                          | 239, 707, 803, 1254, 1340, 1391                        |
| pack                                   |                                                        |
| rationale for omission.....            | 3710                                                   |
| page .....                             | 74, 3461, 3560, 3564                                   |
| page size .....                        | 74                                                     |
| PAGESIZE .....                         | 271, 504, 1318, 1551, 2079, 3560, 3605, 3735           |
| PAGE_SIZE .....                        | 272, 2079                                              |
| paginators                             |                                                        |
| more .....                             | 2969                                                   |
| parallel I/O .....                     | 3605                                                   |
| parameter .....                        | 75, 3513, 3679                                         |
| expansion .....                        | 2328, 3684                                             |
| positional.....                        | 3679                                                   |
| special .....                          | 3679                                                   |
| parameters and variables.....          | 2323                                                   |
| PARENB.....                            | 417                                                    |
| parent directory .....                 | 75, 3461                                               |
| parent process .....                   | 75                                                     |
| parent process ID .....                | 75                                                     |
| PARMRK .....                           | 415                                                    |
| PARODD .....                           | 417                                                    |
| passwd                                 |                                                        |
| rationale for omission.....            | 3710                                                   |
| passwd file.....                       | 3461                                                   |
| paste .....                            | 3016, 3724                                             |

|                                               |                                                  |
|-----------------------------------------------|--------------------------------------------------|
| patch .....                                   | 3020, 3725-3726                                  |
| application.....                              | 3023                                             |
| file format .....                             | 3022                                             |
| filename determination .....                  | 3023                                             |
| PATH.....                                     | 178, 692, 787, 3499                              |
| path prefix .....                             | 76                                               |
| pathchk .....                                 | 3027, 3725                                       |
| pathconf().....                               | 893, 1407, 3455, 3667, 3719, 3721                |
| pathname.....                                 | 75, 3461                                         |
| component.....                                | 76                                               |
| expansion .....                               | 2333, 3689                                       |
| incomplete .....                              | 3455                                             |
| resolution .....                              | 111, 2305, 3478                                  |
| variable values.....                          | 273                                              |
| pathname manipulation                         |                                                  |
| basename.....                                 | 2487                                             |
| dirname.....                                  | 2632                                             |
| pathchk .....                                 | 3027                                             |
| PATH_MAX .....                                | 274, 282, 484, 893, 2308, 3066, 3164, 3668, 3735 |
| pattern.....                                  | 76                                               |
| filename expansion .....                      | 3702                                             |
| for filename expansion .....                  | 2355                                             |
| scanning and processing language.....         | 2452                                             |
| pattern matching .....                        | 2762, 3041, 3315, 3331                           |
| definition .....                              | 2354                                             |
| in case statements.....                       | 2345                                             |
| in shell variables.....                       | 2329                                             |
| multiple character .....                      | 3701                                             |
| multiple characters.....                      | 2355                                             |
| notation.....                                 | 2354, 2768, 3059, 3700                           |
| single character.....                         | 2354, 3701                                       |
| pause() .....                                 | 1408, 3541, 3544, 3720                           |
| pax.....                                      | 3032, 3725-3726                                  |
| archive character set encoding/decoding ..... | 3065                                             |
| cpio file data.....                           | 3055                                             |
| cpio filename.....                            | 3055                                             |
| cpio header .....                             | 3053                                             |
| cpio interchange format .....                 | 3053                                             |
| cpio special entries .....                    | 3055                                             |
| extended header .....                         | 3045                                             |
| extended header file times.....               | 3049                                             |
| extended header keyword precedence .....      | 3048                                             |
| list mode format specifications .....         | 3040                                             |
| ustar format.....                             | 3049                                             |
| ustar interchange format.....                 | 3049                                             |
| pcat                                          |                                                  |
| rationale for omission .....                  | 3711                                             |
| pclose() .....                                | 1409, 3726                                       |
| pd_.....                                      | 475                                              |
| PENDIN.....                                   | 477                                              |
| pending error .....                           | 3621                                             |
| per-thread errno .....                        | 3534                                             |

|                                      |                                  |
|--------------------------------------|----------------------------------|
| performance enhancements.....        | 3716                             |
| period.....                          | 76                               |
| permissions.....                     | 77                               |
| perror().....                        | <b>1411</b>                      |
| persistence.....                     | 77                               |
| persistent connection (I_PLINK)..... | <b>1141</b>                      |
| PF_.....                             | 476                              |
| pg                                   |                                  |
| rationale for omission.....          | 3711                             |
| physical write.....                  | 963                              |
| ph_.....                             | 475                              |
| PID_MAX.....                         | 3648                             |
| pid_t.....                           | 401                              |
| PIO.....                             | <b>9</b>                         |
| pipe.....                            | 77, 891, 1397, 2289, 3456, 3462  |
| pipe().....                          | <b>1413</b> , 3541, 3720         |
| pipelines.....                       | 2341, 3694                       |
| PIPE_BUF.....                        | 274, 893, 2286, 2289, 3668, 3735 |
| PIPE_MAX.....                        | 2291                             |
| plain characters.....                | 2017                             |
| PM_STR.....                          | 267                              |
| pointer to a function.....           | 492                              |
| pointer types.....                   | 3649                             |
| pole error.....                      | 117                              |
| POLL.....                            | 475                              |
| poll().....                          | <b>1416</b>                      |
| POLLERR.....                         | 312, 1416                        |
| pollfd.....                          | <b>312</b>                       |
| POLLHUP.....                         | 312, 1416                        |
| POLLIN.....                          | 312, 1416                        |
| polling.....                         | 77                               |
| POLLNVAL.....                        | 312, 1417                        |
| POLLOUT.....                         | 312, 1416                        |
| POLLPRI.....                         | 312, 1416                        |
| POLLRDBAND.....                      | 312, 1416                        |
| POLLRDNORM.....                      | 312, 1416                        |
| POLLWRBAND.....                      | 312, 1416                        |
| POLLWRNORM.....                      | 312, 1416                        |
| POLL_.....                           | 475                              |
| POLL_ERR.....                        | 336                              |
| POLL_HUP.....                        | 336                              |
| POLL_IN.....                         | 336                              |
| POLL_MSG.....                        | 336                              |
| POLL_OUT.....                        | 336                              |
| POLL_PRI.....                        | 336                              |
| popen().....                         | <b>1420</b> , 3723, 3726-3727    |
| portability.....                     | 3444                             |
| portability codes.....               | 3445                             |
| portable character set.....          | 77, 125, 2940, 3482              |
| portable filename.....               | 78                               |
| portable filename character set..... | 78, 3462                         |
| positional parameter.....            | 78, 2323, 3679                   |

|                                  |                          |
|----------------------------------|--------------------------|
| positive_sign .....              | 155                      |
| POSIX conformance .....          | 15                       |
| POSIX locale .....               | 136, 3486                |
| POSIX shell and utilities .....  | 18                       |
| POSIX system interfaces          |                          |
| conformance .....                | 17                       |
| POSIX.1 symbols .....            | 472, 3526                |
| POSIX.13 .....                   | 3564                     |
| POSIX2_BC_BASE_MAX .....         | 2307-2308, 3729          |
| POSIX2_BC_DIM_MAX .....          | 2307-2308, 3729          |
| POSIX2_BC_SCALE_MAX .....        | 2307-2308, 3729          |
| POSIX2_BC_STRING_MAX .....       | 2307-2308, 3729          |
| POSIX2_CHAR_TERM .....           | 19, 27, 3728             |
| POSIX2_COLL_WEIGHTS_MAX .....    | 2307-2308, 3729          |
| POSIX2_C_BIND .....              | 3669, 3727               |
| POSIX2_C_DEV .....               | 19, 27, 3669, 3727-3728  |
| POSIX2_EXPR_NEST_MAX .....       | 2307-2308, 3729          |
| POSIX2_FORT_DEV .....            | 19, 27, 3669, 3728       |
| POSIX2_FORT_RUN .....            | 19, 27, 3669, 3728       |
| POSIX2_LINE_MAX .....            | 2307, 2309, 3729         |
| POSIX2_LOCALEDEF .....           | 19, 27, 3669, 3725, 3728 |
| POSIX2_PBS .....                 | 19, 28, 3728             |
| POSIX2_PBS_ACCOUNTING .....      | 19, 28, 3728             |
| POSIX2_PBS_CHECKPOINT .....      | 28, 3728                 |
| POSIX2_PBS_LOCATE .....          | 19, 28, 3728             |
| POSIX2_PBS_MESSAGE .....         | 19, 28, 3728             |
| POSIX2_PBS_TRACK .....           | 19, 28, 3728             |
| POSIX2_RE_DUP_MAX .....          | 2307, 2309, 3729         |
| POSIX2_SW_DEV .....              | 19, 28, 3669, 3727       |
| POSIX2_SYMLINKS .....            | 893, 2309, 3669          |
| POSIX2_UPE .....                 | 19, 28, 3669, 3727-3728  |
| POSIX2_VERSION .....             | 3729                     |
| POSIX_ .....                     | 474                      |
| posix_ .....                     | 474                      |
| POSIX_ALLOC_SIZE_MIN .....       | 274, 893, 3547           |
| POSIX_ASYNCHRONOUS_IO .....      | 3741                     |
| POSIX_BARRIERS .....             | 3741                     |
| POSIX_CLOCK_SELECTION .....      | 3742                     |
| POSIX_C_LANG_JUMP .....          | 3741                     |
| POSIX_C_LANG_MATH .....          | 3741                     |
| POSIX_C_LANG_SUPPORT .....       | 3742                     |
| POSIX_C_LANG_SUPPORT_R .....     | 3742                     |
| POSIX_C_LANG_WIDE_CHAR .....     | 3742                     |
| POSIX_C_LANG_WIDE_CHAR_EXT ..... | 3742                     |
| POSIX_C_LIB_EXT .....            | 3742                     |
| POSIX_DEVICE_IO .....            | 3742                     |
| POSIX_DEVICE_IO_EXT .....        | 3742                     |
| POSIX_DEVICE_SPECIFIC .....      | 3742                     |
| POSIX_DEVICE_SPECIFIC_R .....    | 3743                     |
| POSIX_DYNAMIC_LINKING .....      | 3743                     |
| posix_fadvise() .....            | 1423, 3547               |
| POSIX_FADV_DONTNEED .....        | 240, 1423, 3547          |

|                                     |                          |
|-------------------------------------|--------------------------|
| POSIX_FADV_NOREUSE.....             | 240, 1423, 3547          |
| POSIX_FADV_NORMAL .....             | 240, 1423                |
| POSIX_FADV_RANDOM.....              | 240, 1423, 3547          |
| POSIX_FADV_SEQUENTIAL .....         | 240, 1423, 3547          |
| POSIX_FADV_WILLNEED .....           | 240, 1423, 3547          |
| posix_fallocate() .....             | <b>1425</b>              |
| POSIX_FD_MGMT.....                  | 3743                     |
| POSIX_FIFO .....                    | 3743                     |
| POSIX_FIFO_FD.....                  | 3743                     |
| POSIX_FILE_ATTRIBUTES .....         | 3743                     |
| POSIX_FILE_ATTRIBUTES_FD.....       | 3743                     |
| POSIX_FILE_LOCKING .....            | 3743                     |
| POSIX_FILE_SYSTEM.....              | 3743                     |
| POSIX_FILE_SYSTEM_EXT .....         | 3743                     |
| POSIX_FILE_SYSTEM_FD.....           | 3743                     |
| POSIX_FILE_SYSTEM_GLOB.....         | 3743                     |
| POSIX_FILE_SYSTEM_R .....           | 3743                     |
| POSIX_I18N .....                    | 3743                     |
| POSIX_JOB_CONTROL .....             | 3743                     |
| posix_madvise() .....               | <b>1427</b> , 3547       |
| POSIX_MADV_DONTNEED.....            | 372, 1427, 3547          |
| POSIX_MADV_NORMAL.....              | 372, 1427                |
| POSIX_MADV_RANDOM.....              | 372, 1427, 3547          |
| POSIX_MADV_SEQUENTIAL .....         | 372, 1427, 3547          |
| POSIX_MADV_WILLNEED .....           | 373, 1427, 3547          |
| POSIX_MAPPED_FILES.....             | 3743                     |
| posix_memalign() .....              | <b>1431</b>              |
| POSIX_MEMORY_PROTECTION .....       | 3743                     |
| posix_mem_offset() .....            | <b>1429</b> , 3564-3565  |
| POSIX_MULTI_CONCURRENT_LOCALES..... | 3743                     |
| POSIX_MULTI_PROCESS .....           | 3744                     |
| POSIX_MULTI_PROCESS_FD .....        | 3744                     |
| POSIX_NETWORKING .....              | 3744                     |
| posix_openpt() .....                | <b>1433</b>              |
| POSIX_PIPE .....                    | 3744                     |
| POSIX_REALTIME_SIGNALS.....         | 3744                     |
| POSIX_REC_INCR_XFER_SIZE .....      | 274, 893, 3548           |
| POSIX_REC_MAX_XFER_SIZE.....        | 274, 893, 3548           |
| POSIX_REC_MIN_XFER_SIZE.....        | 274, 893, 3548           |
| POSIX_REC_XFER_ALIGN.....           | 274, 893, 3547           |
| POSIX_REGEX .....                   | 3744                     |
| POSIX_ROBUST_MUTEXES.....           | 3744                     |
| POSIX_RW_LOCKS .....                | 3744                     |
| POSIX_SEMAPHORES .....              | 3744                     |
| POSIX_SHELL_FUNC.....               | 3744                     |
| POSIX_SIGNALS .....                 | 3744                     |
| POSIX_SIGNALS_EXT.....              | 3744                     |
| POSIX_SIGNAL_JUMP .....             | 3744                     |
| POSIX_SINGLE_PROCESS .....          | 3745                     |
| posix_spawn() .....                 | <b>1435</b> , 3652, 3720 |
| posix_spawnattr_destroy().....      | <b>1451</b>              |
| posix_spawnattr_getflags().....     | <b>1453</b>              |

|                                              |                        |
|----------------------------------------------|------------------------|
| posix_spawnattr_getpgroup()                  | 1455                   |
| posix_spawnattr_getschedparam()              | 1457                   |
| posix_spawnattr_getschedpolicy()             | 1459                   |
| posix_spawnattr_getsigdefault()              | 1461                   |
| posix_spawnattr_getsigmask()                 | 1463                   |
| posix_spawnattr_init()                       | 1451, 1465             |
| posix_spawnattr_setflags()                   | 1453, 1466             |
| posix_spawnattr_setpgroup()                  | 1455, 1467             |
| posix_spawnattr_setschedparam()              | 1457, 1468             |
| posix_spawnattr_setschedpolicy()             | 1459, 1469             |
| posix_spawnattr_setsigdefault()              | 1461, 1470             |
| posix_spawnattr_setsigmask()                 | 1463, 1471             |
| posix_spawnnp()                              | 1435, 1472, 3652, 3720 |
| posix_spawn_file_actions_addclose()          | 1443                   |
| posix_spawn_file_actions_adddup2()           | 1446                   |
| posix_spawn_file_actions_addopen()           | 1443, 1448             |
| posix_spawn_file_actions_destroy()           | 1449                   |
| posix_spawn_file_actions_init()              | 1449                   |
| POSIX_SPAWN_RESETIDS                         | 1436, 1453             |
| POSIX_SPAWN_SETPGROUP                        | 1436, 1453, 1455       |
| POSIX_SPAWN_SETSCHEDPARAM                    | 1453, 1457             |
| POSIX_SPAWN_SETSCHEDULER                     | 1436, 1453, 1457, 1459 |
| POSIX_SPAWN_SETSIGDEF                        | 1437, 1453, 1461       |
| POSIX_SPAWN_SETSIGMASK                       | 1453, 1463             |
| POSIX_SPIN_LOCKS                             | 3745                   |
| POSIX_SYMBOLIC_LINKS                         | 3745                   |
| POSIX_SYMBOLIC_LINKS_FD                      | 3745                   |
| POSIX_SYSTEM_DATABASE                        | 3745                   |
| POSIX_SYSTEM_DATABASE_R                      | 3745                   |
| POSIX_THREADS_BASE                           | 3745                   |
| POSIX_THREADS_EXT                            | 3745                   |
| POSIX_TIMERS                                 | 3745                   |
| POSIX_TRACE_ADD_EVENTSET                     | 1512                   |
| POSIX_TRACE_ALL_EVENTS                       | 1505                   |
| POSIX_TRACE_APPEND                           | 1478, 1498             |
| posix_trace_attr_destroy()                   | 1473                   |
| posix_trace_attr_getclockres()               | 1475                   |
| posix_trace_attr_getcreatetime()             | 1475                   |
| posix_trace_attr_getgenversion()             | 1475                   |
| posix_trace_attr_getinherited()              | 1477                   |
| posix_trace_attr_getlogfullpolicy()          | 1477                   |
| posix_trace_attr_getlogsize()                | 1480                   |
| posix_trace_attr_getmaxdatasize()            | 1480                   |
| posix_trace_attr_getmaxsystemeventsizesize() | 1480                   |
| posix_trace_attr_getmaxusereventsizesize()   | 1480                   |
| posix_trace_attr_getname()                   | 1475, 1483             |
| posix_trace_attr_getstreamfullpolicy()       | 1477, 1484             |
| posix_trace_attr_getstreamsize()             | 1480, 1485             |
| posix_trace_attr_init()                      | 1473, 1486             |
| posix_trace_attr_setinherited()              | 1477, 1487             |
| posix_trace_attr_setlogfullpolicy()          | 1477, 1487             |
| posix_trace_attr_setlogsize()                | 1480, 1488             |

|                                        |                            |
|----------------------------------------|----------------------------|
| posix_trace_attr_setmaxdatasize()      | 1480, 1488                 |
| posix_trace_attr_setname()             | 1475, 1489                 |
| posix_trace_attr_setstreamfullpolicy() | 1477, 1490                 |
| posix_trace_attr_setstreamsize()       | 1480, 1491                 |
| posix_trace_clear()                    | <b>1492</b>                |
| posix_trace_close()                    | <b>1494</b>                |
| POSIX_TRACE_CLOSE_FOR_CHILD            | 1477                       |
| posix_trace_create()                   | <b>1496</b>                |
| posix_trace_create_withlog()           | 1496                       |
| POSIX_TRACE_ERROR trace event          | 541                        |
| posix_trace_event()                    | <b>1500</b>                |
| posix_trace_eventid_equal()            | <b>1502</b>                |
| posix_trace_eventid_get_name()         | 1502                       |
| posix_trace_eventid_open()             | 1500, 1504, 3641           |
| posix_trace_eventset_add()             | <b>1505</b>                |
| posix_trace_eventset_del()             | 1505                       |
| posix_trace_eventset_empty()           | 1505                       |
| posix_trace_eventset_fill()            | 1505                       |
| posix_trace_eventset_ismember()        | 1505                       |
| posix_trace_eventtypelist_getnext_id() | <b>1507</b>                |
| posix_trace_eventtypelist_rewind()     | 1507                       |
| posix_trace_event_info                 | 538                        |
| POSIX_TRACE_FILTER trace event         | 541, 1512                  |
| POSIX_TRACE_FLUSH                      | 1478                       |
| posix_trace_flush()                    | 1496, 1509                 |
| POSIX_TRACE_FLUSHING                   | 537                        |
| POSIX_TRACE_FULL                       | 536-538                    |
| posix_trace_getnext_event()            | <b>1515</b>                |
| posix_trace_get_attr()                 | <b>1510</b>                |
| posix_trace_get_filter()               | <b>1512</b>                |
| posix_trace_get_status()               | 1510, 1514                 |
| POSIX_TRACE_INHERITED                  | 1477                       |
| POSIX_TRACE_LOOP                       | 537, 1477-1478, 1498, 3646 |
| POSIX_TRACE_NOT_FLUSHING               | 538                        |
| POSIX_TRACE_NOT_FULL                   | 536-538                    |
| POSIX_TRACE_NOT_FULL                   | 1492                       |
| POSIX_TRACE_NOT_TRUNCATED              | 539, 1516                  |
| POSIX_TRACE_NO_OVERRUN                 | 537-538, 1510              |
| posix_trace_open()                     | 1494, 1518                 |
| POSIX_TRACE_OVERFLOW trace event       | 541                        |
| POSIX_TRACE_OVERRUN                    | 537-538                    |
| POSIX_TRACE_RESUME trace event         | 541                        |
| posix_trace_rewind()                   | 1494, 1518                 |
| POSIX_TRACE_RUNNING                    | 536-537, 1521              |
| POSIX_TRACE_SET_EVENTSET               | 1512                       |
| posix_trace_set_filter()               | 1512, 1519                 |
| posix_trace_shutdown()                 | 1496, 1520                 |
| POSIX_TRACE_START trace event          | 541, 1521                  |
| posix_trace_start()                    | <b>1521</b>                |
| posix_trace_status_info                | 536                        |
| POSIX_TRACE_STOP trace event           | 541, 1521                  |
| posix_trace_stop()                     | 1521                       |

|                                                  |                                  |
|--------------------------------------------------|----------------------------------|
| POSIX_TRACE_SUB_EVENTSET .....                   | 1512                             |
| POSIX_TRACE_SUSPENDED .....                      | 536-537, 1521                    |
| POSIX_TRACE_SYSTEM_EVENTS .....                  | 1505                             |
| posix_trace_timedgetnext_event() .....           | 1515, 1523                       |
| posix_trace_trid_eventid_open() .....            | 1502, 1524                       |
| POSIX_TRACE_TRUNCATED_READ .....                 | 539, 1516                        |
| POSIX_TRACE_TRUNCATED_RECORD .....               | 539, 1516                        |
| posix_trace_trygetnext_event() .....             | 1515, 1525                       |
| POSIX_TRACE_UNTIL_FULL .....                     | 537, 1477-1478, 1497             |
| POSIX_TRACE_USER_EVENT_MAX .....                 | 1500                             |
| POSIX_TRACE_WOPID_EVENTS .....                   | 1505                             |
| POSIX_TYPED_MEM_ALLOCATE .....                   | 373, 1322-1323, 1429, 1526, 1528 |
| POSIX_TYPED_MEM_ALLOCATE_CONTIG .....            | 373, 1322-1323, 1429, 1526, 1528 |
| posix_typed_mem_get_info() .....                 | 1526, 3564                       |
| posix_typed_mem_info .....                       | 373                              |
| POSIX_TYPED_MEM_MAP_ALLOCATABLE .....            | 373, 1370, 1528                  |
| posix_typed_mem_open() .....                     | 1528, 3564                       |
| POSIX_USER_GROUPS .....                          | 3745                             |
| POSIX_USER_GROUPS_R .....                        | 3745                             |
| POSIX_VERSION .....                              | 3735                             |
| POSIX_WIDE_CHAR_DEVICE_IO .....                  | 3745                             |
| post-mortem filtering of trace event types ..... | 3643                             |
| pow() .....                                      | 1531                             |
| powf() .....                                     | 1531                             |
| powl() .....                                     | 1531                             |
| pr .....                                         | 3070, 3724, 3726                 |
| pread() .....                                    | 1534, 1752, 3606                 |
| preallocation .....                              | 78                               |
| predefined stream                                |                                  |
| standard error .....                             | 498                              |
| standard input .....                             | 498                              |
| standard output .....                            | 498                              |
| preempted process (or thread) .....              | 78                               |
| preempted thread .....                           | 1603                             |
| previous job .....                               | 78                               |
| PRI .....                                        | 477                              |
| print-related commands                           |                                  |
| fold .....                                       | 2771                             |
| lp .....                                         | 2883                             |
| pr .....                                         | 3070                             |
| printable character .....                        | 79                               |
| printable file .....                             | 79                               |
| printf .....                                     | 3075, 3723-3724                  |
| printf() .....                                   | 900, 1535                        |
| printing .....                                   | 3718                             |
| priority .....                                   | 79, 500                          |
| Priority .....                                   | 2409                             |
| priority                                         |                                  |
| band .....                                       | 79                               |
| inversion .....                                  | 79                               |
| scheduling .....                                 | 79                               |
| priority-based scheduling .....                  | 79                               |



|                                           |                           |
|-------------------------------------------|---------------------------|
| PRIО_ .....                               | 475                       |
| PRIО_ constants                           |                           |
| defined in <sys/resource.h> .....         | 377                       |
| PRIО_INHERIT .....                        | 1658                      |
| PRIО_PGRP .....                           | 377, 1060                 |
| PRIО_PROCESS .....                        | 377, 1060                 |
| PRIО_USER .....                           | 377, 1060                 |
| privilege .....                           | 80, 2961, 2993, 3473      |
| process .....                             | 80                        |
| attributes .....                          | 2301                      |
| concurrent execution .....                | 890                       |
| ID .....                                  | 81, 2301                  |
| ID reuse .....                            | 112, 3479                 |
| ID, 1 .....                               | 552                       |
| ID, rationale .....                       | 3648                      |
| lifetime .....                            | 81, 3462                  |
| memory locking .....                      | 81                        |
| scheduling .....                          | 506, 3570, 3720           |
| setting real and effective user IDs ..... | 1898                      |
| single-threaded .....                     | 890                       |
| termination .....                         | 81, 3462                  |
| virtual time .....                        | 82                        |
| process creation .....                    | 890                       |
| process group .....                       | 80, 3509                  |
| concepts in job control .....             | 3456                      |
| ID .....                                  | 80, 2301, 3456, 3509-3510 |
| leader .....                              | 80                        |
| lifetime .....                            | 80, 3510                  |
| orphaned .....                            | 552, 3461, 3542           |
| termios .....                             | 200                       |
| process group ID .....                    | 1056, 1890, 1902          |
| process lifetime .....                    | 1213                      |
| process management .....                  | 3716, 3720                |
| process shared memory .....               | 1663                      |
| process status report .....               | 3086                      |
| process synchronization .....             | 1663                      |
| process termination .....                 | 551                       |
| process-to-process communication .....    | 81                        |
| prof                                      |                           |
| rationale for omission .....              | 3711                      |
| profiling .....                           | 3727                      |
| program .....                             | 82                        |
| programming manipulation .....            | 3637                      |
| prompting .....                           | 3681-3682                 |
| protocol .....                            | 82, 3620                  |
| protoent .....                            | 302                       |
| PROT_ .....                               | 475                       |
| PROT_EXEC .....                           | 372, 1323, 1332           |
| PROT_NONE .....                           | 372, 505, 1322-1323, 1332 |
| PROT_READ .....                           | 372, 1323, 1332           |
| PROT_READ constants                       |                           |
| in <sys/mman.h> .....                     | 372                       |

|                                                   |                                |
|---------------------------------------------------|--------------------------------|
| PROT_WRITE.....                                   | 372, 1323-1324, 1327, 1332     |
| prs.....                                          | <b>3081</b>                    |
| PS.....                                           | <b>9</b>                       |
| ps.....                                           | <b>3086</b> , 3724, 3726       |
| pselect().....                                    | <b>1536</b>                    |
| pseudo-random sequence generation functions ..... | 1749                           |
| pseudo-terminal .....                             | 82                             |
| psiginfo().....                                   | <b>1541</b>                    |
| psignal().....                                    | 1541                           |
| PST8PDT.....                                      | 2162                           |
| ps_.....                                          | 475                            |
| pthread.....                                      | 3644                           |
| PTHREAD_.....                                     | 475                            |
| pthread_.....                                     | 475                            |
| pthread_atfork().....                             | <b>1543</b>                    |
| pthread_attr_destroy().....                       | <b>1546</b>                    |
| pthread_attr_getdetachstate().....                | <b>1549</b>                    |
| pthread_attr_getguardsize().....                  | <b>1551</b> , 3605             |
| pthread_attr_getinheritsched().....               | <b>1554</b>                    |
| pthread_attr_getschedparam().....                 | <b>1556</b>                    |
| pthread_attr_getschedpolicy().....                | <b>1558</b>                    |
| pthread_attr_getscope().....                      | <b>1560</b>                    |
| pthread_attr_getstack().....                      | <b>1562</b>                    |
| pthread_attr_getstackaddr().....                  | 3651                           |
| pthread_attr_getstacksize().....                  | <b>1565</b>                    |
| pthread_attr_init().....                          | 1546, 1567                     |
| pthread_attr_setdetachstate().....                | 1549, 1568                     |
| pthread_attr_setguardsize().....                  | 1551, 1569, 3605               |
| pthread_attr_setinheritsched().....               | 1554, 1570                     |
| pthread_attr_setschedparam().....                 | 1556, 1571                     |
| pthread_attr_setschedpolicy().....                | 1558, 1572                     |
| pthread_attr_setscope().....                      | 1560, 1573                     |
| pthread_attr_setstack().....                      | 1562, 1574                     |
| pthread_attr_setstackaddr().....                  | 3651                           |
| pthread_attr_setstacksize().....                  | 1565, 1575                     |
| pthread_barrierattr_destroy().....                | <b>1580</b>                    |
| pthread_barrierattr_getpshared().....             | <b>1582</b>                    |
| pthread_barrierattr_init().....                   | 1580, 1584                     |
| pthread_barrierattr_setpshared().....             | 1582, 1585                     |
| pthread_barrier_destroy().....                    | <b>1576</b>                    |
| pthread_barrier_init().....                       | 1576                           |
| PTHREAD_BARRIER_SERIAL_THREAD .....               | 314, 1578, 3596                |
| pthread_barrier_wait().....                       | <b>1578</b> , 3597, 3617       |
| pthread_cancel().....                             | <b>1586</b>                    |
| PTHREAD_CANCELED.....                             | 314, 520, 1623                 |
| PTHREAD_CANCEL_ASYNCHRONOUS .....                 | 314, 517, 1709                 |
| PTHREAD_CANCEL_DEFERRED .....                     | 314, 517, 520, 780, 1600, 1709 |
| PTHREAD_CANCEL_DISABLE .....                      | 314, 516, 520, 1709            |
| PTHREAD_CANCEL_ENABLE .....                       | 314, 516, 520, 1709            |
| PTHREAD_CANCEL_ENABLED.....                       | 780                            |
| pthread_cleanup_pop().....                        | <b>1588</b>                    |
| pthread_cleanup_push().....                       | 1588                           |

|                                    |                              |
|------------------------------------|------------------------------|
| pthread_condattr_destroy()         | 1607                         |
| pthread_condattr_getclock()        | 1609                         |
| pthread_condattr_getpshared()      | 1611                         |
| pthread_condattr_init()            | 1607, 1613                   |
| pthread_condattr_setclock()        | 1609, 1614                   |
| pthread_condattr_setpshared()      | 1611, 1615                   |
| pthread_cond_broadcast()           | 1593                         |
| pthread_cond_destroy()             | 1596                         |
| pthread_cond_init()                | 1596, 3593                   |
| PTHREAD_COND_INITIALIZER           | 314, 1596                    |
| pthread_cond_signal()              | 1593, 1599                   |
| pthread_cond_timedwait()           | 1600, 3535, 3580, 3602, 3730 |
| pthread_cond_wait()                | 1600, 3535, 3552, 3602       |
| pthread_create()                   | 1616, 3593-3594              |
| PTHREAD_CREATE_DETACHED            | 314, 490, 1549, 3615         |
| PTHREAD_CREATE_JOINABLE            | 314, 490, 780, 1549, 1633    |
| PTHREAD_DESTRUCTOR_ITERATIONS      | 272, 1630, 1635, 2079, 3736  |
| pthread_detach()                   | 1619, 3615                   |
| pthread_equal()                    | 1621                         |
| pthread_exit()                     | 1622                         |
| PTHREAD_EXPLICIT_SCHED             | 314, 1554                    |
| pthread_getconcurrency()           | 1624, 3605                   |
| pthread_getcpuclockid()            | 1626, 3583-3584              |
| pthread_getschedparam()            | 1627                         |
| pthread_getspecific()              | 1630                         |
| PTHREAD_INHERIT_SCHED              | 314, 1554                    |
| pthread_join()                     | 1632, 3535, 3615             |
| PTHREAD_KEYS_MAX                   | 272, 1635, 2079, 3736        |
| pthread_key_create()               | 1635, 3596                   |
| pthread_key_delete()               | 1638                         |
| pthread_kill()                     | 1640                         |
| pthread_mutexattr_destroy()        | 1662                         |
| pthread_mutexattr_getprioceiling() | 1667                         |
| pthread_mutexattr_getprotocol()    | 1669                         |
| pthread_mutexattr_getpshared()     | 1672                         |
| pthread_mutexattr_getrobust()      | 1674                         |
| pthread_mutexattr_gettype()        | 1676, 3603                   |
| pthread_mutexattr_init()           | 1662, 1678                   |
| pthread_mutexattr_setprioceiling() | 1667, 1679                   |
| pthread_mutexattr_setprotocol()    | 1669, 1680                   |
| pthread_mutexattr_setpshared()     | 1672, 1681                   |
| pthread_mutexattr_setrobust()      | 1674, 1682                   |
| pthread_mutexattr_settype()        | 1676, 1683, 3603             |
| pthread_mutex_consistent()         | 1641                         |
| PTHREAD_MUTEX_DEFAULT              | 314, 1653, 1676, 3602        |
| pthread_mutex_destroy()            | 1643                         |
| PTHREAD_MUTEX_ERRORCHECK           | 314, 1649, 1653, 1676, 3602  |
| pthread_mutex_getprioceiling()     | 1649                         |
| pthread_mutex_init()               | 1643, 1652, 3593             |
| PTHREAD_MUTEX_INITIALIZER          | 314, 1643                    |
| pthread_mutex_lock()               | 1653, 3535, 3602, 3616-3617  |
| PTHREAD_MUTEX_NORMAL               | 314, 1653, 1676, 3602        |

|                                       |                                               |
|---------------------------------------|-----------------------------------------------|
| PTHREAD_MUTEX_RECURSIVE.....          | 111, 314, 1653, 1676, 3602                    |
| PTHREAD_MUTEX_ROBUST.....             | 1674                                          |
| pthread_mutex_setprioceiling() .....  | 1649, 1657                                    |
| PTHREAD_MUTEX_STALLED.....            | 1674                                          |
| pthread_mutex_timedlock() .....       | <b>1658</b> , 3581                            |
| pthread_mutex_trylock().....          | 1653, 1661, 3602                              |
| pthread_mutex_unlock() .....          | 1653, 1661, 3602                              |
| pthread_once() .....                  | <b>1684</b>                                   |
| PTHREAD_ONCE_INIT.....                | 314, 1684                                     |
| PTHREAD_PRIO_INHERIT.....             | 314, 1669                                     |
| PTHREAD_PRIO_NONE .....               | 314, 1649, 1669                               |
| PTHREAD_PRIO_PROTECT .....            | 314, 1654, 1669                               |
| PTHREAD_PROCESS_PRIVATE.....          | 314, 1582, 1611, 1663, 1672, 1704, 1720, 3604 |
| PTHREAD_PROCESS_SHARED.....           | 314, 1582, 1611, 1663, 1672, 1704, 1720, 3604 |
| pthread_rwlockattr_destroy().....     | <b>1702</b> , 3604                            |
| pthread_rwlockattr_getpshared() ..... | <b>1704</b> , 3604                            |
| pthread_rwlockattr_init().....        | 1702, 1706, 3603                              |
| pthread_rwlockattr_setpshared().....  | 1704, 1707, 3604                              |
| pthread_rwlock_destroy().....         | <b>1686</b>                                   |
| pthread_rwlock_init() .....           | 1686, 3604                                    |
| PTHREAD_RWLOCK_INITIALIZER.....       | 314, 3604                                     |
| pthread_rwlock_rdlock().....          | <b>1689</b> , 3604                            |
| pthread_rwlock_t .....                | 3603                                          |
| pthread_rwlock_timedrdlock() .....    | <b>1692</b>                                   |
| pthread_rwlock_timedwrlock() .....    | <b>1694</b>                                   |
| pthread_rwlock_tryrdlock().....       | 1689, 1696, 3604                              |
| pthread_rwlock_trywrlock().....       | <b>1697</b> , 3604                            |
| pthread_rwlock_unlock() .....         | <b>1699</b> , 3604, 3618                      |
| pthread_rwlock_wrlock().....          | 1697, 1701, 3604                              |
| PTHREAD_SCOPE_PROCESS .....           | 314, 514-515, 1560                            |
| PTHREAD_SCOPE_SYSTEM .....            | 314, 514-515, 1560                            |
| pthread_self() .....                  | <b>1708</b> , 3595                            |
| pthread_setcancelstate() .....        | <b>1709</b>                                   |
| pthread_setcanceltype().....          | 1709                                          |
| pthread_setconcurrency().....         | 1624, 1711, 3604-3605                         |
| pthread_setprio() .....               | 3613                                          |
| pthread_setschedparam().....          | 1627, 1712, 3613                              |
| pthread_setschedprio().....           | <b>1713</b>                                   |
| pthread_setspecific() .....           | 1630, 1715, 3595                              |
| pthread_sigmask().....                | <b>1716</b>                                   |
| pthread_spin_destroy() .....          | <b>1720</b>                                   |
| pthread_spin_init().....              | 1720                                          |
| pthread_spin_lock() .....             | <b>1722</b> , 3597, 3617                      |
| pthread_spin_trylock() .....          | 1722, 3597                                    |
| pthread_spin_unlock().....            | <b>1724</b>                                   |
| PTHREAD_STACK_MIN .....               | 272, 1562, 1565, 2079, 3736                   |
| pthread_testcancel() .....            | 1709, 1726                                    |
| PTHREAD_THREADS_MAX .....             | 272, 1616, 2079, 3736                         |
| PTRDIFF_MAX.....                      | 351                                           |
| PTRDIFF_MIN.....                      | 351                                           |
| ptsname() .....                       | <b>1727</b>                                   |
| public locale .....                   | 2868                                          |

|                             |                        |
|-----------------------------|------------------------|
| putc()                      | 1728, 3607, 3721       |
| putchar()                   | 1730, 3721             |
| putchar_unlocked()          | 1002, 1731             |
| putc_unlocked()             | 1002, 1729             |
| putenv()                    | 1732                   |
| putmsg()                    | 1734                   |
| putpmsg()                   | 1734                   |
| puts()                      | 1738                   |
| pututxline()                | 765, 1740              |
| putwc()                     | 1741                   |
| putwchar()                  | 1742                   |
| PWD                         | 178                    |
| pwd                         | 2318, 2339, 3093, 3725 |
| pwrite()                    | 1743, 2285, 3606       |
| pw_                         | 475                    |
| p_                          | 475                    |
| P_                          | 476                    |
| P_ALL                       | 408, 2212              |
| p_cs_precedes               | 155                    |
| P_PGID                      | 408, 2212              |
| P_PID                       | 408, 2212              |
| p_sep_by_space              | 155                    |
| p_sign_posn                 | 155                    |
| P_tmpdir                    | 355                    |
| qalter                      | 3096                   |
| qdel                        | 3106                   |
| qhold                       | 3109                   |
| qmove                       | 3112                   |
| qmsg                        | 3115                   |
| qrerun                      | 3118                   |
| qrls                        | 3121                   |
| qselect                     | 3124                   |
| qsig                        | 3133                   |
| qsort()                     | 1744                   |
| qstat                       | 3136                   |
| qsub                        | 3141                   |
| queue a signal to a process | 1960                   |
| queue batch job request     | 2415                   |
| queuing of waiting threads  | 3618                   |
| quiet NaN                   | 247                    |
| quote removal               | 2333, 3689             |
| QUOTED_CHAR                 | 192                    |
| quoting                     | 2320, 3676             |
| radix character             | 82                     |
| RADIXCHAR                   | 267                    |
| raise()                     | 1746                   |
| rand()                      | 1748, 3616             |
| random()                    | 1127, 1751             |
| RAND_MAX                    | 358, 1748              |
| rand_r()                    | 1748                   |
| range error                 | 117                    |
| result overflows            | 117                    |

|                                      |                                                                                |
|--------------------------------------|--------------------------------------------------------------------------------|
| result underflows .....              | 117                                                                            |
| RCS                                  |                                                                                |
| rationale for omission .....         | 3711                                                                           |
| RE                                   |                                                                                |
| bracket expression.....              | 3502                                                                           |
| grammar .....                        | 3507                                                                           |
| read.....                            | 2318, 2339, 3154, 3675, 3723                                                   |
| read lock .....                      | 3603                                                                           |
| read() .....                         | 1752, 3456, 3511-3512, 3532, 3541-3542, 3544, 3554-3556, 3560-3561, 3605       |
| .....                                | 3616, 3648, 3721                                                               |
| read-only file system.....           | 82                                                                             |
| read-write attribute .....           | 3603                                                                           |
| read-write lock.....                 | 82, 3603                                                                       |
| readdir() .....                      | 1759, 3721                                                                     |
| readdir_r().....                     | 1759                                                                           |
| reading an active trace stream.....  | 3646                                                                           |
| reading data .....                   | 3511                                                                           |
| readlink().....                      | 1764, 3720                                                                     |
| readlinkat() .....                   | 1764                                                                           |
| readonly .....                       | 2375                                                                           |
| readv() .....                        | 1767                                                                           |
| real group ID.....                   | 83, 2301                                                                       |
| real time .....                      | 83                                                                             |
| real user ID .....                   | 83, 566, 1212, 2301                                                            |
| realloc().....                       | 1769                                                                           |
| realpath().....                      | 1771, 3720                                                                     |
| realtime .....                       | 22                                                                             |
| REALTIME.....                        | 297, 821, 1318, 1320, 1334-1335, 1337, 1340, 1343, 1346, 1348, 1352, 1814-1818 |
| .....                                | 1820, 1913, 1918                                                               |
| realtime .....                       | 3547                                                                           |
| realtime signal delivery .....       | 3538                                                                           |
| realtime signal extension.....       | 83                                                                             |
| realtime signal generation.....      | 3538                                                                           |
| realtime signals.....                | 3552                                                                           |
| REALTIME THREADS .....               | 24                                                                             |
| realtime threads.....                | 24                                                                             |
| REALTIME THREADS .....               | 1554, 1558, 1560, 1627, 1649, 1667, 1669, 1713                                 |
| record .....                         | 83                                                                             |
| recv().....                          | 1774                                                                           |
| recvfrom() .....                     | 1777                                                                           |
| recvmsg() .....                      | 1780                                                                           |
| red                                  |                                                                                |
| rationale for omission .....         | 3711                                                                           |
| redirect input .....                 | 2334, 3690                                                                     |
| redirect output.....                 | 2335, 3690                                                                     |
| redirection .....                    | 83, 2334, 3689                                                                 |
| redirection operator .....           | 84                                                                             |
| referenced shared memory object..... | 84                                                                             |
| references.....                      | 3442                                                                           |
| refresh .....                        | 84                                                                             |
| regcomp().....                       | 1783, 3726                                                                     |
| regerror().....                      | 1783, 3726                                                                     |

|                                |                                                                                                            |
|--------------------------------|------------------------------------------------------------------------------------------------------------|
| regexec()                      | 1783, 3726                                                                                                 |
| regfree()                      | 1783, 3726                                                                                                 |
| region                         | 84                                                                                                         |
| register fork handlers         | 1543                                                                                                       |
| REGTYPE                        | 412                                                                                                        |
| regular expressions            | 84, 2461, 2575, 2644, 2699, 2740, 2765, 2807, 2853, 2973, 2983<br>2995, 3039, 3162, 3181, 3353, 3410, 3500 |
| basic                          | 183                                                                                                        |
| definitions                    | 3500                                                                                                       |
| extended                       | 188                                                                                                        |
| general requirements           | 3501                                                                                                       |
| grammar                        | 191, 3506                                                                                                  |
| related to shell patterns      | 2354                                                                                                       |
| regular file                   | 84, 3462                                                                                                   |
| REG_                           | 475                                                                                                        |
| REG_ constants                 |                                                                                                            |
| defined in <regex.h>           | 322                                                                                                        |
| error return values of regcomp | 1785                                                                                                       |
| used in regcomp                | 1783                                                                                                       |
| REG_BADBR                      | 323, 1785                                                                                                  |
| REG_BADPAT                     | 322, 1785                                                                                                  |
| REG_BADRPT                     | 323, 1785                                                                                                  |
| REG_EBRACE                     | 323, 1785                                                                                                  |
| REG_EBRACK                     | 323, 1785                                                                                                  |
| REG_ECOLLATE                   | 322, 1785                                                                                                  |
| REG_ECTYPE                     | 322, 1785                                                                                                  |
| REG_EESCAPE                    | 322, 1785                                                                                                  |
| REG_EPAREN                     | 323, 1785                                                                                                  |
| REG_ERANGE                     | 323, 1785                                                                                                  |
| REG_ESPACE                     | 323, 1785                                                                                                  |
| REG_ESUBREG                    | 322, 1785                                                                                                  |
| REG_EXTENDED                   | 322, 1783                                                                                                  |
| REG_ICASE                      | 322, 1783                                                                                                  |
| REG_NEWLINE                    | 322, 1783                                                                                                  |
| REG_NOMATCH                    | 322, 1785                                                                                                  |
| REG_NOSUB                      | 322, 1783                                                                                                  |
| REG_NOTBOL                     | 322, 1784                                                                                                  |
| REG_NOTEOL                     | 322, 1784                                                                                                  |
| rejected utilities             | 3709                                                                                                       |
| relational database operator   | 2839                                                                                                       |
| relative pathname              | 85, 111                                                                                                    |
| release batch job request      | 2416                                                                                                       |
| relocatable file               | 85                                                                                                         |
| relocation                     | 85                                                                                                         |
| remainder()                    | 1790                                                                                                       |
| remainderf()                   | 1790                                                                                                       |
| remainderl()                   | 1790                                                                                                       |
| remove a directory             | 1807, 3168                                                                                                 |
| remove a directory entry       | 2177                                                                                                       |
| remove a file                  | 3161                                                                                                       |
| remove()                       | 1792                                                                                                       |
| remque()                       | 1129, 1794                                                                                                 |

|                         |                            |
|-------------------------|----------------------------|
| remquo()                | 1795                       |
| remquof()               | 1795                       |
| remquol()               | 1795                       |
| rename a file           | 1800                       |
| rename()                | 1797, 3721                 |
| renameat()              | 1797                       |
| renice                  | 3157, 3724                 |
| replenishment period    | 3573                       |
| requested batch service | 85, 2412                   |
| requirements            | 15                         |
| rerun batch job request | 2417                       |
| Rerunable               | 2409                       |
| reserved words          | 2323, 3679                 |
| Resource_List           | 2409                       |
| result overflows        | 117                        |
| result underflows       | 117                        |
| return                  | 2378                       |
| rewind()                | 1802                       |
| rewinddir()             | 1803, 3721                 |
| re_                     | 475                        |
| RE_DUP_MAX              | 272, 275, 2079, 2308, 3669 |
| rindex()                | 3651                       |
| rint()                  | 1804                       |
| rintf()                 | 1804                       |
| rintl()                 | 1804                       |
| rlimit                  | 377                        |
| RLIMIT_                 | 475                        |
| RLIMIT_AS               | 378, 1074                  |
| RLIMIT_CORE             | 377, 1073                  |
| RLIMIT_CPU              | 377, 1073                  |
| RLIMIT_DATA             | 378, 1073                  |
| RLIMIT_FSIZE            | 378, 1073                  |
| RLIMIT_NOFILE           | 378, 1073, 1075            |
| RLIMIT_STACK            | 378, 1074                  |
| rlim_                   | 475                        |
| RLIM_                   | 477                        |
| RLIM_INFINITY           | 377, 1073-1074             |
| RLIM_SAVED_CUR          | 377, 1074                  |
| RLIM_SAVED_MAX          | 377, 1074                  |
| rm                      | 3161, 3674, 3725           |
| rm del                  | 3166                       |
| rmdir                   | 3168, 3725                 |
| rmdir()                 | 1806, 3534, 3721           |
| RMSGD                   | 367, 1135                  |
| RMSGN                   | 368, 1135                  |
| RNORM                   | 368, 1135                  |
| robust mutex            | 85, 514, 1647, 3599        |
| root directory          | 85, 2301, 3462, 3478-3479  |
| root file system        | 3462                       |
| root of a file system   | 3463                       |
| round robin             | 508                        |
| round()                 | 1809                       |



|                              |                                  |
|------------------------------|----------------------------------|
| roundf()                     | 1809                             |
| roundl()                     | 1809                             |
| routing                      | 523, 3620                        |
| RPI                          | 9                                |
| RPP                          | 10                               |
| RPROTDAT                     | 368, 1135                        |
| RPROTDIS                     | 368, 1135                        |
| RPROTNORM                    | 368, 1135                        |
| RS                           | 10                               |
| rsh                          |                                  |
| rationale for omission       | 3711                             |
| RS_HIPRI                     | 367, 1041, 1134, 1734            |
| RTLD_                        | 475                              |
| RTLD_DEFAULT                 | 740                              |
| RTLD_GLOBAL                  | 233, 738                         |
| RTLD_LAZY                    | 233, 737                         |
| RTLD_LOCAL                   | 233, 738                         |
| RTLD_NEXT                    | 740                              |
| RTLD_NOW                     | 233, 738                         |
| RTSIG_MAX                    | 272, 332, 2079, 3736             |
| runnable process (or thread) | 85                               |
| running process (or thread)  | 86                               |
| runtime values               |                                  |
| increasable                  | 275                              |
| invariant                    | 270                              |
| rusage                       | 377                              |
| RUSAGE_                      | 475                              |
| RUSAGE_CHILDREN              | 377, 1076                        |
| RUSAGE_SELF                  | 377, 1076                        |
| ru_                          | 475                              |
| R_ANCHOR                     | 192                              |
| R_OK                         | 440                              |
| s6_                          | 475                              |
| sact                         | 3171                             |
| samefile()                   | 3647                             |
| saved resource limits        | 86                               |
| saved set-group-ID           | 86, 2301                         |
| saved set-user-ID            | 86, 2301                         |
| SA_                          | 475                              |
| sa_                          | 475-476                          |
| SA_ macros                   |                                  |
| declared in <signal.h>       | 334                              |
| SA_NOCLDSTOP                 | 334, 491, 1930, 1935, 3457       |
| SA_NOCLDWAIT                 | 334, 549-550, 1076, 1932, 2203   |
| SA_NODEFER                   | 334, 1932                        |
| SA_ONSTACK                   | 334, 779, 1931                   |
| SA_RESETHAND                 | 334, 1931-1932, 3649             |
| SA_RESTART                   | 334, 1539, 1931, 1947, 3649      |
| SA_SIGINFO                   | 334, 1930-1931, 1934, 1959, 3540 |
| scalb()                      | 3651                             |
| scalbln()                    | 1810                             |
| scalblnf()                   | 1810                             |

|                              |                                                                                      |
|------------------------------|--------------------------------------------------------------------------------------|
| scalblnl()                   | 1810                                                                                 |
| scalbn()                     | 1810                                                                                 |
| scalbnf()                    | 1810                                                                                 |
| scalbnl()                    | 1810                                                                                 |
| scandir()                    | 592, 1812                                                                            |
| scanf()                      | 938, 1813                                                                            |
| sccs                         | <b>3174</b>                                                                          |
| SCCS commands                |                                                                                      |
| admin                        | 2424                                                                                 |
| delta                        | 2615                                                                                 |
| get                          | 2788                                                                                 |
| prs                          | 3081                                                                                 |
| rmDEL                        | 3166                                                                                 |
| sact                         | 3171                                                                                 |
| sccs                         | 3174                                                                                 |
| unget                        | 3305                                                                                 |
| val                          | 3333                                                                                 |
| what                         | 3398                                                                                 |
| SCHAR_MAX                    | 280-281                                                                              |
| SCHAR_MIN                    | 280-281                                                                              |
| schedule alarm               | 590                                                                                  |
| scheduling                   | 86                                                                                   |
| scheduling allocation domain | 86, 3611                                                                             |
| scheduling contention scope  | 86, 3611-3612                                                                        |
| scheduling documentation     | 516, 3612                                                                            |
| scheduling policy            | 87, 112, 3479                                                                        |
| round robin                  | 508                                                                                  |
| SCHED_                       | 475                                                                                  |
| sched_                       | 475                                                                                  |
| SCHED_FIFO                   | 324, 503, 507, 515, 779, 888, 1060, 1385, 1556, 1558, 1627, 1667, 1689, 1837<br>3722 |
| sched_getparam()             | 1815                                                                                 |
| sched_getscheduler()         | 1816                                                                                 |
| sched_get_priority_max()     | 1814                                                                                 |
| sched_get_priority_min()     | 1814                                                                                 |
| SCHED_OTHER                  | 324, 507, 510, 1060, 1558, 1627                                                      |
| SCHED_RR                     | 324, 503, 507-508, 515, 779, 888, 1060, 1385, 1556, 1558, 1627, 1689, 1837<br>3722   |
| sched_rr_get_interval()      | 1817                                                                                 |
| sched_setparam()             | 1818                                                                                 |
| sched_setscheduler()         | 1820                                                                                 |
| SCHED_SPORADIC               | 324, 503, 507-508, 779, 1689, 1837, 3722                                             |
| sched_yield()                | 1822                                                                                 |
| SCM_                         | 476                                                                                  |
| SCM_RIGHTS                   | 386                                                                                  |
| SCN                          | 477                                                                                  |
| scope                        | 3439                                                                                 |
| screen                       | 87                                                                                   |
| scroll                       | 87                                                                                   |
| SD                           | <b>10</b>                                                                            |
| sdb                          |                                                                                      |
| rationale for omission       | 3711                                                                                 |

|                                       |                                                               |
|---------------------------------------|---------------------------------------------------------------|
| sdiff                                 |                                                               |
| rationale for omission .....          | 3711                                                          |
| search pattern.....                   | 2587                                                          |
| seconds since the Epoch.....          | 113, 3479-3480                                                |
| security considerations.....          | 551, 666, 957, 1212, 1889, 3450, 3454, 3458, 3471, 3473, 3511 |
| security, monolithic privileges ..... | 3450                                                          |
| sed .....                             | 3179, 3724-3725                                               |
| addresses .....                       | 3181                                                          |
| editing commands.....                 | 3181                                                          |
| regular expressions .....             | 3181                                                          |
| seed48() .....                        | 743, 1823                                                     |
| seekdir() .....                       | 1824                                                          |
| SEEK_.....                            | 477                                                           |
| SEEK_CUR.....                         | 238, 354, 442, 814, 946, 1277                                 |
| SEEK_END.....                         | 238, 354, 442, 814, 872, 946, 1277                            |
| SEEK_GET.....                         | 1802                                                          |
| SEEK_SET.....                         | 238, 354, 442, 503, 580, 587, 814, 946, 1277                  |
| SEGV_.....                            | 475                                                           |
| SEGV_ACCERR.....                      | 336                                                           |
| SEGV_MAPERR.....                      | 336                                                           |
| select batch jobs request .....       | 2417                                                          |
| select() .....                        | 1536, 1826                                                    |
| sem .....                             | 476                                                           |
| sem*().....                           | 3546                                                          |
| semaphore .....                       | 87, 113, 3481, 3550, 3722                                     |
| lock operation .....                  | 114                                                           |
| unlock operation.....                 | 114                                                           |
| semctl().....                         | 1848, 3546                                                    |
| semget().....                         | 1851, 3546                                                    |
| semid.....                            | 501                                                           |
| semop().....                          | 1853, 3546                                                    |
| SEM_.....                             | 475                                                           |
| sem_.....                             | 475                                                           |
| SEM_.....                             | 476                                                           |
| sem_close() .....                     | 1827                                                          |
| sem_destroy().....                    | 1829                                                          |
| SEM_FAILED.....                       | 328, 1835-1836                                                |
| sem_getvalue() .....                  | 1830                                                          |
| sem_init() .....                      | 1832, 3550                                                    |
| SEM_NSEMS_MAX.....                    | 272, 1832, 2079, 3736                                         |
| sem_open() .....                      | 1834, 3550                                                    |
| sem_perm.....                         | 501                                                           |
| sem_post().....                       | 1837                                                          |
| sem_timedwait() .....                 | 1839, 3581                                                    |
| sem_trywait() .....                   | 1843, 3535, 3552                                              |
| SEM_UNDO.....                         | 381, 1853                                                     |
| sem_unlink() .....                    | 1845                                                          |
| SEM_VALUE_MAX .....                   | 272, 1832, 1834, 2079, 3736                                   |
| sem_wait() .....                      | 1843, 1847, 3535, 3552                                        |
| send().....                           | 1858                                                          |
| sendmsg() .....                       | 1861                                                          |
| sendto() .....                        | 1865                                                          |

|                                            |                                             |
|--------------------------------------------|---------------------------------------------|
| sequential lists .....                     | 2343, 3696                                  |
| servent.....                               | <b>302</b>                                  |
| server shutdown request.....               | 2418                                        |
| server status request .....                | 2418                                        |
| service name .....                         | 924                                         |
| session.....                               | 88, 552, 1212, 1890, 1902, 3457, 3461, 3510 |
| session leader .....                       | 88                                          |
| session lifetime .....                     | 88                                          |
| session membership.....                    | 2301                                        |
| set.....                                   | <b>2380</b> , 3681                          |
| set cancelability state .....              | 1709                                        |
| set file creation mask.....                | 2166                                        |
| set process group ID for job control ..... | 1889                                        |
| set-group-ID.....                          | 551, 662, 786, 819, 2301, 2577              |
| set-user-ID.....                           | 551, 786, 1008, 1212, 2301, 2545, 2577      |
| set-user-ID scripts .....                  | 3203                                        |
| SETALL.....                                | 381, 1848, 1851                             |
| setbuf() .....                             | <b>1869</b>                                 |
| setegid().....                             | <b>1871</b>                                 |
| setenv().....                              | <b>1872</b>                                 |
| seteuid() .....                            | <b>1874</b>                                 |
| setgid().....                              | <b>1875</b> , 3463                          |
| setgrent().....                            | 752, 1877, 3471                             |
| sethostent() .....                         | 754, 1878                                   |
| setitimer().....                           | 1035, 1879                                  |
| setjmp() .....                             | <b>1880</b> , 3723                          |
| setkey().....                              | <b>1882</b>                                 |
| setlocale() .....                          | <b>1883</b> , 3723                          |
| extensions to .....                        | 1884                                        |
| setlogmask() .....                         | 687, 1887, 3726                             |
| setnetent().....                           | 756, 1888                                   |
| setpgid().....                             | <b>1889</b> , 3456, 3458, 3509-3510         |
| setpgrp().....                             | <b>1892</b>                                 |
| setpriority().....                         | 1060, 1893, 3571                            |
| setprotoent() .....                        | 758, 1894                                   |
| setpwent().....                            | 760, 1895, 3471                             |
| setregid().....                            | <b>1896</b>                                 |
| setreuid().....                            | <b>1898</b>                                 |
| setrlimit() .....                          | 1073, 1900, 3674                            |
| setservent() .....                         | 763, 1901                                   |
| setsid() .....                             | <b>1902</b> , 3509                          |
| setsockopt().....                          | <b>1904</b>                                 |
| setstate() .....                           | 1127, 1906                                  |
| setuid() .....                             | <b>1907</b> , 3463                          |
| setutxent().....                           | 765, 1910                                   |
| SETVAL.....                                | 381, 1848, 1851                             |
| setvbuf() .....                            | <b>1911</b>                                 |
| sh.....                                    | <b>3189</b> , 3725, 3732                    |
| command history list.....                  | 3193                                        |
| command line editing.....                  | 3193                                        |
| vi line editing command mode .....         | 3195                                        |
| vi line editing insert mode.....           | 3194                                        |

|                                                     |                                                   |
|-----------------------------------------------------|---------------------------------------------------|
| vi-mode command line editing .....                  | 3194                                              |
| shall .....                                         | 6, 3442                                           |
| rationale .....                                     | 3442                                              |
| shar                                                |                                                   |
| rationale for omission .....                        | 3711                                              |
| shared memory .....                                 | 3562                                              |
| shared memory object .....                          | 88                                                |
| shell .....                                         | 88                                                |
| SHELL .....                                         | 178                                               |
| shell .....                                         | 552, 785, 1039, 1056, 1212, 1890, 2209, 3456-3458 |
| SHELL .....                                         | 3499                                              |
| shell .....                                         | 3509, 3511, 3536, 3541-3542                       |
| commands .....                                      | 2338, 3692                                        |
| errors .....                                        | 3691                                              |
| execution environment .....                         | 2353, 2430, 3156, 3292, 3679, 3700                |
| grammar .....                                       | 2347, 3699                                        |
| grammar rules .....                                 | 2348, 3700                                        |
| grammar, lexical conventions .....                  | 2347, 3700                                        |
| introduction .....                                  | 2319                                              |
| job control .....                                   | 1212, 3456, 3536, 3542                            |
| login .....                                         | 785, 1039                                         |
| variables .....                                     | 2324, 3681                                        |
| shell command language .....                        | 2319                                              |
| alias substitution .....                            | 2322                                              |
| appending redirected output .....                   | 2335                                              |
| arithmetic expansion .....                          | 2332                                              |
| command substitution .....                          | 2331                                              |
| compound commands .....                             | 2343                                              |
| consequences of shell errors .....                  | 2337                                              |
| double-quote .....                                  | 2320                                              |
| duplicating an input file descriptor .....          | 2336                                              |
| duplicating an output file descriptor .....         | 2336                                              |
| escape character (backslash) .....                  | 2320                                              |
| exit status and errors .....                        | 2337                                              |
| exit status for commands .....                      | 2337                                              |
| field splitting .....                               | 2333                                              |
| function definition command .....                   | 2346                                              |
| grammar .....                                       | 2347                                              |
| here-document .....                                 | 2335                                              |
| introduction .....                                  | 2319                                              |
| lists .....                                         | 2341                                              |
| open file descriptors for reading and writing ..... | 2337                                              |
| parameter expansion .....                           | 2328                                              |
| parameters and variables .....                      | 2323                                              |
| pathname expansion .....                            | 2333                                              |
| pattern matching notation .....                     | 2354                                              |
| patterns matching a single character .....          | 2354                                              |
| patterns matching multiple characters .....         | 2355                                              |
| patterns used for filename expansion .....          | 2355                                              |
| pipelines .....                                     | 2341                                              |
| positional parameters .....                         | 2323                                              |
| quote removal .....                                 | 2333                                              |

|                                         |                          |
|-----------------------------------------|--------------------------|
| quoting.....                            | 2320                     |
| redirecting input.....                  | 2334                     |
| redirecting output .....                | 2335                     |
| redirection .....                       | 2334                     |
| reserved words .....                    | 2323                     |
| shell commands.....                     | 2338                     |
| shell execution environment .....       | 2353                     |
| shell grammar lexical conventions ..... | 2347                     |
| shell grammar rules .....               | 2348                     |
| shell variables .....                   | 2324                     |
| signals and error handling.....         | 2353                     |
| simple commands .....                   | 2338                     |
| single-quote.....                       | 2320                     |
| special built-in utilities .....        | 2356                     |
| special parameter .....                 | 2324                     |
| tilde expansion.....                    | 2327                     |
| token recognition.....                  | 2321                     |
| word expansions .....                   | 2327                     |
| shell script .....                      | 89                       |
| exec .....                              | 785                      |
| shell, the.....                         | 88                       |
| Shell_Path_List .....                   | 2409                     |
| shift.....                              | <b>2387</b>              |
| shl                                     |                          |
| rationale for omission .....            | 3711                     |
| SHM .....                               | <b>10</b> , 476          |
| shm .....                               | 476                      |
| shm*() .....                            | 3546                     |
| shmat() .....                           | <b>1920</b>              |
| shmctl() .....                          | <b>1922</b> , 3546       |
| shmdt() .....                           | <b>1924</b> , 3546       |
| shmget() .....                          | <b>1926</b>              |
| shmid .....                             | 501                      |
| SHMLBA .....                            | 383, 1920                |
| shm_ .....                              | 475                      |
| SHM_ .....                              | 476                      |
| shm_open().....                         | <b>1913</b> , 3561-3564  |
| shm_perm.....                           | 501                      |
| SHM_RDONLY .....                        | 383, 1920                |
| SHM_RND .....                           | 383, 1920                |
| shm_unlink().....                       | <b>1918</b> , 3562-3564  |
| should .....                            | 6, 3442                  |
| rationale.....                          | 3442                     |
| SHRT_MAX.....                           | 281                      |
| SHRT_MIN.....                           | 281                      |
| shutdown() .....                        | <b>1928</b>              |
| SHUT_ .....                             | 476                      |
| SHUT_RD.....                            | 388                      |
| SHUT_RDWR .....                         | 388                      |
| SHUT_WR.....                            | 388                      |
| SIGABRT.....                            | 332, 560, 3469, 3535     |
| sigaction() .....                       | <b>1930</b> , 3538, 3540 |

|                          |                                                                                            |
|--------------------------|--------------------------------------------------------------------------------------------|
| sigaddset()              | 1937                                                                                       |
| SIGALRM                  | 332, 590, 1035, 1978                                                                       |
| sigaltstack()            | 1938                                                                                       |
| SIGBUS                   | 332, 336, 505, 1324, 1327, 1716, 3469, 3536                                                |
| SIGCANCEL                | 1586                                                                                       |
| SIGCHLD                  | 332, 336, 549-550, 688, 1076, 1101, 1930, 1935, 1944, 2085, 2203, 2212, 3457<br>3537, 3541 |
| SIGCLD                   | 1935, 3541                                                                                 |
| SIGCONT                  | 332, 495, 550, 552, 1211-1212, 2665, 3457, 3538, 3541-3542                                 |
| sigdelset()              | 1940                                                                                       |
| sigemptyset()            | 1941                                                                                       |
| SIGEMT                   | 3536                                                                                       |
| SIGEV_                   | 475                                                                                        |
| sigev_                   | 475                                                                                        |
| SIGEV_NONE               | 331, 489, 504, 3538                                                                        |
| SIGEV_SIGNAL             | 331, 489, 2128, 3538-3539                                                                  |
| SIGEV_THREAD             | 331, 490, 1234, 3538                                                                       |
| sigfillset()             | 1943                                                                                       |
| SIGFPE                   | 332, 336, 1716, 1951, 3469, 3536, 3538                                                     |
| sighold()                | 1944                                                                                       |
| SIGHUP                   | 332, 549-550, 552, 681, 2643, 2665, 3336, 3377, 3542                                       |
| sigignore()              | 1944                                                                                       |
| SIGILL                   | 332, 336, 1716, 1951, 3469, 3536                                                           |
| siginfo_t                | 335                                                                                        |
| SIGINT                   | 332, 890, 2085, 2353, 2617, 2643, 2664, 3389, 3458, 3609                                   |
| siginterrupt()           | 1947                                                                                       |
| SIGIOT                   | 3535-3536                                                                                  |
| sigismember()            | 1949                                                                                       |
| SIGKILL                  | 332, 1212, 1930, 1934-1935, 1944, 3535, 3538, 3542                                         |
| siglongjmp()             | 1950, 3532, 3543, 3723                                                                     |
| signal                   | 89, 488, 3463, 3621, 3700                                                                  |
| acceptance               | 3537                                                                                       |
| actions                  | 3541                                                                                       |
| concepts                 | 3535                                                                                       |
| delivery                 | 488, 3537                                                                                  |
| error handling           | 2353                                                                                       |
| generation               | 488, 3537                                                                                  |
| names                    | 3535                                                                                       |
| realtime delivery        | 489                                                                                        |
| realtime generation      | 489                                                                                        |
| stack                    | 89                                                                                         |
| signal batch job request | 2418                                                                                       |
| signal handler           | 1951                                                                                       |
| signal processes         | 2844                                                                                       |
| signal()                 | 1951, 3535, 3537                                                                           |
| signaling NaN            | 247                                                                                        |
| signbit()                | 1954                                                                                       |
| signgam                  | 1226                                                                                       |
| signgam()                | 1955                                                                                       |
| sigpause()               | 1944, 1956                                                                                 |
| sigpending()             | 1957                                                                                       |
| SIGPIPE                  | 332, 811, 851, 913, 918, 947, 950, 1735, 2288, 3469, 3534                                  |

|                         |                                                                                 |
|-------------------------|---------------------------------------------------------------------------------|
| SIGPOLL.....            | 332, 336, 681, 1133-1134                                                        |
| sigprocmask().....      | 1716, 1958, 3537                                                                |
| SIGPROF.....            | 332, 1035                                                                       |
| sigqueue().....         | <b>1959</b>                                                                     |
| SIGQUEUE_MAX.....       | 272, 1959, 2079                                                                 |
| SIGQUIT.....            | 332, 2085, 2353, 2643                                                           |
| sigrlse().....          | 1944, 1961                                                                      |
| SIGRTMAX.....           | 332, 489-490, 1933, 1959, 1966, 1970, 3539-3540                                 |
| SIGRTMIN.....           | 332, 489-490, 1933, 1959, 1966, 1970, 3539-3540                                 |
| SIGSEGV.....            | 332, 336, 505, 1074, 1370, 1551, 1716, 1951, 3469, 3536                         |
| sigset().....           | 1944, 1961                                                                      |
| sigsetjmp().....        | <b>1962</b> , 3723                                                              |
| sigset_t.....           | 3535                                                                            |
| SIGSTKSZ.....           | 334, 1938                                                                       |
| SIGSTOP.....            | 332, 489, 1930, 1935, 1944, 3542                                                |
| sigsuspend().....       | <b>1964</b> , 3541, 3544                                                        |
| SIGSYS.....             | 332, 3536                                                                       |
| SIGTERM.....            | 332, 2665, 3535                                                                 |
| sigtimedwait().....     | <b>1966</b> , 3535, 3555, 3581                                                  |
| SIGTRAP.....            | 332, 336, 3536                                                                  |
| SIGTSTP.....            | 332, 489, 2710, 3458, 3542                                                      |
| SIGTTIN.....            | 332, 489, 854, 860, 1754, 3458, 3511, 3542                                      |
| SIGTTOU.....            | 332, 489, 810, 850, 913, 917, 947, 949, 2096, 2098, 2100, 2107, 2109-2110, 2112 |
| .....                   | 2288, 3457, 3511, 3542                                                          |
| SIGURG.....             | 332, 1134                                                                       |
| SIGUSR1.....            | 332, 3535                                                                       |
| SIGUSR2.....            | 332, 3535                                                                       |
| SIGVTALRM.....          | 332, 1035                                                                       |
| sigwait().....          | <b>1970</b> , 3535, 3616                                                        |
| sigwaitinfo().....      | 1966, 1972, 3535, 3555                                                          |
| sigwait_multiple()..... | 3537                                                                            |
| SIGXCPU.....            | 332, 1073                                                                       |
| SIGXFSZ.....            | 332, 1073, 2155                                                                 |
| SIG_.....               | 477                                                                             |
| SIG_ATOMIC_MAX.....     | 351                                                                             |
| SIG_ATOMIC_MIN.....     | 351                                                                             |
| SIG_BLOCK.....          | 334, 1716                                                                       |
| SIG_DFL.....            | 331, 491, 779, 1074, 1930, 1932, 1951-1952, 3537-3538, 3541                     |
| SIG_ERR.....            | 331, 1952                                                                       |
| SIG_HOLD.....           | 331, 1944                                                                       |
| SIG_IGN.....            | 331, 491, 549-550, 779, 786, 1076, 1930, 1951, 2203, 2353, 3457, 3537-3538      |
| .....                   | 3541, 3544                                                                      |
| SIG_SETMASK.....        | 334, 1716                                                                       |
| SIG_UNBLOCK.....        | 334, 1716                                                                       |
| simple commands.....    | 2338, 3692                                                                      |
| sin().....              | <b>1973</b>                                                                     |
| sin6_.....              | 475                                                                             |
| sinf().....             | 1973                                                                            |
| single-quote.....       | 89, 2320, 3676                                                                  |
| sinh().....             | <b>1975</b>                                                                     |
| sinhf().....            | 1975                                                                            |
| sinhl().....            | 1975                                                                            |



|                             |                                     |
|-----------------------------|-------------------------------------|
| sinl()                      | 1973, 1977                          |
| sin_                        | 475                                 |
| SIO                         | <b>10</b>                           |
| SIOCATMARK                  | 1981                                |
| sival_                      | 475                                 |
| size                        |                                     |
| rationale for omission      | 3711                                |
| SIZE_MAX                    | 351                                 |
| size_t                      | 401                                 |
| SI_                         | 475                                 |
| si_                         | 475                                 |
| SI_ASYNCIO                  | 336, 492                            |
| SI_MSGQ                     | 336, 492                            |
| SI_QUEUE                    | 336, 492                            |
| SI_TIMER                    | 336, 492                            |
| SI_USER                     | 336, 492, 3540                      |
| slash                       | 89                                  |
| sleep                       | <b>3207</b> , 3723                  |
| sleep()                     | <b>1978</b> , 3543-3544, 3720, 3722 |
| SLR(1) grammars             | 3429                                |
| sl_                         | 475                                 |
| SND                         | 475                                 |
| SNDTIMEO                    | 528                                 |
| SNDZERO                     | 368, 1137                           |
| snprintf()                  | 900, 1980                           |
| SO                          | 476                                 |
| sockaddr_in                 | <b>306</b>                          |
| sockaddr_in6                | <b>306</b>                          |
| socketmark()                | <b>1981</b>                         |
| socket                      | 89, 522, 3620                       |
| address                     | 89                                  |
| address families            | 522                                 |
| addressing                  | 522                                 |
| asynchronous errors         | 526                                 |
| connection indication queue | 526                                 |
| I/O mode                    | 524, 3621                           |
| Internet Protocols          | 530, 3622                           |
| IPv4                        | 531, 3622                           |
| IPv6                        | 531, 3622                           |
| local UNIX connection       | 3622                                |
| local UNIX connections      | 530                                 |
| options                     | 527                                 |
| out-of-band data            | 525                                 |
| out-of-band data state      | 3621                                |
| owner                       | 524, 3621                           |
| pending error               | 524                                 |
| protocols                   | 522                                 |
| queue limit                 | 3621                                |
| queue limits                | 524                                 |
| receive queue               | 525, 3621                           |
| signals                     | 526                                 |
| types                       | 523, 3621                           |

|                            |                                                         |
|----------------------------|---------------------------------------------------------|
| socket()                   | 1983                                                    |
| socketpair()               | 1985                                                    |
| SOCK_                      | 477                                                     |
| SOCK_DGRAM                 | 386, 530, 1983, 1985                                    |
| SOCK_RAW                   | 386, 531                                                |
| SOCK_SEQPACKET             | 386, 531, 1983, 1985                                    |
| SOCK_STREAM                | 386, 530, 1983, 1985                                    |
| soft limit                 | 90                                                      |
| software development       | 3718, 3726                                              |
| SOL_SOCKET                 | 386                                                     |
| SOMAXCONN                  | 387                                                     |
| sort                       | 3210, 3724-3725                                         |
| source code                | 90                                                      |
| SO_ACCEPTCONN              | 387, 528, 1904                                          |
| SO_BROADCAST               | 387, 528                                                |
| SO_DEBUG                   | 387, 528                                                |
| SO_DONTROUTE               | 387, 528                                                |
| SO_ERROR                   | 387, 528, 1904                                          |
| SO_KEEPAIVE                | 387, 528                                                |
| SO_LINGER                  | 387, 528                                                |
| SO_OOBINLINE               | 387, 528                                                |
| SO_RCVBUF                  | 387, 528                                                |
| SO_RCVLOWAT                | 387, 528                                                |
| SO_RCVTIMEO                | 387, 528, 1904                                          |
| SO_REUSEADDR               | 387, 528                                                |
| SO_SNDBUF                  | 387, 528                                                |
| SO_SNDLOWAT                | 387, 528                                                |
| SO_SNDTIMEO                | 387, 1904                                               |
| SO_TYPE                    | 387, 528, 1904                                          |
| space character            | 90                                                      |
| spawn                      | 90                                                      |
| spawn example              | 3652                                                    |
| special built-in           | 90, 2566, 2993, 3006, 3094, 3203, 3261, 3280, 3697-3698 |
| special built-in utilities | 2356, 3702                                              |
| break                      | 2358, 2362                                              |
| characteristics            | 2356                                                    |
| colon                      | 2360                                                    |
| dot                        | 2364                                                    |
| eval                       | 2366                                                    |
| exec                       | 2368                                                    |
| exit                       | 2370                                                    |
| export                     | 2372                                                    |
| readonly                   | 2375                                                    |
| return                     | 2378                                                    |
| set                        | 2380                                                    |
| shift                      | 2387                                                    |
| times                      | 2389                                                    |
| trap                       | 2391                                                    |
| unset                      | 2395                                                    |
| special characters         | 3512                                                    |
| special control character  | 3514                                                    |
| special parameter          | 91, 2324, 3679                                          |

|                                          |                                                          |
|------------------------------------------|----------------------------------------------------------|
| special targets.....                     | 2939                                                     |
| specific implementation .....            | 3455                                                     |
| SPEC_CHAR.....                           | 192                                                      |
| spell                                    |                                                          |
| rationale for omission .....             | 3711                                                     |
| spin lock .....                          | 91, 3597-3598                                            |
| split.....                               | <b>3217</b> , 3724                                       |
| split files                              |                                                          |
| csplit .....                             | 2583                                                     |
| split.....                               | 3217                                                     |
| SPN.....                                 | <b>10</b>                                                |
| spoofing .....                           | 2376                                                     |
| sporadic server .....                    | 91                                                       |
| sporadic server policy                   |                                                          |
| execution capacity .....                 | 508                                                      |
| replenishment period .....               | 508                                                      |
| scheduling .....                         | 3573                                                     |
| sprintf() .....                          | 900, 1987                                                |
| spurious wakeup.....                     | 1594                                                     |
| sqrt() .....                             | <b>1988</b>                                              |
| sqrth() .....                            | 1988                                                     |
| sqrthl() .....                           | 1988                                                     |
| srand() .....                            | 1748, 1990                                               |
| srand48() .....                          | 743, 1991                                                |
| srandom().....                           | 1127, 1992                                               |
| SS .....                                 | <b>10</b>                                                |
| sscanf() .....                           | 938, 1993                                                |
| SSIZE_MAX .....                          | 281, 403, 1343, 1359, 1752, 1764, 2019, 2285, 3648, 3736 |
| ssize_t.....                             | 401                                                      |
| SS_ .....                                | 475                                                      |
| ss_ .....                                | 475-476                                                  |
| SS_DISABLE .....                         | 334, 1938-1939                                           |
| SS_ONSTACK.....                          | 334, 1938                                                |
| SS_REPL_MAX.....                         | 272, 3576                                                |
| stack size.....                          | 1546                                                     |
| stack_t .....                            | <b>334</b>                                               |
| standard error .....                     | 91, 2334                                                 |
| standard I/O stream.....                 | 495, 3545                                                |
| standard input .....                     | 91, 2334                                                 |
| standard output.....                     | 91, 2334                                                 |
| standard utilities.....                  | 91                                                       |
| START .....                              | 2098                                                     |
| stat .....                               | 3674                                                     |
| stat data structure.....                 | <b>391</b>                                               |
| stat() .....                             | 954, 1994, 3453, 3561, 3674, 3720                        |
| state-dependent character encoding ..... | 3483                                                     |
| statvfs().....                           | 960, 1995, 3674                                          |
| stderr .....                             | 355, 1996, 3545                                          |
| STDERR_FILENO .....                      | 446, 1996                                                |
| stdin.....                               | 355, 1996, 3545                                          |
| STDIN_FILENO .....                       | 446, 1420, 1996                                          |

|                                         |                                                                                |
|-----------------------------------------|--------------------------------------------------------------------------------|
| stdio                                   |                                                                                |
| locking functions .....                 | 865                                                                            |
| with explicit client locking.....       | 1002                                                                           |
| stdout .....                            | 355, 1996, 3545                                                                |
| STDOUT_FILENO .....                     | 446, 1420, 1996                                                                |
| STOP.....                               | 2098                                                                           |
| stpcpy() .....                          | 1998, 2008                                                                     |
| stpncpy() .....                         | 1999, 2036                                                                     |
| STR .....                               | 477                                                                            |
| strbuf .....                            | 365                                                                            |
| strcasemp().....                        | 2000                                                                           |
| strcasemp_l() .....                     | 2000                                                                           |
| strcat().....                           | 2002                                                                           |
| strchr() .....                          | 2003                                                                           |
| strcmp() .....                          | 2004                                                                           |
| strcoll() .....                         | 2006                                                                           |
| strcoll_l() .....                       | 2006                                                                           |
| strcpy() .....                          | 2008                                                                           |
| strcspn().....                          | 2011                                                                           |
| strdup() .....                          | 2012                                                                           |
| stream.....                             | 92                                                                             |
| byte-oriented.....                      | 498                                                                            |
| interaction with file descriptors ..... | 496                                                                            |
| stream orientation .....                | 498                                                                            |
| wide-oriented.....                      | 498                                                                            |
| stream-full-policy attribute .....      | 537-538, 540, 1478                                                             |
| stream-min-size attribute .....         | 540, 1481                                                                      |
| STREAMS .....                           | 25, 92, 236, 365, 482, 681, 800, 822, 1041, 1132, 1149, 1393, 1416, 1536, 1734 |
| .....                                   | 1753, 2287, 3545                                                               |
| access.....                             | 500                                                                            |
| end .....                               | 92                                                                             |
| head .....                              | 92                                                                             |
| head/tail.....                          | 499                                                                            |
| multiplexed .....                       | 1140                                                                           |
| multiplexor.....                        | 92                                                                             |
| overview .....                          | 499                                                                            |
| STREAM_MAX .....                        | 272, 826, 884, 1420, 2079, 2139, 3736                                          |
| strerror() .....                        | 2014                                                                           |
| strerror_l().....                       | 2014                                                                           |
| strerror_r() .....                      | 2014                                                                           |
| strfdinsert .....                       | 365                                                                            |
| strfmon() .....                         | 2017                                                                           |
| strfmon_l() .....                       | 2017                                                                           |
| strftime() .....                        | 2022                                                                           |
| strftime_l() .....                      | 2022                                                                           |
| string .....                            | 92                                                                             |
| strings.....                            | 3221, 3726                                                                     |
| strioctl .....                          | 365                                                                            |
| strip .....                             | 3224, 3726                                                                     |
| strlen() .....                          | 2031                                                                           |
| strncasecmp() .....                     | 2000, 2033                                                                     |
| strncasecmp_l() .....                   | 2000, 2033                                                                     |

|                                       |                  |
|---------------------------------------|------------------|
| strncat()                             | 2034             |
| strncmp()                             | 2035             |
| strncpy()                             | 2036             |
| strndup()                             | 2012, 2038       |
| strnlen()                             | 2039             |
| strpbrk()                             | 2040             |
| strpeek                               | 365              |
| strptime()                            | 2041             |
| strchr()                              | 2046             |
| strrecvfd                             | 365              |
| strsignal()                           | 2047             |
| strspn()                              | 2048             |
| strstr()                              | 2049             |
| strtod()                              | 2050             |
| strtof()                              | 2050             |
| strtoimax()                           | 2054             |
| strtok()                              | 2056             |
| strtok_r()                            | 2056             |
| strtol()                              | 2059             |
| strtold()                             | 2050, 2062       |
| strtoll()                             | 2059, 2063       |
| strtoul()                             | 2064             |
| strtoull()                            | 2064             |
| strtoumax()                           | 2054, 2067       |
| structures, additions to              | 3528             |
| strxfrm()                             | 2068             |
| strxfrm_l()                           | 2068             |
| str_                                  | 475              |
| str_list                              | 365              |
| str_mlist                             | 366              |
| stty                                  | 3226, 3498, 3726 |
| combination modes                     | 3231             |
| control modes                         | 3226             |
| input modes                           | 3227             |
| local modes                           | 3229             |
| output modes                          | 3228             |
| special control character assignments | 3230             |
| ST_                                   | 476              |
| st_                                   | 476              |
| st_gid                                | 2437             |
| st_mode                               | 2437             |
| st_mtime                              | 2437             |
| ST_NOSUID                             | 396, 779, 960    |
| ST_RDONLY                             | 396, 960         |
| st_size                               | 2437             |
| st_uid                                | 2437             |
| su                                    |                  |
| rationale for omission                | 3711             |
| subprofiling                          | 20, 3447         |
| subprofiling option groups            | 3741             |
| subshell                              | 93, 3458         |
| successfully completed                | 3470             |

|                                     |                                                                      |
|-------------------------------------|----------------------------------------------------------------------|
| successfully transferred.....       | 93                                                                   |
| sum.....                            | 3674                                                                 |
| rationale for omission.....         | 3712                                                                 |
| sun_.....                           | 476                                                                  |
| superuser.....                      | 566, 666, 1230, 2177, 2749, 2895, 3058, 3450, 3463, 3473, 3681, 3709 |
| supplementary group ID.....         | 93, 2301, 3463                                                       |
| supplementary groups.....           | 666, 1030, 3473                                                      |
| supported threads functions.....    | 3600                                                                 |
| suseconds_t.....                    | 401                                                                  |
| suspended job.....                  | 93                                                                   |
| SVID.....                           | 1962                                                                 |
| SVR4.....                           | 1326, 1373                                                           |
| sv_.....                            | 475                                                                  |
| SV_.....                            | 477                                                                  |
| swab().....                         | <b>2070</b>                                                          |
| swapcontext().....                  | 3650                                                                 |
| swprintf().....                     | 982, 2071                                                            |
| swscanf().....                      | 992, 2072                                                            |
| SWTCH.....                          | 477                                                                  |
| symbolic constant.....              | 93, 3444, 3464                                                       |
| symbolic link.....                  | 94, 3465                                                             |
| symbolic name.....                  | 3444                                                                 |
| symbols.....                        | 3526                                                                 |
| POSIX.1.....                        | 472                                                                  |
| symlink().....                      | <b>2073</b>                                                          |
| symlinkat().....                    | 2073                                                                 |
| SYMLINK_MAX.....                    | 274, 283, 893, 2074                                                  |
| SYMLOOP_MAX.....                    | 272, 2079, 3532                                                      |
| SYMTYPE.....                        | 412                                                                  |
| sync().....                         | <b>2076</b>                                                          |
| synchronized I/O.....               | 3555, 3721                                                           |
| completion.....                     | 94                                                                   |
| data integrity completion.....      | 94, 3470, 3555                                                       |
| file integrity completion.....      | 94, 3470, 3555                                                       |
| synchronized I/O operation.....     | 94                                                                   |
| synchronized input and output.....  | 94                                                                   |
| synchronous I/O operation.....      | 95                                                                   |
| synchronously accept a signal.....  | 1967                                                                 |
| synchronously-generated signal..... | 95, 3469                                                             |
| sysconf().....                      | <b>2077</b> , 3455, 3558, 3560, 3562, 3609, 3667, 3719, 3721         |
| syslog().....                       | 687, 2084, 3726                                                      |
| system.....                         | 95                                                                   |
| boot.....                           | 95                                                                   |
| call.....                           | 3469                                                                 |
| clock.....                          | 95                                                                   |
| configuration values.....           | 2796                                                                 |
| console.....                        | 95, 3470                                                             |
| crash.....                          | 95, 963                                                              |
| database.....                       | 3470                                                                 |
| databases.....                      | 96                                                                   |
| documentation.....                  | 96                                                                   |
| name.....                           | 2168, 3296                                                           |

|                                    |                                                                              |
|------------------------------------|------------------------------------------------------------------------------|
| process .....                      | 96, 3470                                                                     |
| reboot .....                       | 96, 3470                                                                     |
| trace event .....                  | 96                                                                           |
| trace event type definitions ..... | 540                                                                          |
| system documentation .....         | 3443                                                                         |
| system environment .....           | 3718, 3726                                                                   |
| System III .....                   | 666, 2168, 3462, 3647                                                        |
| system interfaces .....            | 547, 3649                                                                    |
| System V .....                     | 552, 590, 666, 787, 818, 895, 1056, 1212, 1302, 1807, 1902, 1935, 1962, 2102 |
| .....                              | 2168, 3453, 3458, 3536                                                       |
| system() .....                     | 2085, 3723, 3726-3727                                                        |
| system-wide .....                  | 96                                                                           |
| S_ .....                           | 475                                                                          |
| s_ .....                           | 475                                                                          |
| S_ .....                           | 477                                                                          |
| S_ constants                       |                                                                              |
| defined in <sys/stat.h> .....      | 392                                                                          |
| S_ macros                          |                                                                              |
| defined in <sys/stat.h> .....      | 392                                                                          |
| S_BANDURG .....                    | 367, 1134                                                                    |
| S_ERROR .....                      | 367, 1134                                                                    |
| S_HANGUP .....                     | 367, 1134                                                                    |
| S_HIPRI .....                      | 367, 1133                                                                    |
| S_IFBLK .....                      | 392, 1311                                                                    |
| S_IFCHR .....                      | 392, 1311                                                                    |
| S_IFDIR .....                      | 392, 1311                                                                    |
| S_IFIFO .....                      | 392, 1311                                                                    |
| S_IFLNK .....                      | 392                                                                          |
| S_IFMT .....                       | 392                                                                          |
| S_IFREG .....                      | 392, 1311                                                                    |
| S_IFSOCK .....                     | 392                                                                          |
| S_INPUT .....                      | 367, 1133                                                                    |
| S_IRGRP .....                      | 804, 951, 954, 1311                                                          |
| S_IROTH .....                      | 804, 951, 954, 1311                                                          |
| S_IRUSR .....                      | 804, 951, 954, 1311                                                          |
| S_IRWXG .....                      | 1311                                                                         |
| S_IRWXO .....                      | 1311                                                                         |
| S_IRWXU .....                      | 1311                                                                         |
| S_ISBLK .....                      | 392                                                                          |
| S_ISCHR .....                      | 392                                                                          |
| S_ISDIR .....                      | 392                                                                          |
| S_ISFIFO .....                     | 392                                                                          |
| S_ISGID .....                      | 394, 660, 662, 1311, 2155, 2286                                              |
| S_ISLNK .....                      | 393                                                                          |
| S_ISREG .....                      | 392                                                                          |
| S_ISSOCK .....                     | 393                                                                          |
| S_ISUID .....                      | 394, 660, 662, 1311, 2155, 2286                                              |
| S_ISVTX .....                      | 660, 1311                                                                    |
| S_IWGRP .....                      | 804, 951, 954, 1311                                                          |
| S_IWOTH .....                      | 804, 951, 954, 1311                                                          |
| S_IWUSR .....                      | 804, 951, 954, 1311                                                          |
| S_IXGRP .....                      | 1311                                                                         |

|                             |                  |
|-----------------------------|------------------|
| S_IXOTH.....                | 1311             |
| S_IXUSR.....                | 1311             |
| S_MSG.....                  | 367, 1133        |
| S_OUTPUT.....               | 367, 1133        |
| S_RDBAND.....               | 367, 1133-1134   |
| S_RDNORM.....               | 367, 1133        |
| S_TYPEISMQ.....             | 393              |
| S_TYPEISSEM.....            | 393              |
| S_TYPEISSHM.....            | 393              |
| S_TYPEISTMO.....            | 393              |
| S_WRBAND.....               | 367, 1133        |
| S_WRNORM.....               | 367, 1133        |
| tab character.....          | 97               |
| TABDLY.....                 | 416              |
| TABn.....                   | 416              |
| tabs.....                   | 3235, 3724       |
| TABSIZE.....                | 624, 1275        |
| tag file creation.....      | 2587             |
| tail.....                   | 3239, 3724       |
| talk.....                   | 3243, 3725       |
| tan().....                  | 2090             |
| tanf().....                 | 2090             |
| tanh().....                 | 2093             |
| tanhf().....                | 2093             |
| tanh1().....                | 2093             |
| tanl().....                 | 2090, 2095       |
| tar                         |                  |
| rationale for omission..... | 3712             |
| tar format.....             | 3049             |
| target queue.....           | 2415             |
| target rule.....            | 2934             |
| tcdrain().....              | 2096             |
| tcflow().....               | 2098             |
| tcflush().....              | 2100             |
| tcgetattr().....            | 2102, 3457       |
| tcgetpgrp().....            | 2104, 3458, 3510 |
| tcgetsid().....             | 2106             |
| TCIFLUSH.....               | 418, 2100        |
| TCIOFF.....                 | 418, 2098        |
| TCIOFLUSH.....              | 418, 2100        |
| TCION.....                  | 418, 2098        |
| TCOFLUSH.....               | 2100             |
| TCOOFF.....                 | 418, 2098        |
| TCOON.....                  | 418, 2098        |
| TCP_.....                   | 475              |
| TCP_NODELAY.....            | 310              |
| TCSADRAIN.....              | 418, 2109        |
| TCSAFLUSH.....              | 418, 2109        |
| TCSANOW.....                | 418, 2109        |
| tcsendbreak().....          | 2107             |
| tcsetattr().....            | 2109, 3457, 3508 |
| tcsetpgrp().....            | 2112, 3457-3458  |



|                                           |                  |
|-------------------------------------------|------------------|
| TCT.....                                  | 10               |
| tdelete() .....                           | 2114             |
| tee .....                                 | 3247, 3723       |
| TEF.....                                  | 11               |
| telldir().....                            | 2118             |
| tempnam() .....                           | 2119             |
| TERM .....                                | 178, 3498        |
| terminal.....                             | 97               |
| access control.....                       | 2102, 2110, 3511 |
| controlling .....                         | 200              |
| device file.....                          | 3509             |
| device file, closing .....                | 3513             |
| device name .....                         | 2159             |
| type.....                                 | 198, 3508        |
| terminal characteristics                  |                  |
| stty .....                                | 3226             |
| tabs .....                                | 3235             |
| tput .....                                | 3269             |
| tty.....                                  | 3284             |
| terminal device .....                     | 97               |
| terminate a process .....                 | 551, 2844        |
| terminology.....                          | 3442             |
| termios .....                             | 199              |
| canonical mode input processing .....     | 202              |
| control modes .....                       | 209              |
| controlling terminal .....                | 200              |
| input modes .....                         | 206              |
| local modes.....                          | 210              |
| non-canonical mode input processing ..... | 202              |
| output modes.....                         | 207              |
| process group.....                        | 200              |
| special control characters .....          | 212              |
| termios structure .....                   | 2102, 3513       |
| test .....                                | 3250, 3723, 3725 |
| TeX.....                                  | 3725             |
| text column.....                          | 97               |
| text file .....                           | 97, 3470         |
| tfind() .....                             | 2114, 2121       |
| tgamma().....                             | 2122             |
| tgammaf() .....                           | 2122             |
| tgamma1() .....                           | 2122             |
| TGEXEC.....                               | 412              |
| TGREAD.....                               | 412              |
| TGWRITE .....                             | 412              |
| THOUSEP.....                              | 267              |
| thread .....                              | 97, 3471         |
| thread cancelability                      |                  |
| states.....                               | 3616             |
| type.....                                 | 3616             |
| thread cancellation.....                  | 3614, 3616       |
| cleanup handlers .....                    | 520              |
| thread cancellation points.....           | 3616             |

|                                      |                                     |
|--------------------------------------|-------------------------------------|
| thread concurrency level.....        | 3604                                |
| thread creation.....                 | 1617                                |
| thread creation attributes.....      | 1546, 3593                          |
| thread ID.....                       | 97, 513, 1621, 3471, 3609           |
| thread interactions.....             | 3620                                |
| thread list.....                     | 98                                  |
| thread mutex.....                    | 513, 3610                           |
| thread read-write lock.....          | 3618                                |
| thread scheduling.....               | 514, 3610                           |
| thread stack guard size.....         | 3605                                |
| thread termination.....              | 1622                                |
| thread-safe function.....            | 3471                                |
| thread-safety.....                   | 98, 114, 512, 865, 3471, 3481, 3606 |
| rationale.....                       | 3481                                |
| thread-specific data.....            | 3595                                |
| thread-specific data key.....        | 98                                  |
| creation.....                        | 1636                                |
| deletion.....                        | 1638                                |
| thread-specific data management..... | 1630                                |
| threads.....                         | 512, 3592                           |
| extensions.....                      | 3601                                |
| implementation models.....           | 3594                                |
| regular file operations.....         | 521                                 |
| tilde.....                           | 98                                  |
| tilde expansion.....                 | 2327, 3683                          |
| time.....                            | 3259, 3723, 3726                    |
| time().....                          | 2125, 3532                          |
| timeouts.....                        | 98, 3585                            |
| timer.....                           | 98                                  |
| ID.....                              | 2130                                |
| overflow.....                        | 98                                  |
| timers.....                          | 3577                                |
| TIMER_.....                          | 476                                 |
| timer_.....                          | 476                                 |
| TIMER_ABSTIME.....                   | 425, 511, 676, 2132, 3577-3579      |
| timer_create().....                  | 2128                                |
| timer_delete().....                  | 2131                                |
| timer_getoverrun().....              | 2132                                |
| timer_gettime().....                 | 2132                                |
| TIMER_MAX.....                       | 273, 2079, 3736                     |
| timer_settime().....                 | 2132, 3577-3579                     |
| timer_t.....                         | 401                                 |
| times.....                           | 2389                                |
| times().....                         | 2135, 3532, 3584, 3720              |
| timespec.....                        | 424, 3480                           |
| timestamp clock.....                 | 3645                                |
| timeval.....                         | 379, 398                            |
| timezone().....                      | 2138                                |
| time_t.....                          | 401, 3480                           |
| tm.....                              | 424                                 |
| TMAGIC.....                          | 412                                 |
| TMAGLEN.....                         | 412                                 |

|                               |                                               |
|-------------------------------|-----------------------------------------------|
| TMPDIR.....                   | 178, 3036                                     |
| tmpfile().....                | 2139                                          |
| tmpnam().....                 | 2141                                          |
| TMP_MAX.....                  | 354, 2119, 2140-2141                          |
| tms.....                      | 400                                           |
| tms_.....                     | 476                                           |
| tm_.....                      | 476                                           |
| toascii().....                | 2143                                          |
| TOEXEC.....                   | 412                                           |
| token.....                    | 99                                            |
| token recognition.....        | 2321, 3678                                    |
| tolower().....                | 2144                                          |
| tolower_l().....              | 2144                                          |
| TOREAD.....                   | 412                                           |
| TOSTOP.....                   | 417, 810, 850, 913, 917, 947, 949, 2288, 3457 |
| touch.....                    | 3263, 3674, 3725                              |
| toupper().....                | 2146                                          |
| toupper_l().....              | 2146                                          |
| towctrans().....              | 2148                                          |
| towctrans_l().....            | 2148                                          |
| towlower().....               | 2150                                          |
| towlower_l().....             | 2150                                          |
| TOWRITE.....                  | 412                                           |
| towupper().....               | 2152                                          |
| towupper_l().....             | 2152                                          |
| TPI.....                      | 11                                            |
| TPP.....                      | 11                                            |
| TPS.....                      | 11                                            |
| tput.....                     | 3269, 3726                                    |
| tr.....                       | 3272, 3724-3725                               |
| trace analyzer.....           | 3634                                          |
| trace analyzer process.....   | 99                                            |
| trace controller process..... | 99                                            |
| trace event.....              | 99                                            |
| POSIX_TRACE_ERROR.....        | 541                                           |
| POSIX_TRACE_FILTER.....       | 541, 1512                                     |
| POSIX_TRACE_OVERFLOW.....     | 541                                           |
| POSIX_TRACE_RESUME.....       | 541                                           |
| POSIX_TRACE_START.....        | 541, 1521                                     |
| POSIX_TRACE_STOP.....         | 541, 1521                                     |
| trace event type.....         | 99, 3643                                      |
| filtering.....                | 3643                                          |
| mapping.....                  | 99                                            |
| trace examples.....           | 3632                                          |
| trace filter.....             | 99                                            |
| trace functions.....          | 544                                           |
| trace generation version..... | 99                                            |
| trace log.....                | 100                                           |
| trace model.....              | 3627                                          |
| trace operation control.....  | 3632                                          |
| trace point.....              | 100                                           |
| trace storage.....            | 3631                                          |

|                                   |                                                                              |
|-----------------------------------|------------------------------------------------------------------------------|
| trace stream .....                | 100                                                                          |
| attribute .....                   | 3637                                                                         |
| identifier .....                  | 100                                                                          |
| states .....                      | 3630                                                                         |
| trace system .....                | 100                                                                          |
| trace-name attribute .....        | 540, 1475                                                                    |
| traced process .....              | 100                                                                          |
| TRACE_EVENT_NAME_MAX .....        | 273, 1500, 1502                                                              |
| TRACE_NAME_MAX .....              | 273                                                                          |
| TRACE_SYS_MAX .....               | 273, 1497                                                                    |
| TRACE_USER_EVENT_MAX .....        | 273, 1500, 1502                                                              |
| tracing .....                     | 25, 114                                                                      |
| TRACING .....                     | 1473, 1475, 1477, 1480, 1492, 1494, 1496, 1500, 1502, 1505, 1507, 1510, 1512 |
| .....                             | 1515, 1521                                                                   |
| tracing .....                     | 3481, 3622                                                                   |
| all processes .....               | 3631                                                                         |
| detailed objectives .....         | 3623                                                                         |
| status of a trace stream .....    | 100                                                                          |
| track batch job request .....     | 2419                                                                         |
| trap .....                        | <b>2391</b>                                                                  |
| TRAP_ .....                       | 475                                                                          |
| TRAP_BRKPT .....                  | 336                                                                          |
| TRAP_TRACE .....                  | 336                                                                          |
| TRC .....                         | <b>11</b>                                                                    |
| TRI .....                         | <b>11</b>                                                                    |
| triggering .....                  | 3645                                                                         |
| TRL .....                         | <b>11</b>                                                                    |
| troff .....                       | 3725                                                                         |
| trojan horse .....                | 2896, 3450                                                                   |
| true .....                        | 2318, 2339, 3279, 3675, 3723                                                 |
| trunc() .....                     | <b>2154</b>                                                                  |
| truncate() .....                  | <b>2155</b>                                                                  |
| truncation-status attribute ..... | 1500                                                                         |
| truncf() .....                    | 2154, 2157                                                                   |
| truncl() .....                    | 2154, 2157                                                                   |
| TSA .....                         | <b>12</b>                                                                    |
| tsearch() .....                   | 2114, 2158                                                                   |
| TSGID .....                       | 412                                                                          |
| TSH .....                         | <b>12</b>                                                                    |
| tsort .....                       | <b>3281</b>                                                                  |
| TSP .....                         | <b>12</b>                                                                    |
| TSS .....                         | <b>12</b>                                                                    |
| TSUID .....                       | 412                                                                          |
| TSVTX .....                       | 412                                                                          |
| tty .....                         | <b>3284</b> , 3726                                                           |
| ttynam() .....                    | <b>2159</b> , 3606                                                           |
| ttynam_r() .....                  | 2159                                                                         |
| TTY_NAME_MAX .....                | 273, 2079, 2159, 3736                                                        |
| TUEXEC .....                      | 412                                                                          |
| TUREAD .....                      | 412                                                                          |
| TUWRITE .....                     | 412                                                                          |
| TVERSION .....                    | 412                                                                          |

|                                    |                              |
|------------------------------------|------------------------------|
| TVERSLEN .....                     | 412                          |
| tv_ .....                          | 476                          |
| twalk() .....                      | 2114, 2161                   |
| TYM .....                          | 12                           |
| type .....                         | 3286                         |
| typed memory .....                 | 3564                         |
| name space .....                   | 100                          |
| object .....                       | 101                          |
| pool .....                         | 101                          |
| port .....                         | 101                          |
| TZ .....                           | 179, 3499                    |
| tzname .....                       | 2162                         |
| TZNAME_MAX .....                   | 273, 2079, 3736              |
| tzset .....                        | 2162                         |
| tzset() .....                      | 2162, 3723                   |
| T_FMT .....                        | 267                          |
| T_FMT_AMPM .....                   | 267                          |
| t_scalar_t .....                   | 365                          |
| t_uscalar_t .....                  | 365, 1135                    |
| ualarm() .....                     | 3651, 3720                   |
| UCHAR_MAX .....                    | 280-281                      |
| ucontext_t .....                   | 334                          |
| uc_ .....                          | 475                          |
| UID_MAX .....                      | 3648                         |
| uid_t .....                        | 401, 3471                    |
| UINT .....                         | 477                          |
| UINTMAX_MAX .....                  | 351                          |
| UINTN_MAX .....                    | 350                          |
| UINTPTR_MAX .....                  | 351                          |
| UINT_FASTN_MAX .....               | 350                          |
| UINT_LEASTN_MAX .....              | 350                          |
| UINT_MAX .....                     | 281, 590, 1979               |
| UIO_MAXIOV .....                   | 476                          |
| ulimit .....                       | 3288                         |
| ulimit() .....                     | 2164                         |
| ULLONG_MAX .....                   | 281, 2065                    |
| ULONG_MAX .....                    | 281, 2065, 2260, 3667        |
| UL_ .....                          | 476                          |
| UL_GETFSIZE .....                  | 432, 2164                    |
| UL_SETFSIZE .....                  | 432, 2164                    |
| umask .....                        | 2318, 2339, 3290, 3675, 3726 |
| umask() .....                      | 2166, 3720                   |
| umount() .....                     | 3460                         |
| unalias .....                      | 2318, 2339, 3294, 3675, 3724 |
| uname .....                        | 3296, 3726                   |
| uname() .....                      | 2168, 3719                   |
| unary primaries .....              | 3252                         |
| unbind .....                       | 101                          |
| unbounded priority inversion ..... | 3613                         |
| uncompress .....                   | 3299                         |
| undefined .....                    | 6, 3443                      |
| rationale .....                    | 3443                         |

|                                         |                                   |
|-----------------------------------------|-----------------------------------|
| underlying function .....               | 497                               |
| unexpand .....                          | 3302, 3724                        |
| unget .....                             | 3305                              |
| ungetc() .....                          | 2170                              |
| ungetwc() .....                         | 2172                              |
| unicast .....                           | 531                               |
| uniq .....                              | 3308, 3724-3725                   |
| unlink .....                            | 3312                              |
| unlink() .....                          | 2174, 3534, 3561-3562, 3564, 3721 |
| unlinkat() .....                        | 2174                              |
| unlockpt() .....                        | 2180                              |
| unpack                                  |                                   |
| rationale for omission .....            | 3712                              |
| unsafe functions .....                  | 3542                              |
| unset .....                             | 2395                              |
| unsetenv() .....                        | 2181                              |
| unspecified .....                       | 6, 3443                           |
| rationale .....                         | 3443                              |
| until loop .....                        | 2346, 3697                        |
| UP .....                                | 12                                |
| upper multiplexing .....                | 92                                |
| upshifting .....                        | 101                               |
| US-ASCII .....                          | 1148                              |
| uselocale() .....                       | 2182                              |
| user database .....                     | 101, 3471                         |
| access .....                            | 3472                              |
| user ID .....                           | 102, 2822                         |
| logname .....                           | 2881                              |
| newgrp .....                            | 2987                              |
| real and effective .....                | 1898                              |
| setting real and effective .....        | 1898                              |
| who .....                               | 3401                              |
| user name .....                         | 102                               |
| user requirements .....                 | 3715                              |
| user trace event .....                  | 102                               |
| user trace event type definitions ..... | 543                               |
| User_List .....                         | 2410                              |
| USER_PROCESS .....                      | 455, 765-766                      |
| USHRT_MAX .....                         | 281                               |
| usleep() .....                          | 3651, 3720                        |
| ustar format .....                      | 3049                              |
| UTC .....                               | 2162                              |
| utility .....                           | 102, 118, 3481                    |
| argument syntax .....                   | 3514                              |
| conventions .....                       | 3514                              |
| description defaults .....              | 3670                              |
| limits .....                            | 3667                              |
| option parsing .....                    | 2802                              |
| syntax guidelines .....                 | 215, 3515                         |
| utimbuf .....                           | 454                               |
| utime() .....                           | 2184, 3720                        |
| utimensat() .....                       | 976, 2187                         |

|                        |                             |
|------------------------|-----------------------------|
| utimes()               | 976, 2187                   |
| UTIME_NOW              | 393, 976                    |
| UTIME_OMIT             | 393, 976                    |
| utim_                  | 476                         |
| utmpx                  | 455                         |
| uts_                   | 476                         |
| ut_                    | 476                         |
| UU                     | 12                          |
| uucp                   | 3314, 3725                  |
| uudecode               | 3318, 3724-3725             |
| uuencode               | 3321, 3724-3725             |
| uustat                 | 3326                        |
| uux                    | 3329                        |
| val                    | 3333                        |
| variable               | 103, 3679                   |
| variable assignment    | 118, 3481                   |
| Variable_List          | 2410                        |
| va_arg()               | 2188                        |
| va_copy()              | 2188                        |
| va_end()               | 2188                        |
| va_start()             | 2188                        |
| VDISCARD               | 477                         |
| vdprintf()             | 2189                        |
| VDSUSP                 | 477                         |
| VEOF                   | 414, 3514                   |
| VEOL                   | 414, 3514                   |
| VERASE                 | 414                         |
| Version 7              | 590, 1212, 2168, 3478, 3676 |
| vertical-tab character | 103                         |
| vfork()                | 3651                        |
| vfprintf()             | 2189                        |
| VFS                    | 396                         |
| vfscanf()              | 2191                        |
| vfwprintf()            | 2192                        |
| vfwscanf()             | 2194                        |
| vhangup()              | 3459                        |
| vi                     | 3336, 3724-3725             |
| <ESC>                  | 3376                        |
| append                 | 3357                        |
| change                 | 3358                        |
| change to end-of-line  | 3359                        |
| clear and redisplay    | 3344                        |
| command descriptions   | 3337                        |
| control-D              | 3373                        |
| control-H              | 3373                        |
| control-T              | 3375                        |
| control-U              | 3375                        |
| control-V              | 3375                        |
| current and line above | 3351                        |
| delete                 | 3359                        |
| delete character       | 3369                        |
| delete to end-of-line  | 3360                        |

|                                      |                            |
|--------------------------------------|----------------------------|
| display information .....            | 3343                       |
| edit the alternate file .....        | 3345                       |
| enter ex mode .....                  | 3365                       |
| execute .....                        | 3356                       |
| execute an ex command .....          | 3355                       |
| exit .....                           | 3371                       |
| find character.....                  | 3360-3361                  |
| find regular expression.....         | 3353                       |
| Initialization.....                  | 3337                       |
| input mode commands .....            | 3371                       |
| insert.....                          | 3362                       |
| insert empty line.....               | 3364                       |
| join .....                           | 3362                       |
| mark position.....                   | 3363                       |
| move back.....                       | 3351-3352, 3357-3358       |
| move cursor.....                     | 3343, 3346-3347, 3366-3367 |
| move down .....                      | 3344                       |
| move forward.....                    | 3351-3352                  |
| move to bigword .....                | 3360, 3368                 |
| move to bottom of screen.....        | 3362                       |
| move to first character in line..... | 3355                       |
| move to first non-<blank> .....      | 3351                       |
| move to line.....                    | 3361                       |
| move to matching character .....     | 3348                       |
| move to middle of screen.....        | 3363                       |
| move to next section .....           | 3350                       |
| move to specific column.....         | 3352                       |
| move to top of screen.....           | 3361                       |
| move to word.....                    | 3360, 3368                 |
| move up.....                         | 3344                       |
| newline .....                        | 3374                       |
| nul.....                             | 3373, 3376                 |
| page backwards.....                  | 3342                       |
| page forward.....                    | 3343                       |
| put from buffer.....                 | 3364-3365                  |
| redraw screen.....                   | 3345                       |
| redraw window .....                  | 3370                       |
| regular expression.....              | 3356                       |
| repeat.....                          | 3353                       |
| repeat find .....                    | 3355, 3363                 |
| repeat substitution .....            | 3349                       |
| replace character .....              | 3365-3366                  |
| replace text with command .....      | 3347                       |
| return to previous context.....      | 3349                       |
| return to previous section .....     | 3350                       |
| reverse case .....                   | 3357                       |
| reverse find character .....         | 3352                       |
| scroll backward.....                 | 3345                       |
| scroll backward by line.....         | 3345                       |
| scroll forward.....                  | 3342                       |
| scroll forward by line.....          | 3342                       |
| search for tagstring .....           | 3346                       |



|                                       |                                          |
|---------------------------------------|------------------------------------------|
| shift left .....                      | 3355                                     |
| shift right .....                     | 3356                                     |
| substitute character .....            | 3366                                     |
| substitute lines .....                | 3366                                     |
| terminate command or input mode ..... | 3346                                     |
| undo .....                            | 3367                                     |
| undo current line .....               | 3368                                     |
| yank .....                            | 3369                                     |
| yank current line .....               | 3370                                     |
| VINTR .....                           | 414                                      |
| virtual processor .....               | 3472                                     |
| VISIT .....                           | 2114, 2161                               |
| visual mode .....                     | 2662                                     |
| VKILL .....                           | 414                                      |
| VLNEXT .....                          | 477                                      |
| VMIN .....                            | 3514                                     |
| vprintf() .....                       | 2189, 2195                               |
| VQUIT .....                           | 414                                      |
| VREPRINT .....                        | 477                                      |
| vscanf() .....                        | 2191, 2196                               |
| vsprintf() .....                      | 2189, 2197                               |
| vsprintf() .....                      | 2189, 2197                               |
| vsscanf() .....                       | 2191, 2198                               |
| VSTART .....                          | 414                                      |
| VSTATUS .....                         | 477                                      |
| VSTOP .....                           | 414                                      |
| VSUSP .....                           | 414                                      |
| vswprintf() .....                     | 2192, 2199                               |
| vswscanf() .....                      | 2194, 2200                               |
| VTDLY .....                           | 416                                      |
| VTIME .....                           | 3514                                     |
| VTn .....                             | 416                                      |
| VWERASE .....                         | 477                                      |
| vwprintf() .....                      | 2192, 2201                               |
| vscanf() .....                        | 2194, 2202                               |
| wait .....                            | 2318, 2339, 3391, 3675, 3723             |
| for process termination .....         | 2209                                     |
| for thread termination .....          | 1633                                     |
| wait() .....                          | 2203, 3532, 3536, 3541-3542, 3544, 3720  |
| waitid() .....                        | 2212, 3542, 3720                         |
| waiting on a condition .....          | 1602                                     |
| waitpid() .....                       | 2203, 2214, 3457, 3461, 3542, 3647, 3720 |
| wall .....                            |                                          |
| rationale for omission .....          | 3712                                     |
| WARNING .....                         | 879                                      |
| warning .....                         |                                          |
| OB .....                              | 9                                        |
| OF .....                              | 9                                        |
| wc .....                              | 3395, 3724                               |
| WCHAR_MAX .....                       | 351, 457                                 |
| WCHAR_MIN .....                       | 351, 457                                 |
| WCONTINUED .....                      | 408, 2203, 2212                          |

|                 |                                                                                 |
|-----------------|---------------------------------------------------------------------------------|
| wcpncpy()       | 2215, 2226                                                                      |
| wcpncpy()       | 2216, 2235                                                                      |
| wcrtomb()       | 2217                                                                            |
| wscasecmp()     | 2219                                                                            |
| wscasecmp_l()   | 2219                                                                            |
| wscat()         | 2221                                                                            |
| wcschr()        | 2222                                                                            |
| wscmp()         | 2223                                                                            |
| wscoll()        | 2224                                                                            |
| wscoll_l()      | 2224                                                                            |
| wscopy()        | 2226                                                                            |
| wscspn()        | 2227                                                                            |
| wcsdup()        | 2228                                                                            |
| wcsftime()      | 2229                                                                            |
| wcslen()        | 2231                                                                            |
| wcsncasecmp()   | 2219, 2232                                                                      |
| wcsncasecmp_l() | 2219, 2232                                                                      |
| wcsncat()       | 2233                                                                            |
| wcsncmp()       | 2234                                                                            |
| wcsncpy()       | 2235                                                                            |
| wcsnlen()       | 2231, 2237                                                                      |
| wcsnrtombs()    | 2238, 2241                                                                      |
| wcspbrk()       | 2239                                                                            |
| wcsrchr()       | 2240                                                                            |
| wcsrtombs()     | 2241                                                                            |
| wcsspn()        | 2243                                                                            |
| wcsstr()        | 2244                                                                            |
| wcstod()        | 2245                                                                            |
| wcstof()        | 2245                                                                            |
| wcstoimax()     | 2249                                                                            |
| wcstok()        | 2250                                                                            |
| wcstol()        | 2252                                                                            |
| wcstold()       | 2245, 2255                                                                      |
| wcstoll()       | 2252, 2256                                                                      |
| wcstombs()      | 2257                                                                            |
| wcstoul()       | 2259                                                                            |
| wcstoull()      | 2259                                                                            |
| wcstoumax()     | 2249, 2262                                                                      |
| wcswcs()        | 3652                                                                            |
| wcswidth()      | 2263                                                                            |
| wcsxfrm()       | 2264                                                                            |
| wcsxfrm_l()     | 2264                                                                            |
| wctob()         | 2266                                                                            |
| wctomb()        | 2267                                                                            |
| wctrans()       | 2269                                                                            |
| wctrans_l()     | 2269                                                                            |
| wctype()        | 2271                                                                            |
| wctype_l()      | 2271                                                                            |
| wcwidth()       | 2273                                                                            |
| WEOF            | 457, 462, 546, 1180, 1182, 1186, 1188, 1190, 1192, 1194, 1196, 1198, 1200, 1202 |
|                 | 1204, 2150, 2152, 2172                                                          |
| WERASE          | 3511                                                                            |

|                                             |                                                                            |
|---------------------------------------------|----------------------------------------------------------------------------|
| WEXITED .....                               | 408, 2212                                                                  |
| WEXITSTATUS .....                           | 358, 408, 2204                                                             |
| we_ .....                                   | 476                                                                        |
| what .....                                  | <b>3398</b>                                                                |
| while loop .....                            | 2346, 3697                                                                 |
| white space .....                           | 103                                                                        |
| who .....                                   | <b>3401</b> , 3725-3726                                                    |
| wide characters .....                       | 129                                                                        |
| wide-character code .....                   | 3483                                                                       |
| wide-character code (C language) .....      | 103                                                                        |
| wide-character input/output functions ..... | 103                                                                        |
| wide-character string .....                 | 103                                                                        |
| wide-oriented stream .....                  | 498                                                                        |
| WIFCONTINUED .....                          | 408, 2204                                                                  |
| WIFEXITED .....                             | 358, 408, 2204                                                             |
| WIFSIGNALED .....                           | 358, 408, 2204                                                             |
| WIFSTOPPED .....                            | 358, 408, 2204, 2209                                                       |
| WINT_MAX .....                              | 351                                                                        |
| WINT_MIN .....                              | 351                                                                        |
| wmemchr() .....                             | <b>2274</b>                                                                |
| wmemcmp() .....                             | <b>2275</b>                                                                |
| wmemcpy() .....                             | <b>2276</b>                                                                |
| wmemmove() .....                            | <b>2277</b>                                                                |
| wmemset() .....                             | <b>2278</b>                                                                |
| WNOHANG .....                               | 358, 408, 1935, 2203, 2212                                                 |
| WNOWAIT .....                               | 408, 2212                                                                  |
| word .....                                  | 104                                                                        |
| word counting .....                         | 3395                                                                       |
| word expansions .....                       | 2327, 3682                                                                 |
| wordexp() .....                             | <b>2279</b> , 3726                                                         |
| wordfree() .....                            | 2279, 3726                                                                 |
| WORD_BIT .....                              | 280-281                                                                    |
| working directory .....                     | 104                                                                        |
| worldwide portability interface .....       | 104                                                                        |
| wprintf() .....                             | 982, 2284                                                                  |
| WRDE_ .....                                 | 476                                                                        |
| WRDE_APPEND .....                           | 464, 2280                                                                  |
| WRDE_BADCHAR .....                          | 464, 2281                                                                  |
| WRDE_BADVAL .....                           | 464, 2281                                                                  |
| WRDE_CMDSUB .....                           | 464, 2281                                                                  |
| WRDE_DOOFFS .....                           | 464, 2280                                                                  |
| WRDE_NOCMD .....                            | 464, 2280                                                                  |
| WRDE_NOSPACE .....                          | 464, 2281                                                                  |
| WRDE_REUSE .....                            | 464, 2280                                                                  |
| WRDE_SHOWERR .....                          | 464, 2280                                                                  |
| WRDE_SYNTAX .....                           | 464, 2281                                                                  |
| WRDE_UNDEF .....                            | 464, 2280                                                                  |
| write .....                                 | 104, 3405, 3724-3725                                                       |
| write lock .....                            | 3603                                                                       |
| write to a file .....                       | 2289                                                                       |
| write() .....                               | <b>2285</b> , 3456-3457, 3511, 3532, 3541-3542, 3544, 3554-3556, 3560-3561 |
| .....                                       | 3648, 3721                                                                 |

|                                |                            |
|--------------------------------|----------------------------|
| writev()                       | 2294                       |
| writing data                   | 3512                       |
| wscanf()                       | 992, 2296                  |
| WSTOPPED                       | 408, 2212                  |
| WSTOPSIG                       | 358, 408, 2204             |
| WTERMSIG                       | 358, 408, 2204             |
| WUNTRACED                      | 358, 408, 2203, 2209, 3457 |
| W_OK                           | 440                        |
| xargs                          | 3408, 3723                 |
| XOPEN_UNIX                     | 19, 29                     |
| XOPEN_UUCP                     | 19, 29                     |
| XSI                            | 12, 104, 3472              |
| conformance                    | 15, 19, 105                |
| XSI interprocess communication | 501                        |
| XSI IPC                        | 3546                       |
| XSI options groups             | 22                         |
| XSI STREAMS                    | 25                         |
| XSI system interfaces          |                            |
| conformance                    | 19                         |
| XSI_C_LANG_SUPPORT             | 3745                       |
| XSI_DBM                        | 3746                       |
| XSI_DEVICE_IO                  | 3746                       |
| XSI_DEVICE_SPECIFIC            | 3746                       |
| XSI_FILE_SYSTEM                | 3746                       |
| XSI_IPC                        | 3746                       |
| XSI_JUMP                       | 3746                       |
| XSI_MATH                       | 3746                       |
| XSI_MULTI_PROCESS              | 3746                       |
| XSI_SIGNALS                    | 3746                       |
| XSI_SINGLE_PROCESS             | 3746                       |
| XSI_SYSTEM_DATABASE            | 3746                       |
| XSI_SYSTEM_LOGGING             | 3746                       |
| XSI_THREADS_EXT                | 3746                       |
| XSI_TIMERS                     | 3746                       |
| XSI_USER_GROUPS                | 3746                       |
| XSI_WIDE_CHAR                  | 3746                       |
| XSR                            | 13                         |
| X_OK                           | 440, 567                   |
| y0()                           | 2297                       |
| y1()                           | 2297                       |
| yacc                           | 3415, 3726, 3728           |
| algorithms                     | 3426                       |
| code file                      | 3417                       |
| completing the program         | 3426                       |
| conflicts                      | 3424                       |
| debugging the parser           | 3426                       |
| declarations section           | 3418                       |
| description file               | 3417                       |
| error handling                 | 3424                       |
| grammar rules                  | 3420                       |
| header file                    | 3417                       |
| input grammar                  | 3422                       |

|                                        |             |
|----------------------------------------|-------------|
| input language.....                    | 3417        |
| interface to the lexical analyzer..... | 3425        |
| lexical structure of the grammar.....  | 3418        |
| library.....                           | 3426        |
| limits.....                            | 3427        |
| programs section.....                  | 3421        |
| YESEXPR.....                           | 267         |
| yn().....                              | 2297        |
| zcat.....                              | <b>3432</b> |
| zombie process.....                    | 105, 549    |

